| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ICC_ColorSpace.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/awt/color/ColorSpace.html)   [**NEXT CLASS**](http://docs.google.com/java/awt/color/ICC_Profile.html) | [**FRAMES**](http://docs.google.com/index.html?java/awt/color/ICC_ColorSpace.html)    [**NO FRAMES**](http://docs.google.com/ICC_ColorSpace.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#2et92p0) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: FIELD | [CONSTR](#2s8eyo1) | [METHOD](#3rdcrjn) |

## **java.awt.color**

Class ICC\_ColorSpace

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 [java.awt.color.ColorSpace](http://docs.google.com/java/awt/color/ColorSpace.html)  
 **java.awt.color.ICC\_ColorSpace**

**All Implemented Interfaces:** [Serializable](http://docs.google.com/java/io/Serializable.html)

public class **ICC\_ColorSpace**extends [ColorSpace](http://docs.google.com/java/awt/color/ColorSpace.html)

The ICC\_ColorSpace class is an implementation of the abstract ColorSpace class. This representation of device independent and device dependent color spaces is based on the International Color Consortium Specification ICC.1:2001-12, File Format for Color Profiles (see <http://www.color.org>).

Typically, a Color or ColorModel would be associated with an ICC Profile which is either an input, display, or output profile (see the ICC specification). There are other types of ICC Profiles, e.g. abstract profiles, device link profiles, and named color profiles, which do not contain information appropriate for representing the color space of a color, image, or device (see ICC\_Profile). Attempting to create an ICC\_ColorSpace object from an inappropriate ICC Profile is an error.

ICC Profiles represent transformations from the color space of the profile (e.g. a monitor) to a Profile Connection Space (PCS). Profiles of interest for tagging images or colors have a PCS which is one of the device independent spaces (one CIEXYZ space and two CIELab spaces) defined in the ICC Profile Format Specification. Most profiles of interest either have invertible transformations or explicitly specify transformations going both directions. Should an ICC\_ColorSpace object be used in a way requiring a conversion from PCS to the profile's native space and there is inadequate data to correctly perform the conversion, the ICC\_ColorSpace object will produce output in the specified type of color space (e.g. TYPE\_RGB, TYPE\_CMYK, etc.), but the specific color values of the output data will be undefined.

The details of this class are not important for simple applets, which draw in a default color space or manipulate and display imported images with a known color space. At most, such applets would need to get one of the default color spaces via ColorSpace.getInstance().

**See Also:**[ColorSpace](http://docs.google.com/java/awt/color/ColorSpace.html), [ICC\_Profile](http://docs.google.com/java/awt/color/ICC_Profile.html), [Serialized Form](http://docs.google.com/serialized-form.html#java.awt.color.ICC_ColorSpace)

