

# Windows Color System

Article01/24/2023

Overview of the Windows Color System technology.

To develop Windows Color System, you need these headers:

- [wcsplugin.h](#)

For programming guidance for this technology, see:

- [Windows Color System](#)

## Enumerations

<b>BMFORMAT</b>  The values of the <b>BMFORMAT</b> enumerated type are used by several WCS functions to indicate the format that particular bitmaps are in.
<b>COLORDATATYPE</b>  Used by WCS functions to indicate the data type of vector content.
<b>COLORPROFILESUBTYPE</b>  Specifies the subtype of the color profile.
<b>COLORPROFILETYPE</b>  Specifies the type of color profile.
<b>COLORTYPE</b>  The values of the <b>COLORTYPE</b> enumeration are used by several WCS functions. Variables of type <b>COLOR</b> are defined in the color spaces enumerated by the <b>COLORTYPE</b> enumeration.
<b>WCS_PROFILE_MANAGEMENT_SCOPE</b>  Specifies the scope of a profile management operation, such as associating a profile with a device.

## Functions

<a href="#">AssociateColorProfileWithDeviceA</a>	Associates a specified color profile with a specified device. (ANSI)
<a href="#">AssociateColorProfileWithDeviceW</a>	Associates a specified color profile with a specified device. (Unicode)
<a href="#">CheckBitmapBits</a>	Checks whether the pixels in a specified bitmap lie within the output <a href="#">gamut</a> of a specified transform.
<a href="#">CheckColors</a>	Determines whether the colors in an array lie within the output <a href="#">gamut</a> of a specified transform.
<a href="#">CheckColorsInGamut</a>	The CheckColorsInGamut function determines whether a specified set of RGB triples lies in the output gamut of a specified device. The RGB triples are interpreted in the input logical color space.
<a href="#">CloseColorProfile</a>	Closes an open profile handle.
<a href="#">CMCheckColors</a>	Determines whether given colors lie within the output <a href="#">gamut</a> of a specified transform.
<a href="#">CMCheckColorsInGamut</a>	Determines whether specified RGB triples lie in the output <a href="#">gamut</a> of a specified transform.
<a href="#">CMCheckRBGs</a>	Checks bitmap colors against an output gamut.
<a href="#">CMConvertColorNameToIndex</a>	Converts color names in a named color space to index numbers in a color profile.
<a href="#">CMConvertIndexToColorName</a>	Transforms indices in a color space to an array of names in a named color space. (CMConvertIndexToColorName)

## [CMCreateDeviceLinkProfile](#)

Creates a [device link profile](#) in the format specified by the International Color Consortium in its ICC Profile Format Specification.

## [CMCreateMultiProfileTransform](#)

Accepts an array of profiles or a single [device link profile](#) and creates a color transform. This transform is a mapping from the color space specified by the first profile to that of the second profile and so on to the last one.

## [CMCreateProfile](#)

Creates a display color profile from a [LOGCOLORSPACEA](#) structure.

## [CMCreateProfileW](#)

Creates a display color profile from a [LOGCOLORSPACEW](#) structure.

## [CMCreateTransform](#)

Deprecated. There is no replacement API because this one was no longer being used. Developers of alternate CMM modules are not required to implement it. (CMCreateTransform)

## [CMCreateTransformExt](#)

Creates a color transform that maps from an input [LOGCOLORSPACEA](#) to an optional target space and then to an output device, using a set of flags that define how the transform should be created.

## [CMCreateTransformExtW](#)

Creates a color transform that maps from an input [LOGCOLORSPACEW](#) to an optional target space and then to an output device, using a set of flags that define how the transform should be created.

## [CMCreateTransformW](#)

Deprecated. There is no replacement API because this one was no longer being used. Developers of alternate CMM modules are not required to implement it. (CMCreateTransformW)

## [CMDeleteTransform](#)

Deletes a specified color transform, and frees any memory associated with it.

## [CMGetInfo](#)

Retrieves various information about the color management module (CMM).

<p><a href="#">CMGetNamedProfileInfo</a></p> <p>Retrieves information about the specified named color profile.</p>
<p><a href="#">CMGetPS2ColorRenderingDictionary</a></p> <p>CMGetPS2ColorRenderingDictionary and the additional parameters associated with it are to be determined.</p>
<p><a href="#">CMGetPS2ColorRenderingIntent</a></p> <p>Retrieves the PostScript Level 2 color <a href="#">rendering intent</a> from a profile.</p>
<p><a href="#">CMGetPS2ColorSpaceArray</a></p> <p>CMGetPS2ColorSpaceArray and the parameters, returns, and remarks associated with it are to be determined.</p>
<p><a href="#">CMIsProfileValid</a></p> <p>Reports whether the given profile is a valid ICC profile that can be used for color management.</p>
<p><a href="#">CMTranslateColors</a></p> <p>Translates an array of colors from a source <a href="#">color space</a> to a destination color space using a color transform.</p>
<p><a href="#">CMTranslateRGB</a></p> <p>Translates an application-supplied RGBQuad into the device <a href="#">color space</a>.</p>
<p><a href="#">CMTranslateRBGs</a></p> <p>Translates a bitmap from one <a href="#">color space</a> to another using a color transform.</p>
<p><a href="#">CMTranslateRBGsExt</a></p> <p>Translates a bitmap from one defined format into a different defined format and calls a callback function periodically, if one is specified, to report progress and permit the calling application to terminate the translation.</p>
<p><a href="#">CMYK</a></p> <p>The CMYK macro creates a CMYK color value by combining the specified cyan, magenta, yellow, and black values.</p>

## [ColorCorrectPalette](#)

The ColorCorrectPalette function corrects the entries of a palette using the WCS 1.0 parameters in the specified device context.

## [ColorimetricToDeviceColors](#)

Returns the appropriate XYZ colors in response to the specified number of colors, channels, device colors and the proprietary plug-in algorithms. (IDeviceModelPlugIn.ColorimetricToDeviceColors)

## [ColorimetricToDeviceColorsWithBlack](#)

Returns the appropriate device colors in response to the incoming number of colors, channels, black information, Commission Internationale l'Eclairge XYZ (CIEXYZ) colors and the proprietary plug-in algorithms.

## [ColorMatchToTarget](#)

The ColorMatchToTarget function enables you to preview colors as they would appear on the target device.

## [ColorProfileAddDisplayAssociation](#)

ColorProfileAddDisplayAssociation associates an installed color profile with a specified display in the given scope.