| **Field Summary** | |
| --- | --- |

| **Fields inherited from class java.awt.color.**[**ColorSpace**](http://docs.google.com/java/awt/color/ColorSpace.html) |
| --- |
| [CS\_CIEXYZ](http://docs.google.com/java/awt/color/ColorSpace.html#CS_CIEXYZ), [CS\_GRAY](http://docs.google.com/java/awt/color/ColorSpace.html#CS_GRAY), [CS\_LINEAR\_RGB](http://docs.google.com/java/awt/color/ColorSpace.html#CS_LINEAR_RGB), [CS\_PYCC](http://docs.google.com/java/awt/color/ColorSpace.html#CS_PYCC), [CS\_sRGB](http://docs.google.com/java/awt/color/ColorSpace.html#CS_sRGB), [TYPE\_2CLR](http://docs.google.com/java/awt/color/ColorSpace.html#TYPE_2CLR), [TYPE\_3CLR](http://docs.google.com/java/awt/color/ColorSpace.html#TYPE_3CLR), [TYPE\_4CLR](http://docs.google.com/java/awt/color/ColorSpace.html#TYPE_4CLR), [TYPE\_5CLR](http://docs.google.com/java/awt/color/ColorSpace.html#TYPE_5CLR), [TYPE\_6CLR](http://docs.google.com/java/awt/color/ColorSpace.html#TYPE_6CLR), [TYPE\_7CLR](http://docs.google.com/java/awt/color/ColorSpace.html#TYPE_7CLR), [TYPE\_8CLR](http://docs.google.com/java/awt/color/ColorSpace.html#TYPE_8CLR), [TYPE\_9CLR](http://docs.google.com/java/awt/color/ColorSpace.html#TYPE_9CLR), [TYPE\_ACLR](http://docs.google.com/java/awt/color/ColorSpace.html#TYPE_ACLR), [TYPE\_BCLR](http://docs.google.com/java/awt/color/ColorSpace.html#TYPE_BCLR), [TYPE\_CCLR](http://docs.google.com/java/awt/color/ColorSpace.html#TYPE_CCLR), [TYPE\_CMY](http://docs.google.com/java/awt/color/ColorSpace.html#TYPE_CMY), [TYPE\_CMYK](http://docs.google.com/java/awt/color/ColorSpace.html#TYPE_CMYK), [TYPE\_DCLR](http://docs.google.com/java/awt/color/ColorSpace.html#TYPE_DCLR), [TYPE\_ECLR](http://docs.google.com/java/awt/color/ColorSpace.html#TYPE_ECLR), [TYPE\_FCLR](http://docs.google.com/java/awt/color/ColorSpace.html#TYPE_FCLR), [TYPE\_GRAY](http://docs.google.com/java/awt/color/ColorSpace.html#TYPE_GRAY), [TYPE\_HLS](http://docs.google.com/java/awt/color/ColorSpace.html#TYPE_HLS), [TYPE\_HSV](http://docs.google.com/java/awt/color/ColorSpace.html#TYPE_HSV), [TYPE\_Lab](http://docs.google.com/java/awt/color/ColorSpace.html#TYPE_Lab), [TYPE\_Luv](http://docs.google.com/java/awt/color/ColorSpace.html#TYPE_Luv), [TYPE\_RGB](http://docs.google.com/java/awt/color/ColorSpace.html#TYPE_RGB), [TYPE\_XYZ](http://docs.google.com/java/awt/color/ColorSpace.html#TYPE_XYZ), [TYPE\_YCbCr](http://docs.google.com/java/awt/color/ColorSpace.html#TYPE_YCbCr), [TYPE\_Yxy](http://docs.google.com/java/awt/color/ColorSpace.html#TYPE_Yxy) |

| **Constructor Summary** | |
| --- | --- |
| [**ICC\_ColorSpace**](http://docs.google.com/java/awt/color/ICC_ColorSpace.html#ICC_ColorSpace(java.awt.color.ICC_Profile))([ICC\_Profile](http://docs.google.com/java/awt/color/ICC_Profile.html) profile)            Constructs a new ICC\_ColorSpace from an ICC\_Profile object. |

| **Method Summary** | |
| --- | --- |
| float[] | [**fromCIEXYZ**](http://docs.google.com/java/awt/color/ICC_ColorSpace.html#fromCIEXYZ(float%5B%5D))(float[] colorvalue)            Transforms a color value assumed to be in the CS\_CIEXYZ conversion color space into this ColorSpace. |
| float[] | [**fromRGB**](http://docs.google.com/java/awt/color/ICC_ColorSpace.html#fromRGB(float%5B%5D))(float[] rgbvalue)            Transforms a color value assumed to be in the default CS\_sRGB color space into this ColorSpace. |
| float | [**getMaxValue**](http://docs.google.com/java/awt/color/ICC_ColorSpace.html#getMaxValue(int))(int component)            Returns the maximum normalized color component value for the specified component. |
| float | [**getMinValue**](http://docs.google.com/java/awt/color/ICC_ColorSpace.html#getMinValue(int))(int component)            Returns the minimum normalized color component value for the specified component. |
| [ICC\_Profile](http://docs.google.com/java/awt/color/ICC_Profile.html) | [**getProfile**](http://docs.google.com/java/awt/color/ICC_ColorSpace.html#getProfile())()            Returns the ICC\_Profile for this ICC\_ColorSpace. |
| float[] | [**toCIEXYZ**](http://docs.google.com/java/awt/color/ICC_ColorSpace.html#toCIEXYZ(float%5B%5D))(float[] colorvalue)            Transforms a color value assumed to be in this ColorSpace into the CS\_CIEXYZ conversion color space. |
| float[] | [**toRGB**](http://docs.google.com/java/awt/color/ICC_ColorSpace.html#toRGB(float%5B%5D))(float[] colorvalue)            Transforms a color value assumed to be in this ColorSpace into a value in the default CS\_sRGB color space. |