## [ColorProfileGetDisplayDefault](#)

ColorProfileGetDisplayDefault gets the default color profile for a given display in the specified scope.

## [ColorProfileGetDisplayList](#)

ColorProfileGetDisplayList retrieves the list of profiles associated with a given display in the specified scope.

## [ColorProfileGetDisplayUserScope](#)

ColorProfileGetDisplayUserScope gets the currently selected color profile scope of the provided display - either user or system.

## [ColorProfileRemoveDisplayAssociation](#)

ColorProfileRemoveDisplayAssociation disassociates an installed color profile from a specified display in the given scope.

## [ColorProfileSetDisplayDefaultAssociation](#)

ColorProfileSetDisplayDefaultAssociation sets an installed color profile as the default profile for a specified display in the given scope.

## [ConvertColorNameToIndex](#)

Converts color names in a named color space to index numbers in an International Color Consortium (ICC) color profile.

## [ConvertIndexToColorName](#)

Transforms indices in a color space to an array of names in a named color space.  
([ConvertIndexToColorName](#))

## [CreateColorSpaceA](#)

The CreateColorSpace function creates a logical color space. (ANSI)

## [CreateColorSpaceW](#)

The CreateColorSpace function creates a logical color space. (Unicode)

## [CreateColorTransformA](#)

Creates a color transform that applications can use to perform color management. (ANSI)

## [CreateColorTransformW](#)

Creates a color transform that applications can use to perform color management. (Unicode)

## [CreateDeviceLinkProfile](#)

Creates an International Color Consortium (ICC) *device link profile* from a set of color profiles, using the specified intents.

## [CreateMultiProfileTransform](#)

Accepts an array of profiles or a single [device link profile](#) and creates a color transform that applications can use to perform color mapping.

## [CreateProfileFromLogColorSpaceA](#)

Converts a logical [color space](#) to a [device profile](#). (ANSI)

## [CreateProfileFromLogColorSpaceW](#)

Converts a logical [color space](#) to a [device profile](#). (Unicode)

## [DeleteColorSpace](#)

The DeleteColorSpace function removes and destroys a specified color space.

## [DeleteColorTransform](#)

Deletes a given color transform.

## [DeviceToColorimetricColors](#)

Returns the appropriate XYZ colors in response to the specified number of colors, channels, device colors and the proprietary plug-in algorithms. (IDeviceModelPlugIn.DeviceToColorimetricColors)

## [DisassociateColorProfileFromDeviceA](#)

Disassociates a specified color profile with a specified device on a specified computer. (ANSI)

## [DisassociateColorProfileFromDeviceW](#)

Disassociates a specified color profile with a specified device on a specified computer. (Unicode)

## [EnumColorProfilesA](#)

Enumerates all the profiles satisfying the given enumeration criteria. (ANSI)

## [EnumColorProfilesW](#)

Enumerates all the profiles satisfying the given enumeration criteria. (Unicode)

## [EnumICMProfilesA](#)

The EnumICMProfiles function enumerates the different output color profiles that the system supports for a given device context. (ANSI)

## [EnumICMProfilesW](#)

The EnumICMProfiles function enumerates the different output color profiles that the system supports for a given device context. (Unicode)

## [GetCMMInfo](#)

Retrieves various information about the color management module (CMM) that created the specified color transform.

## [GetColorDirectoryA](#)

Retrieves the path of the Windows COLOR directory on a specified machine. (ANSI)

## [GetColorDirectoryW](#)

Retrieves the path of the Windows COLOR directory on a specified machine. (Unicode)

## [GetColorProfileElement](#)

Copies data from a specified tagged profile element of a specified color profile into a buffer.

## [GetColorProfileElementTag](#)

Retrieves the tag name specified by *dwIndex* in the tag table of a given International Color Consortium (ICC) color profile, where *dwIndex* is a one-based index into that table.

## [GetColorProfileFromHandle](#)

Given a handle to an open color profile, the **GetColorProfileFromHandle** function copies the contents of the profile into a buffer supplied by the application. If the handle is a Windows Color System (WCS) handle, then the DMP is returned and the CAMP and GMMP associated with the HPROFILE are ignored.

## [GetColorProfileHeader](#)

Retrieves or derives ICC header structure from either ICC color profile or WCS XML profile. Drivers and applications should assume returning **TRUE** only indicates that a properly structured header is returned. Each tag will still need to be validated independently using either legacy ICM2 APIs or XML schema APIs.

## [GetColorSpace](#)

The GetColorSpace function retrieves the handle to the input color space from a specified device context.

## [GetCountColorProfileElements](#)

Retrieves the number of tagged elements in a given color profile.

## [GetCValue](#)

The GetCValue macro retrieves the cyan color value from a CMYK color value.

## [GetDeviceGammaRamp](#)

The GetDeviceGammaRamp function gets the gamma ramp on direct color display boards having drivers that support downloadable gamma ramps in hardware.

## [GetGamutBoundaryMesh](#)

Returns the triangular mesh from the plug-in. This function is used to compute the GamutBoundaryDescription.

## [GetGamutBoundaryMeshSize](#)

Returns the required data structure sizes for the GetGamutBoundaryMesh function.

## [GetICMProfileA](#)

The GetICMProfile function retrieves the file name of the current output color profile for a specified device context. (ANSI)

## [GetICMProfileW](#)

The GetICMProfile function retrieves the file name of the current output color profile for a specified device context. (Unicode)

## [GetKValue](#)

The GetKValue macro retrieves the black color value from a CMYK color value.

## [GetLogColorSpaceA](#)

The GetLogColorSpace function retrieves the color space definition identified by a specified handle. (ANSI)

## [GetLogColorSpaceW](#)

The GetLogColorSpace function retrieves the color space definition identified by a specified handle. (Unicode)

## [GetMValue](#)

The GetMValue macro retrieves the magenta color value from a CMYK color value.

## [GetNamedProfileInfo](#)

Retrieves information about the International Color Consortium (ICC) named color profile that is specified in the first parameter.

## [GetNeutralAxis](#)

The IDeviceModelPlugIn::GetNeutralAxis return the XYZ colorimetry of sample points along the device's neutral axis.

## [GetNeutralAxisSize](#)

The IDeviceModelPlugIn::GetNeutralAxisSize function returns the number of data points along the neutral axis that are returned by the GetNeutralAxis function.