| **Methods inherited from class java.awt.color.**[**ColorSpace**](http://docs.google.com/java/awt/color/ColorSpace.html) |
| --- |
| [getInstance](http://docs.google.com/java/awt/color/ColorSpace.html#getInstance(int)), [getName](http://docs.google.com/java/awt/color/ColorSpace.html#getName(int)), [getNumComponents](http://docs.google.com/java/awt/color/ColorSpace.html#getNumComponents()), [getType](http://docs.google.com/java/awt/color/ColorSpace.html#getType()), [isCS\_sRGB](http://docs.google.com/java/awt/color/ColorSpace.html#isCS_sRGB()) |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [toString](http://docs.google.com/java/lang/Object.html#toString()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Constructor Detail** |
| --- |

### ICC\_ColorSpace

public **ICC\_ColorSpace**([ICC\_Profile](http://docs.google.com/java/awt/color/ICC_Profile.html) profile)

Constructs a new ICC\_ColorSpace from an ICC\_Profile object.

**Parameters:**profile - the specified ICC\_Profile object **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if profile is inappropriate for representing a ColorSpace.

| **Method Detail** |
| --- |

### getProfile

public [ICC\_Profile](http://docs.google.com/java/awt/color/ICC_Profile.html) **getProfile**()

Returns the ICC\_Profile for this ICC\_ColorSpace.

**Returns:**the ICC\_Profile for this ICC\_ColorSpace.

### toRGB

public float[] **toRGB**(float[] colorvalue)

Transforms a color value assumed to be in this ColorSpace into a value in the default CS\_sRGB color space.

This method transforms color values using algorithms designed to produce the best perceptual match between input and output colors. In order to do colorimetric conversion of color values, you should use the toCIEXYZ method of this color space to first convert from the input color space to the CS\_CIEXYZ color space, and then use the fromCIEXYZ method of the CS\_sRGB color space to convert from CS\_CIEXYZ to the output color space. See [toCIEXYZ](http://docs.google.com/java/awt/color/ICC_ColorSpace.html#toCIEXYZ(float%5B%5D)) and [fromCIEXYZ](http://docs.google.com/java/awt/color/ICC_ColorSpace.html#fromCIEXYZ(float%5B%5D)) for further information.

**Specified by:**[toRGB](http://docs.google.com/java/awt/color/ColorSpace.html#toRGB(float%5B%5D)) in class [ColorSpace](http://docs.google.com/java/awt/color/ColorSpace.html) **Parameters:**colorvalue - a float array with length of at least the number of components in this ColorSpace. **Returns:**a float array of length 3. **Throws:** [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - if array length is not at least the number of components in this ColorSpace.

### fromRGB

public float[] **fromRGB**(float[] rgbvalue)

Transforms a color value assumed to be in the default CS\_sRGB color space into this ColorSpace.

This method transforms color values using algorithms designed to produce the best perceptual match between input and output colors. In order to do colorimetric conversion of color values, you should use the toCIEXYZ method of the CS\_sRGB color space to first convert from the input color space to the CS\_CIEXYZ color space, and then use the fromCIEXYZ method of this color space to convert from CS\_CIEXYZ to the output color space. See [toCIEXYZ](http://docs.google.com/java/awt/color/ICC_ColorSpace.html#toCIEXYZ(float%5B%5D)) and [fromCIEXYZ](http://docs.google.com/java/awt/color/ICC_ColorSpace.html#fromCIEXYZ(float%5B%5D)) for further information.

**Specified by:**[fromRGB](http://docs.google.com/java/awt/color/ColorSpace.html#fromRGB(float%5B%5D)) in class [ColorSpace](http://docs.google.com/java/awt/color/ColorSpace.html) **Parameters:**rgbvalue - a float array with length of at least 3. **Returns:**a float array with length equal to the number of components in this ColorSpace. **Throws:** [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - if array length is not at least 3.

### toCIEXYZ

public float[] **toCIEXYZ**(float[] colorvalue)

Transforms a color value assumed to be in this ColorSpace into the CS\_CIEXYZ conversion color space.

This method transforms color values using relative colorimetry, as defined by the ICC Specification. This means that the XYZ values returned by this method are represented relative to the D50 white point of the CS\_CIEXYZ color space. This representation is useful in a two-step color conversion process in which colors are transformed from an input color space to CS\_CIEXYZ and then to an output color space. This representation is not the same as the XYZ values that would be measured from the given color value by a colorimeter. A further transformation is necessary to compute the XYZ values that would be measured using current CIE recommended practices. The paragraphs below explain this in more detail.

The ICC standard uses a device independent color space (DICS) as the mechanism for converting color from one device to another device. In this architecture, colors are converted from the source device's color space to the ICC DICS and then from the ICC DICS to the destination device's color space. The ICC standard defines device profiles which contain transforms which will convert between a device's color space and the ICC DICS. The overall conversion of colors from a source device to colors of a destination device is done by connecting the device-to-DICS transform of the profile for the source device to the DICS-to-device transform of the profile for the destination device. For this reason, the ICC DICS is commonly referred to as the profile connection space (PCS). The color space used in the methods toCIEXYZ and fromCIEXYZ is the CIEXYZ PCS defined by the ICC Specification. This is also the color space represented by ColorSpace.CS\_CIEXYZ.

The XYZ values of a color are often represented as relative to some white point, so the actual meaning of the XYZ values cannot be known without knowing the white point of those values. This is known as relative colorimetry. The PCS uses a white point of D50, so the XYZ values of the PCS are relative to D50. For example, white in the PCS will have the XYZ values of D50, which is defined to be X=.9642, Y=1.000, and Z=0.8249. This white point is commonly used for graphic arts applications, but others are often used in other applications.

To quantify the color characteristics of a device such as a printer or monitor, measurements of XYZ values for particular device colors are typically made. For purposes of this discussion, the term device XYZ values is used to mean the XYZ values that would be measured from device colors using current CIE recommended practices.

Converting between device XYZ values and the PCS XYZ values returned by this method corresponds to converting between the device's color space, as represented by CIE colorimetric values, and the PCS. There are many factors involved in this process, some of which are quite subtle. The most important, however, is the adjustment made to account for differences between the device's white point and the white point of the PCS. There are many techniques for doing this and it is the subject of much current research and controversy. Some commonly used methods are XYZ scaling, the von Kries transform, and the Bradford transform. The proper method to use depends upon each particular application.

The simplest method is XYZ scaling. In this method each device XYZ value is converted to a PCS XYZ value by multiplying it by the ratio of the PCS white point (D50) to the device white point.