<p><a href="#">GetNumChannels</a></p> <p>Returns the number of device channels in the parameter pNumChannels.</p>
<p><a href="#">GetPrimarySamples</a></p> <p>Returns the measurement color for the primary sample.</p>
<p><a href="#">GetPS2ColorRenderingDictionary</a></p> <p>Retrieves the PostScript Level 2 color rendering dictionary from the specified ICC color profile.</p>
<p><a href="#">GetPS2ColorRenderingIntent</a></p> <p>Retrieves the PostScript Level 2 color <a href="#">rendering intent</a> from an ICC color profile.</p>
<p><a href="#">GetPS2ColorSpaceArray</a></p> <p>Retrieves the PostScript Level 2 <a href="#">color space</a> array from an ICC color profile.</p>
<p><a href="#">GetStandardColorSpaceProfileA</a></p> <p>Retrieves the color profile registered for the specified standard <a href="#">color space</a>. (ANSI)</p>
<p><a href="#">GetStandardColorSpaceProfileW</a></p> <p>Retrieves the color profile registered for the specified standard <a href="#">color space</a>. (Unicode)</p>
<p><a href="#">GetYValue</a></p> <p>The GetYValue macro retrieves the yellow color value from a CMYK color value.</p>
<p><a href="#">ICMENUMPROCA</a></p> <p>The EnumICMProfilesProcCallback callback is an application-defined callback function that processes color profile data from EnumICMProfiles . (ANSI)</p>
<p><a href="#">ICMENUMPROCW</a></p> <p>The EnumICMProfilesProcCallback callback is an application-defined callback function that processes color profile data from EnumICMProfiles . (Unicode)</p>
<p><a href="#">Initialize</a></p> <p>Takes a pointer to a Stream that contains the whole device model plug-in as input, and initializes any internal parameters required by the plug-in.</p>

<a href="#">Initialize</a>
Initializes a gamut map model profile (GMMP) by using the specified source and destination gamut boundary descriptions and optional source and destination device model plug-ins.
<a href="#">InstallColorProfileA</a>
Installs a given profile for use on a specified machine. The profile is also copied to the COLOR directory. (ANSI)
<a href="#">InstallColorProfileW</a>
Installs a given profile for use on a specified machine. The profile is also copied to the COLOR directory. (Unicode)
<a href="#">IsColorProfileTagPresent</a>
Reports whether a specified International Color Consortium (ICC) tag is present in the specified color profile.
<a href="#">IsColorProfileValid</a>
Allows you to determine whether the specified profile is a valid International Color Consortium (ICC) profile, or a valid Windows Color System (WCS) profile handle that can be used for color management.
<a href="#">OpenColorProfileA</a>
Creates a handle to a specified color profile. The handle can then be used in other profile management functions. (ANSI)
<a href="#">OpenColorProfileW</a>
Creates a handle to a specified color profile. The handle can then be used in other profile management functions. (Unicode)
<a href="#">PBM CALLBACKFN</a>
TBD (PBM CALLBACKFN)
<a href="#">PCMSC CALLBACKA</a>
*PCMSC CALLBACKA* (or <a href="#">ApplyCallbackFunction</a> ) is a callback function that you implement that updates the WCS configuration data while the dialog box displayed by the <a href="#">SetupColorMatchingW</a> function is executing.

## [PCMSCALLBACKW](#)

\**PCMSCALLBACKW*\* (or **ApplyCallbackFunction**) is a callback function that you implement that updates the WCS configuration data while the dialog box displayed by the **SetupColorMatchingW** function is executing.

## [RegisterCMMA](#)

Associates a specified identification value with the specified color management module dynamic link library (CMM DLL). When this ID appears in a color profile, Windows can then locate the corresponding CMM so as to create a transform. (ANSI)

## [RegisterCMMW](#)

Associates a specified identification value with the specified color management module dynamic link library (CMM DLL). When this ID appears in a color profile, Windows can then locate the corresponding CMM so as to create a transform. (Unicode)

## [SelectCMM](#)

Allows you to select the preferred color management module (CMM) to use.

## [SetColorProfileElement](#)

Sets the element data for a tagged profile element in an ICC color profile.

## [SetColorProfileElementReference](#)

Creates in a specified ICC color profile a new tag that references the same data as an existing tag.

## [SetColorProfileElementSize](#)

Sets the size of a tagged element in an ICC color profile.

## [SetColorProfileHeader](#)

Sets the header data in a specified ICC color profile.

## [SetColorSpace](#)

The SetColorSpace function defines the input color space for a given device context.

## [SetDeviceGammaRamp](#)

The SetDeviceGammaRamp function sets the gamma ramp on direct color display boards having drivers that support downloadable gamma ramps in hardware.

## [SetICMMMode](#)

The SetICMMMode function causes Image Color Management to be enabled, disabled, or queried on a given device context (DC).

## [SetICMProfileA](#)

The SetICMProfile function sets a specified color profile as the output profile for a specified device context (DC). (ANSI)

## [SetICMProfileW](#)

The SetICMProfile function sets a specified color profile as the output profile for a specified device context (DC). (Unicode)

## [SetStandardColorSpaceProfileA](#)

Registers a specified profile for a given standard [color space](#). The profile can be queried using [GetStandardColorSpaceProfileW](#). (ANSI)

## [SetStandardColorSpaceProfileW](#)

Registers a specified profile for a given standard [color space](#). The profile can be queried using [GetStandardColorSpaceProfileW](#). (Unicode)

## [SetTransformDeviceModellInfo](#)

Provides the plug-in with parameters to determine where in the transform sequence the specific plug-in occurs.

## [SetupColorMatchingA](#)

Creates a Color Management dialog box that lets the user choose whether to enable color management and, if so, provides control over the color profiles used and over the [rendering intent](#). (ANSI)

## [SetupColorMatchingW](#)

Creates a Color Management dialog box that lets the user choose whether to enable color management and, if so, provides control over the color profiles used and over the [rendering intent](#). (Unicode)

## [SourceToDestinationAppearanceColors](#)

Returns the appropriate gamut-mapped appearance colors in response to the specified number of colors and the CIEJCh colors.