Xd, Yd, Zd are the device XYZ values  
 Xdw, Ydw, Zdw are the device XYZ white point values  
 Xp, Yp, Zp are the PCS XYZ values  
 Xd50, Yd50, Zd50 are the PCS XYZ white point values  
   
 Xp = Xd \* (Xd50 / Xdw)  
 Yp = Yd \* (Yd50 / Ydw)  
 Zp = Zd \* (Zd50 / Zdw)

Conversion from the PCS to the device would be done by inverting these equations:

Xd = Xp \* (Xdw / Xd50)  
 Yd = Yp \* (Ydw / Yd50)  
 Zd = Zp \* (Zdw / Zd50)

Note that the media white point tag in an ICC profile is not the same as the device white point. The media white point tag is expressed in PCS values and is used to represent the difference between the XYZ of device illuminant and the XYZ of the device media when measured under that illuminant. The device white point is expressed as the device XYZ values corresponding to white displayed on the device. For example, displaying the RGB color (1.0, 1.0, 1.0) on an sRGB device will result in a measured device XYZ value of D65. This will not be the same as the media white point tag XYZ value in the ICC profile for an sRGB device.

**Specified by:**[toCIEXYZ](http://docs.google.com/java/awt/color/ColorSpace.html#toCIEXYZ(float%5B%5D)) in class [ColorSpace](http://docs.google.com/java/awt/color/ColorSpace.html) **Parameters:**colorvalue - a float array with length of at least the number of components in this ColorSpace. **Returns:**a float array of length 3. **Throws:** [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - if array length is not at least the number of components in this ColorSpace.

### fromCIEXYZ

public float[] **fromCIEXYZ**(float[] colorvalue)

Transforms a color value assumed to be in the CS\_CIEXYZ conversion color space into this ColorSpace.

This method transforms color values using relative colorimetry, as defined by the ICC Specification. This means that the XYZ argument values taken by this method are represented relative to the D50 white point of the CS\_CIEXYZ color space. This representation is useful in a two-step color conversion process in which colors are transformed from an input color space to CS\_CIEXYZ and then to an output color space. The color values returned by this method are not those that would produce the XYZ value passed to the method when measured by a colorimeter. If you have XYZ values corresponding to measurements made using current CIE recommended practices, they must be converted to D50 relative values before being passed to this method. The paragraphs below explain this in more detail.

The ICC standard uses a device independent color space (DICS) as the mechanism for converting color from one device to another device. In this architecture, colors are converted from the source device's color space to the ICC DICS and then from the ICC DICS to the destination device's color space. The ICC standard defines device profiles which contain transforms which will convert between a device's color space and the ICC DICS. The overall conversion of colors from a source device to colors of a destination device is done by connecting the device-to-DICS transform of the profile for the source device to the DICS-to-device transform of the profile for the destination device. For this reason, the ICC DICS is commonly referred to as the profile connection space (PCS). The color space used in the methods toCIEXYZ and fromCIEXYZ is the CIEXYZ PCS defined by the ICC Specification. This is also the color space represented by ColorSpace.CS\_CIEXYZ.

The XYZ values of a color are often represented as relative to some white point, so the actual meaning of the XYZ values cannot be known without knowing the white point of those values. This is known as relative colorimetry. The PCS uses a white point of D50, so the XYZ values of the PCS are relative to D50. For example, white in the PCS will have the XYZ values of D50, which is defined to be X=.9642, Y=1.000, and Z=0.8249. This white point is commonly used for graphic arts applications, but others are often used in other applications.

To quantify the color characteristics of a device such as a printer or monitor, measurements of XYZ values for particular device colors are typically made. For purposes of this discussion, the term device XYZ values is used to mean the XYZ values that would be measured from device colors using current CIE recommended practices.

Converting between device XYZ values and the PCS XYZ values taken as arguments by this method corresponds to converting between the device's color space, as represented by CIE colorimetric values, and the PCS. There are many factors involved in this process, some of which are quite subtle. The most important, however, is the adjustment made to account for differences between the device's white point and the white point of the PCS. There are many techniques for doing this and it is the subject of much current research and controversy. Some commonly used methods are XYZ scaling, the von Kries transform, and the Bradford transform. The proper method to use depends upon each particular application.

The simplest method is XYZ scaling. In this method each device XYZ value is converted to a PCS XYZ value by multiplying it by the ratio of the PCS white point (D50) to the device white point.