<a href="#">TranslateBitmapBits</a>
Translates the colors of a bitmap having a defined format so as to produce another bitmap in a requested format.
<a href="#">TranslateColors</a>
Translates an array of colors from the source <a href="#">color space</a> to the destination color space as defined by a color transform.
<a href="#">UninstallColorProfileA</a>
Removes a specified color profile from a specified computer. Associated files are optionally deleted from the system. (ANSI)
<a href="#">UninstallColorProfileW</a>
Removes a specified color profile from a specified computer. Associated files are optionally deleted from the system. (Unicode)
<a href="#">UnregisterCMMA</a>
Dissociates a specified ID value from a given color management module dynamic-link library (CMM DLL). (ANSI)
<a href="#">UnregisterCMMW</a>
Dissociates a specified ID value from a given color management module dynamic-link library (CMM DLL). (Unicode)
<a href="#">UpdateICMRegKeyA</a>
The UpdateICMRegKey function manages color profiles and Color Management Modules in the system. (ANSI)
<a href="#">UpdateICMRegKeyW</a>
The UpdateICMRegKey function manages color profiles and Color Management Modules in the system. (Unicode)
<a href="#">WcsAssociateColorProfileWithDevice</a>
WcsAssociateColorProfileWithDevice associates a specified WCS color profile with a specified device.
<a href="#">WcsCheckColors</a>
Determines whether the colors in an array are within the output gamut of a specified WCS color transform.

## [WcsCreateIccProfile](#)

Converts a WCS profile into an International Color Consortium (ICC) profile.

## [WcsDisassociateColorProfileFromDevice](#)

Disassociates a specified WCS color profile from a specified device on a computer.

## [WcsEnumColorProfiles](#)

Enumerates all color profiles that satisfy the enumeration criteria in the specified profile management scope.

## [WcsEnumColorProfilesSize](#)

Returns the size, in bytes, of the buffer that is required by the [WcsEnumColorProfiles](#) function to enumerate color profiles.

## [WcsGetCalibrationManagementState](#)

Determines whether system management of the display calibration state is enabled.

## [WcsGetDefaultColorProfile](#)

Retrieves the default color profile for a device, or for a device-independent default if the device is not specified.

## [WcsGetDefaultColorProfileSize](#)

Returns the size, in bytes, of the default color profile name (including the **NULL** terminator), for a device.

## [WcsGetDefaultRenderingIntent](#)

Retrieves the default rendering intent in the specified profile management scope.

## [WcsGetUsePerUserProfiles](#)

Determines whether the user chose to use a per-user profile association list for the specified device.

## [WcsOpenColorProfileA](#)

Creates a handle to a specified color profile. (ANSI)

## [WcsOpenColorProfileW](#)

Creates a handle to a specified color profile. (Unicode)

## [WcsSetCalibrationManagementState](#)

Enables or disables system management of the display calibration state.

## [WcsSetDefaultColorProfile](#)

Sets the default color profile name for the specified profile type in the specified profile management scope.

## [WcsSetDefaultRenderingIntent](#)

Sets the default rendering intent in the specified profile management scope.

## [WcsSetUsePerUserProfiles](#)

Enables a user to specify whether or not to use a per-user profile association list for the specified device.

## [WcsTranslateColors](#)

Translates an array of colors from the source color space to the destination color space as defined by a color transform.

# Interfaces

## [IDeviceModelPlugIn](#)

Describes the methods that are defined for the IDeviceModelPlugIn Component Object Model (COM) interface.

## [IGamutMapModelPlugIn](#)

Describes the methods that are defined for the IGamutMapModelPlugIn Component Object Model (COM) interface.

# Structures

## [BlackInformation](#)

Contains information for device models that have a black color channel.

## [CIEXYZ](#)

The CIEXYZ structure contains the x,y, and z coordinates of a specific color in a specified color space.

## [CIEXYZTRIPLE](#)

The CIEXYZTRIPLE structure contains the x,y, and z coordinates of the three colors that correspond to the red, green, and blue endpoints for a specified logical color space.

## [CMYKCOLOR](#)

Description of the CMYKCOLOR structure.

## [COLOR](#)

Description of the COLOR union.

## [COLORMATCHSETUPA](#)

The COLORMATCHSETUP structure contains information that the [SetupColorMatchingW](#) function uses to initialize the **ColorManagement** dialog box. (ANSI)

## [COLORMATCHSETUPW](#)

The COLORMATCHSETUP structure contains information that the [SetupColorMatchingW](#) function uses to initialize the **ColorManagement** dialog box. (Unicode)

## [ENUMTYPEA](#)

Contains information that defines the profile enumeration constraints. (ANSI)

## [ENUMTYPEW](#)

Contains information that defines the profile enumeration constraints. (Unicode)

## [GamutBoundaryDescription](#)

Defines a gamut boundary.

## [GamutShell](#)

Contains information that defines a gamut shell, which is represented by a list of indexed triangles. The vertex buffer contains the vertices data.

## [GamutShellTriangle](#)

Contains three vertex indices for accessing a vertex buffer.

## [GENERIC3CHANNEL](#)

TBD (GENERIC3CHANNEL)

## [GRAYCOLOR](#)

Description of the GRAYCOLOR structure.

## [HiFiCOLOR](#)

Description of the HiFiCOLOR structure.

## [JabColorF](#)

JabColorF (wcsplugin.h) is a structure.

## [JChColorF](#)

JChColorF (wcsplugin.h) is a structure.

## [LabCOLOR](#)

TBD (LabCOLOR)

## [LOGCOLORSPACEA](#)

The LOGCOLORSPACE structure contains information that defines a logical color space. (ANSI)

## [LOGCOLORSPACEW](#)

The LOGCOLORSPACE structure contains information that defines a logical color space. (Unicode)

## [NAMED\\_PROFILE\\_INFO](#)

The NAMED\_PROFILE\_INFO structure is used to store information about a named color profile.

## [NAMEDCOLOR](#)

TBD (NAMEDCOLOR)

## [PrimaryJabColors](#)

This structure contains eight primary colors in Jab coordinates.

## [PrimaryXYZColors](#)

This structure contains eight primary colors in XYZ coordinates.

## [PROFILE](#)

Contains information that defines a color profile.

## [PROFILEHEADER](#)

Contains information that describes the contents of a device profile file. This header occurs at the beginning of a device profile file.

## [RGBCOLOR](#)

TBD (RGBCOLOR)

## [XYZCOLOR](#)

TBD (XYZCOLOR)

## [XYZColorF](#)

XYZColorF (wcsplugin.h) is a structure.

## [YxyCOLOR](#)

TBD (YxyCOLOR)

# Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# icm.h header

Article09/22/2022

This header is used by multiple technologies. For more information, see:

- [Print DDI reference](#)
- [Windows Color System](#)

icm.h contains the following programming interfaces:

## Functions

<a href="#">AssociateColorProfileWithDeviceA</a>
Associates a specified color profile with a specified device. (ANSI)
<a href="#">AssociateColorProfileWithDeviceW</a>
Associates a specified color profile with a specified device. (Unicode)
<a href="#">CheckBitmapBits</a>
Checks whether the pixels in a specified bitmap lie within the output <a href="#">gamut</a> of a specified transform.
<a href="#">CheckColors</a>
Determines whether the colors in an array lie within the output <a href="#">gamut</a> of a specified transform.
<a href="#">CloseColorProfile</a>
Closes an open profile handle.
<a href="#">CMCheckColors</a>
Determines whether given colors lie within the output <a href="#">gamut</a> of a specified transform.
<a href="#">CMCheckColorsInGamut</a>
Determines whether specified RGB triples lie in the output <a href="#">gamut</a> of a specified transform.
<a href="#">CMCheckRGBs</a>
Checks bitmap colors against an output gamut.

<a href="#">CMConvertColorNameToIndex</a>	Converts color names in a named color space to index numbers in a color profile.
<a href="#">CMConvertIndexToColorName</a>	Transforms indices in a color space to an array of names in a named color space. (CMConvertIndexToColorName)
<a href="#">CMCreateDeviceLinkProfile</a>	Creates a <a href="#">device link profile</a> in the format specified by the International Color Consortium in its ICC Profile Format Specification.
<a href="#">CMCreateMultiProfileTransform</a>	Accepts an array of profiles or a single <a href="#">device link profile</a> and creates a color transform. This transform is a mapping from the color space specified by the first profile to that of the second profile and so on to the last one.
<a href="#">CMCreateProfile</a>	Creates a display color profile from a <a href="#">LOGCOLORSPACEA</a> structure.
<a href="#">CMCreateProfileW</a>	Creates a display color profile from a <a href="#">LOGCOLORSPACEW</a> structure.
<a href="#">CMCreateTransform</a>	Deprecated. There is no replacement API because this one was no longer being used. Developers of alternate CMM modules are not required to implement it. (CMCreateTransform)
<a href="#">CMCreateTransformExt</a>	Creates a color transform that maps from an input <a href="#">LOGCOLORSPACEA</a> to an optional target space and then to an output device, using a set of flags that define how the transform should be created.
<a href="#">CMCreateTransformExtW</a>	Creates a color transform that maps from an input <a href="#">LOGCOLORSPACEW</a> to an optional target space and then to an output device, using a set of flags that define how the transform should be created.
<a href="#">CMCreateTransformW</a>	Deprecated. There is no replacement API because this one was no longer being used. Developers of alternate CMM modules are not required to implement it. (CMCreateTransformW)

## [CMDeleteTransform](#)

Deletes a specified color transform, and frees any memory associated with it.

## [CMGetInfo](#)

Retrieves various information about the color management module (CMM).

## [CMGetNamedProfileInfo](#)

Retrieves information about the specified named color profile.

## [CMGetPS2ColorRenderingDictionary](#)

CMGetPS2ColorRenderingDictionary and the additional parameters associated with it are to be determined.

## [CMGetPS2ColorRenderingIntent](#)

Retrieves the PostScript Level 2 color [rendering intent](#) from a profile.

## [CMGetPS2ColorSpaceArray](#)

CMGetPS2ColorSpaceArray and the parameters, returns, and remarks associated with it are to be determined.

## [CMIsProfileValid](#)

Reports whether the given profile is a valid ICC profile that can be used for color management.

## [CMTranslateColors](#)

Translates an array of colors from a source [color space](#) to a destination color space using a color transform.

## [CMTranslateRGB](#)

Translates an application-supplied RGBQuad into the device [color space](#).

## [CMTranslateRBGs](#)

Translates a bitmap from one [color space](#) to another using a color transform.

## [CMTranslateRBGsExt](#)

Translates a bitmap from one defined format into a different defined format and calls a callback function periodically, if one is specified, to report progress and permit the calling application to terminate the translation.

## [ColorProfileAddDisplayAssociation](#)

ColorProfileAddDisplayAssociation associates an installed color profile with a specified display in the given scope.

## [ColorProfileGetDisplayDefault](#)

ColorProfileGetDisplayDefault gets the default color profile for a given display in the specified scope.

## [ColorProfileGetDisplayList](#)

ColorProfileGetDisplayList retrieves the list of profiles associated with a given display in the specified scope.

## [ColorProfileGetDisplayUserScope](#)

ColorProfileGetDisplayUserScope gets the currently selected color profile scope of the provided display - either user or system.

## [ColorProfileRemoveDisplayAssociation](#)

ColorProfileRemoveDisplayAssociation disassociates an installed color profile from a specified display in the given scope.

## [ColorProfileSetDisplayDefaultAssociation](#)

ColorProfileSetDisplayDefaultAssociation sets an installed color profile as the default profile for a specified display in the given scope.

## [ConvertColorNameToIntIndex](#)

Converts color names in a named color space to index numbers in an International Color Consortium (ICC) color profile.

## [ConvertIndexToColorName](#)

Transforms indices in a color space to an array of names in a named color space.  
(ConvertIndexToColorName)

## [CreateColorTransformA](#)

Creates a color transform that applications can use to perform color management. (ANSI)

## [CreateColorTransformW](#)

Creates a color transform that applications can use to perform color management. (Unicode)

<a href="#">CreateDeviceLinkProfile</a>
Creates an International Color Consortium (ICC) <i>device link profile</i> from a set of color profiles, using the specified intents.
<a href="#">CreateMultiProfileTransform</a>
Accepts an array of profiles or a single <a href="#">device link profile</a> and creates a color transform that applications can use to perform color mapping.
<a href="#">CreateProfileFromLogColorSpaceA</a>
Converts a logical <a href="#">color space</a> to a <a href="#">device profile</a> . (ANSI)
<a href="#">CreateProfileFromLogColorSpaceW</a>
Converts a logical <a href="#">color space</a> to a <a href="#">device profile</a> . (Unicode)
<a href="#">DeleteColorTransform</a>
Deletes a given color transform.
<a href="#">DisassociateColorProfileFromDeviceA</a>
Disassociates a specified color profile with a specified device on a specified computer. (ANSI)
<a href="#">DisassociateColorProfileFromDeviceW</a>
Disassociates a specified color profile with a specified device on a specified computer. (Unicode)
<a href="#">EnumColorProfilesA</a>
Enumerates all the profiles satisfying the given enumeration criteria. (ANSI)
<a href="#">EnumColorProfilesW</a>
Enumerates all the profiles satisfying the given enumeration criteria. (Unicode)
<a href="#">GetCMMInfo</a>
Retrieves various information about the color management module (CMM) that created the specified color transform.
<a href="#">GetColorDirectoryA</a>
Retrieves the path of the Windows COLOR directory on a specified machine. (ANSI)

## [GetColorDirectoryW](#)

Retrieves the path of the Windows COLOR directory on a specified machine. (Unicode)

## [GetColorProfileElement](#)

Copies data from a specified tagged profile element of a specified color profile into a buffer.