Xd, Yd, Zd are the device XYZ values  
 Xdw, Ydw, Zdw are the device XYZ white point values  
 Xp, Yp, Zp are the PCS XYZ values  
 Xd50, Yd50, Zd50 are the PCS XYZ white point values  
   
 Xp = Xd \* (Xd50 / Xdw)  
 Yp = Yd \* (Yd50 / Ydw)  
 Zp = Zd \* (Zd50 / Zdw)

Conversion from the PCS to the device would be done by inverting these equations:

Xd = Xp \* (Xdw / Xd50)  
 Yd = Yp \* (Ydw / Yd50)  
 Zd = Zp \* (Zdw / Zd50)

Note that the media white point tag in an ICC profile is not the same as the device white point. The media white point tag is expressed in PCS values and is used to represent the difference between the XYZ of device illuminant and the XYZ of the device media when measured under that illuminant. The device white point is expressed as the device XYZ values corresponding to white displayed on the device. For example, displaying the RGB color (1.0, 1.0, 1.0) on an sRGB device will result in a measured device XYZ value of D65. This will not be the same as the media white point tag XYZ value in the ICC profile for an sRGB device.

**Specified by:**[fromCIEXYZ](http://docs.google.com/java/awt/color/ColorSpace.html#fromCIEXYZ(float%5B%5D)) in class [ColorSpace](http://docs.google.com/java/awt/color/ColorSpace.html) **Parameters:**colorvalue - a float array with length of at least 3. **Returns:**a float array with length equal to the number of components in this ColorSpace. **Throws:** [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - if array length is not at least 3.

### getMinValue

public float **getMinValue**(int component)

Returns the minimum normalized color component value for the specified component. For TYPE\_XYZ spaces, this method returns minimum values of 0.0 for all components. For TYPE\_Lab spaces, this method returns 0.0 for L and -128.0 for a and b components. This is consistent with the encoding of the XYZ and Lab Profile Connection Spaces in the ICC specification. For all other types, this method returns 0.0 for all components. When using an ICC\_ColorSpace with a profile that requires different minimum component values, it is necessary to subclass this class and override this method.

**Overrides:**[getMinValue](http://docs.google.com/java/awt/color/ColorSpace.html#getMinValue(int)) in class [ColorSpace](http://docs.google.com/java/awt/color/ColorSpace.html) **Parameters:**component - The component index. **Returns:**The minimum normalized component value. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if component is less than 0 or greater than numComponents - 1.**Since:** 1.4

### getMaxValue

public float **getMaxValue**(int component)

Returns the maximum normalized color component value for the specified component. For TYPE\_XYZ spaces, this method returns maximum values of 1.0 + (32767.0 / 32768.0) for all components. For TYPE\_Lab spaces, this method returns 100.0 for L and 127.0 for a and b components. This is consistent with the encoding of the XYZ and Lab Profile Connection Spaces in the ICC specification. For all other types, this method returns 1.0 for all components. When using an ICC\_ColorSpace with a profile that requires different maximum component values, it is necessary to subclass this class and override this method.

**Overrides:**[getMaxValue](http://docs.google.com/java/awt/color/ColorSpace.html#getMaxValue(int)) in class [ColorSpace](http://docs.google.com/java/awt/color/ColorSpace.html) **Parameters:**component - The component index. **Returns:**The maximum normalized component value. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if component is less than 0 or greater than numComponents - 1.**Since:** 1.4

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ICC_ColorSpace.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/awt/color/ColorSpace.html)   [**NEXT CLASS**](http://docs.google.com/java/awt/color/ICC_Profile.html) | [**FRAMES**](http://docs.google.com/index.html?java/awt/color/ICC_ColorSpace.html)    [**NO FRAMES**](http://docs.google.com/ICC_ColorSpace.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#2et92p0) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: FIELD | [CONSTR](#2s8eyo1) | [METHOD](#3rdcrjn) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).