## [GetColorProfileElementTag](#)

Retrieves the tag name specified by *dwIndex* in the tag table of a given International Color Consortium (ICC) color profile, where *dwIndex* is a one-based index into that table.

## [GetColorProfileFromHandle](#)

Given a handle to an open color profile, the **GetColorProfileFromHandle** function copies the contents of the profile into a buffer supplied by the application. If the handle is a Windows Color System (WCS) handle, then the DMP is returned and the CAMP and GMMP associated with the HPROFILE are ignored.

## [GetColorProfileHeader](#)

Retrieves or derives ICC header structure from either ICC color profile or WCS XML profile. Drivers and applications should assume returning TRUE only indicates that a properly structured header is returned. Each tag will still need to be validated independently using either legacy ICM2 APIs or XML schema APIs.

## [GetCountColorProfileElements](#)

Retrieves the number of tagged elements in a given color profile.

## [GetNamedProfileInfo](#)

Retrieves information about the International Color Consortium (ICC) named color profile that is specified in the first parameter.

## [GetPS2ColorRenderingDictionary](#)

Retrieves the PostScript Level 2 color rendering dictionary from the specified ICC color profile.

## [GetPS2ColorRenderingIntent](#)

Retrieves the PostScript Level 2 color [rendering intent](#) from an ICC color profile.

## [GetPS2ColorSpaceArray](#)

Retrieves the PostScript Level 2 [color space](#) array from an ICC color profile.

## [GetStandardColorSpaceProfileA](#)

Retrieves the color profile registered for the specified standard [color space](#). (ANSI)

## [GetStandardColorSpaceProfileW](#)

Retrieves the color profile registered for the specified standard [color space](#). (Unicode)

## [InstallColorProfileA](#)

Installs a given profile for use on a specified machine. The profile is also copied to the COLOR directory. (ANSI)

## [InstallColorProfileW](#)

Installs a given profile for use on a specified machine. The profile is also copied to the COLOR directory. (Unicode)

## [IsColorProfileTagPresent](#)

Reports whether a specified International Color Consortium (ICC) tag is present in the specified color profile.

## [IsColorProfileValid](#)

Allows you to determine whether the specified profile is a valid International Color Consortium (ICC) profile, or a valid Windows Color System (WCS) profile handle that can be used for color management.

## [OpenColorProfileA](#)

Creates a handle to a specified color profile. The handle can then be used in other profile management functions. (ANSI)

## [OpenColorProfileW](#)

Creates a handle to a specified color profile. The handle can then be used in other profile management functions. (Unicode)

## [RegisterCMMA](#)

Associates a specified identification value with the specified color management module dynamic link library (CMM DLL). When this ID appears in a color profile, Windows can then locate the corresponding CMM so as to create a transform. (ANSI)

## [RegisterCMMW](#)

Associates a specified identification value with the specified color management module dynamic link library (CMM DLL). When this ID appears in a color profile, Windows can then locate the corresponding CMM so as to create a transform. (Unicode)

## [SelectCMM](#)

Allows you to select the preferred color management module (CMM) to use.

## [SetColorProfileElement](#)

Sets the element data for a tagged profile element in an ICC color profile.

## [SetColorProfileElementReference](#)

Creates in a specified ICC color profile a new tag that references the same data as an existing tag.

## [SetColorProfileElementSize](#)

Sets the size of a tagged element in an ICC color profile.

## [SetColorProfileHeader](#)

Sets the header data in a specified ICC color profile.

## [SetStandardColorSpaceProfileA](#)

Registers a specified profile for a given standard [color space](#). The profile can be queried using [GetStandardColorSpaceProfileW](#). (ANSI)

## [SetStandardColorSpaceProfileW](#)

Registers a specified profile for a given standard [color space](#). The profile can be queried using [GetStandardColorSpaceProfileW](#). (Unicode)

## [SetupColorMatchingA](#)

Creates a Color Management dialog box that lets the user choose whether to enable color management and, if so, provides control over the color profiles used and over the [rendering intent](#). (ANSI)

## [SetupColorMatchingW](#)

Creates a Color Management dialog box that lets the user choose whether to enable color management and, if so, provides control over the color profiles used and over the [rendering intent](#). (Unicode)

<a href="#">TranslateBitmapBits</a>
Translates the colors of a bitmap having a defined format so as to produce another bitmap in a requested format.
<a href="#">TranslateColors</a>
Translates an array of colors from the source <a href="#">color space</a> to the destination color space as defined by a color transform.
<a href="#">UninstallColorProfileA</a>
Removes a specified color profile from a specified computer. Associated files are optionally deleted from the system. (ANSI)
<a href="#">UninstallColorProfileW</a>
Removes a specified color profile from a specified computer. Associated files are optionally deleted from the system. (Unicode)
<a href="#">UnregisterCMMA</a>
Dissociates a specified ID value from a given color management module dynamic-link library (CMM DLL). (ANSI)
<a href="#">UnregisterCMMW</a>
Dissociates a specified ID value from a given color management module dynamic-link library (CMM DLL). (Unicode)
<a href="#">WcsAssociateColorProfileWithDevice</a>
WcsAssociateColorProfileWithDevice associates a specified WCS color profile with a specified device.
<a href="#">WcsCheckColors</a>
Determines whether the colors in an array are within the output gamut of a specified WCS color transform.
<a href="#">WcsCreateIccProfile</a>
Converts a WCS profile into an International Color Consortium (ICC) profile.
<a href="#">WcsDisassociateColorProfileFromDevice</a>
Disassociates a specified WCS color profile from a specified device on a computer.

## [WcsEnumColorProfiles](#)

Enumerates all color profiles that satisfy the enumeration criteria in the specified profile management scope.

## [WcsEnumColorProfilesSize](#)

Returns the size, in bytes, of the buffer that is required by the [WcsEnumColorProfiles](#) function to enumerate color profiles.

## [WcsGetCalibrationManagementState](#)

Determines whether system management of the display calibration state is enabled.

## [WcsGetDefaultColorProfile](#)

Retrieves the default color profile for a device, or for a device-independent default if the device is not specified.

## [WcsGetDefaultColorProfileSize](#)

Returns the size, in bytes, of the default color profile name (including the **NULL** terminator), for a device.

## [WcsGetDefaultRenderingIntent](#)

Retrieves the default rendering intent in the specified profile management scope.

## [WcsGetUsePerUserProfiles](#)

Determines whether the user chose to use a per-user profile association list for the specified device.

## [WcsOpenColorProfileA](#)

Creates a handle to a specified color profile. (ANSI)

## [WcsOpenColorProfileW](#)

Creates a handle to a specified color profile. (Unicode)

## [WcsSetCalibrationManagementState](#)

Enables or disables system management of the display calibration state.

## [WcsSetDefaultColorProfile](#)

Sets the default color profile name for the specified profile type in the specified profile management scope.

## [WcsSetDefaultRenderingIntent](#)

Sets the default rendering intent in the specified profile management scope.

## [WcsSetUsePerUserProfiles](#)

Enables a user to specify whether or not to use a per-user profile association list for the specified device.

## [WcsTranslateColors](#)

Translates an array of colors from the source color space to the destination color space as defined by a color transform.

# Callback functions

## [PBMCALLBACKFN](#)

TBD (PBMCALLBACKFN)

## [PCMSCALLBACKA](#)

\*PCMSCALLBACKA\* (or [ApplyCallbackFunction](#)) is a callback function that you implement that updates the WCS configuration data while the dialog box displayed by the [SetupColorMatchingW](#) function is executing.

## [PCMSCALLBACKW](#)

\*PCMSCALLBACKW\* (or [ApplyCallbackFunction](#)) is a callback function that you implement that updates the WCS configuration data while the dialog box displayed by the [SetupColorMatchingW](#) function is executing.

# Structures

## [CMYKCOLOR](#)

Description of the CMYKCOLOR structure.

## [COLOR](#)

Description of the COLOR union.

## [COLORMATCHSETUPA](#)

The **COLORMATCHSETUP** structure contains information that the [SetupColorMatchingW](#) function uses to initialize the **ColorManagement** dialog box. (ANSI)

## [COLORMATCHSETUPW](#)

The **COLORMATCHSETUP** structure contains information that the [SetupColorMatchingW](#) function uses to initialize the **ColorManagement** dialog box. (Unicode)

## [ENUMTYPEA](#)

Contains information that defines the profile enumeration constraints. (ANSI)

## [ENUMTYPEW](#)

Contains information that defines the profile enumeration constraints. (Unicode)

## [GENERIC3CHANNEL](#)

TBD (GENERIC3CHANNEL)

## [GRAYCOLOR](#)

Description of the GRAYCOLOR structure.

## [HiFiCOLOR](#)

Description of the HiFiCOLOR structure.

## [LabCOLOR](#)

TBD (LabCOLOR)

## [NAMED\\_PROFILE\\_INFO](#)

The **NAMED\_PROFILE\_INFO** structure is used to store information about a named color profile.

## [NAMEDCOLOR](#)

TBD (NAMEDCOLOR)

## [PROFILE](#)

Contains information that defines a color profile.

## [PROFILEHEADER](#)

Contains information that describes the contents of a device profile file. This header occurs at the beginning of a device profile file.

## [RGBCOLOR](#)

TBD (RGBCOLOR)

## [XYZCOLOR](#)

TBD (XYZCOLOR)

## [YxyCOLOR](#)

TBD (YxyCOLOR)

# Enumerations

## [BMFORMAT](#)

The values of the **BMFORMAT** enumerated type are used by several WCS functions to indicate the format that particular bitmaps are in.

## [COLORDATATYPE](#)

Used by WCS functions to indicate the data type of vector content.

## [COLORPROFILESUBTYPE](#)

Specifies the subtype of the color profile.

## [COLORPROFILETYPE](#)

Specifies the type of color profile.

## [COLORTYPE](#)

The values of the **COLORTYPE** enumeration are used by several WCS functions. Variables of type **COLOR** are defined in the color spaces enumerated by the **COLORTYPE** enumeration.

## [WCS\\_PROFILE\\_MANAGEMENT\\_SCOPE](#)

Specifies the scope of a profile management operation, such as associating a profile with a device.

---

# Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# AssociateColorProfileWithDeviceA function (icm.h)

Associates a specified color profile with a specified device.

## ! Note

This API does not support "advanced color" profiles for HDR monitors. Use [ColorProfileAddDisplayAssociation](#) for managing advanced color profiles.

## Syntax

C++

```
BOOL AssociateColorProfileWithDeviceA(  
    PCSTR pMachineName,  
    PCSTR pProfileName,  
    PCSTR pDeviceName  
) ;
```

## Parameters

pMachineName

Reserved. Must be **NULL**. This parameter is intended to point to the name of the machine on which to associate the specified profile and device. A **NULL** pointer indicates the local machine.

pProfileName

Points to the file name of the profile to associate.

pDeviceName

Points to the name of the device to associate.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Remarks

The **AssociateColorProfileWithDevice** function will fail if the profile has not been installed on the computer using the [InstallColorProfileW](#) function.

Note that under Windows (Windows 95 or later), the PostScript device driver for printers assumes a CMYK color model. Therefore, all PostScript printers must use a CMYK color profile. Windows 2000 does not have this limitation.

If the specified device is a monitor, this function updates the default profile.

Several profiles are typically associated with printers, based on paper and ink types. There is no default. The GDI selects the best one from the associated profiles when your application creates a device context (DC).

Scanners also have no default profile. However, it is atypical to associate more than one profile with a scanner.

**AssociateColorProfileWithDevice** always adds the specified profile to the current user's per-user profile association list for the specified device. Before adding the profile to the list, **AssociateColorProfileWithDevice** determines whether the user has previously expressed the desire to use a per-user profile association list for the device. If so, then **AssociateColorProfileWithDevice** simply adds the specified profile to the existing per-user profile association list for the device. If not, then **AssociateColorProfileWithDevice** creates a new per-user profile association list for the device by copying the system-wide association list for that device. It then appends the specified profile to the per-user list. From that point on, the current user will be using a per-user profile association list for the specified device, as if [WcsSetUsePerUserProfiles](#) had been called for *pDevice* with the *usePerUserProfiles* parameter set to TRUE.

## Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 10 Build 20348
Minimum supported server	Windows 10 Build 20348
Header	icm.h
Library	Mscms.Lib

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [DisassociateColorProfileFromDevice](#)

---

Last updated on 10/31/2025

# AssociateColorProfileWithDeviceW function (icm.h)

Article07/27/2022

Associates a specified color profile with a specified device.

## ⓘ Note

This API does not support "advanced color" profiles for HDR monitors. Use [ColorProfileAddDisplayAssociation](#) for managing advanced color profiles.

## Syntax

C++

```
BOOL AssociateColorProfileWithDeviceW(
    PCWSTR pMachineName,
    PCWSTR pProfileName,
    PCWSTR pDeviceName
);
```

## Parameters

pMachineName

Reserved. Must be **NULL**. This parameter is intended to point to the name of the machine on which to associate the specified profile and device. A **NULL** pointer indicates the local machine.

pProfileName

Points to the file name of the profile to associate.

pDeviceName

Points to the name of the device to associate.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Remarks

The **AssociateColorProfileWithDevice** function will fail if the profile has not been installed on the computer using the [InstallColorProfileW](#) function.

Note that under Windows (Windows 95 or later), the PostScript device driver for printers assumes a CMYK color model. Therefore, all PostScript printers must use a CMYK color profile. Windows 2000 does not have this limitation.

If the specified device is a monitor, this function updates the default profile.

Several profiles are typically associated with printers, based on paper and ink types. There is no default. The GDI selects the best one from the associated profiles when your application creates a device context (DC).

Scanners also have no default profile. However, it is atypical to associate more than one profile with a scanner.

**AssociateColorProfileWithDevice** always adds the specified profile to the current user's per-user profile association list for the specified device. Before adding the profile to the list, **AssociateColorProfileWithDevice** determines whether the user has previously expressed the desire to use a per-user profile association list for the device. If so, then **AssociateColorProfileWithDevice** simply adds the specified profile to the existing per-user profile association list for the device. If not, then **AssociateColorProfileWithDevice** creates a new per-user profile association list for the device by copying the system-wide association list for that device. It then appends the specified profile to the per-user list. From that point on, the current user will be using a per-user profile association list for the specified device, as if [WcsSetUsePerUserProfiles](#) had been called for *pDevice* with the *usePerUserProfiles* parameter set to **TRUE**.

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h

Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [DisassociateColorProfileFromDeviceW](#)

---

## Feedback

Was this page helpful?



[Get help at Microsoft Q&A](#)

# BMFORMAT enumeration (icm.h)

Article03/13/2023

The values of the **BMFORMAT** enumerated type are used by several WCS functions to indicate the format that particular bitmaps are in.

## Syntax

C++

```
typedef enum {
    BM_x555RGB = 0x0000,
    BM_x555XYZ = 0x0101,
    BM_x555Yxy,
    BM_x555Lab,
    BM_x555G3CH,
    BM_RGBTRIPLETS = 0x0002,
    BM_BGRTRIPLETS = 0x0004,
    BM_XYZTRIPLETS = 0x0201,
    BM_YxyTRIPLETS,
    BM_LabTRIPLETS,
    BM_G3CHTRIPLETS,
    BM_5CHANNEL,
    BM_6CHANNEL,
    BM_7CHANNEL,
    BM_8CHANNEL,
    BM_GRAY,
    BM_XRGBQUADS = 0x0008,
    BM_XBGRQUADS = 0x0010,
    BM_XG3CHQUADS = 0x0304,
    BM_KYMCQUADS,
    BM_CMYKQUADS = 0x0020,
    BM_10b_RGB = 0x0009,
    BM_10b_XYZ = 0x0401,
    BM_10b_Yxy,
    BM_10b_Lab,
    BM_10b_G3CH,
    BM_NAMED_INDEX,
    BM_16b_RGB = 0x000A,
    BM_16b_XYZ = 0x0501,
    BM_16b_Yxy,
    BM_16b_Lab,
    BM_16b_G3CH,
    BM_16b_GRAY,
    BM_565RGB = 0x0001,
    BM_32b_scRGB = 0x0601,
    BM_32b_scARGB = 0x0602,
    BM_S2DOT13FIXED_scRGB = 0x0603,
    BM_S2DOT13FIXED_scARGB = 0x0604,
    BM_R10G10B10A2 = 0x0701,
    BM_R10G10B10A2_XR = 0x0702,
    BM_R16G16B16A16_FLOAT = 0x0703
} BMFORMAT;
```

## Constants

**BM\_x555RGB**

Value: 0x0000

16 bits per pixel. RGB color space. 5 bits per channel. The most significant bit is ignored.

**BM\_x555XYZ**

Value: 0x0101

16 bits per pixel. CIE device-independent XYZ color space. 5 bits per channel. The most significant bit is ignored.

**BM\_x555Yxy**

16 bits per pixel. Yxy color space. 5 bits per channel. The most significant bit is ignored.

**BM\_x555Lab**

16 bits per pixel. L\*a\*b color space. 5 bits per channel. The most significant bit is ignored.

**BM\_x555G3CH**

16 bits per pixel. G3CH color space. 5 bits per channel. The most significant bit is ignored.

**BM\_RGBTRIPLETS**

Value: 0x0002

24 bits per pixel maximum. For three channel colors, such as Red,Green,Blue, the total size is 24 bits per pixel. For single channel colors, such as gray, the total size is 8 bits per pixel.

**BM\_BGRTRIPLETS**

Value: 0x0004

24 bits per pixel maximum. For three channel colors, such as Red,Green,Blue, the total size is 24 bits per pixel. For single channel colors, such as gray, the total size is 8 bits per pixel.

**BM\_XYZTRIPLETS**

Value: 0x0201

24 bits per pixel maximum. For three channel, X, Y and Z values, the total size is 24 bits per pixel. For single channel gray scale, the total size is 8 bits per pixel.

**① Note**

The TranslateBitmapBits function does not support BM\_XYZTRIPLETS as an input.

**BM\_YxyTRIPLETS**

24 bits per pixel maximum. For three channel, Y, x and y values, the total size is 24 bits per pixel. For single channel gray scale, the total size is 8 bits per pixel.

**① Note**

The TranslateBitmapBits function does not support BM\_YxyTRIPLETS as an input.

**BM\_LabTRIPLETS**

24 bits per pixel maximum. For three channel, L, a and b values, the total size is 24 bits per pixel. For single channel gray scale, the total size is 8 bits per pixel.

**BM\_G3CHTRIPLETS**

24 bits per pixel maximum. For three channel values, the total size is 24 bits per pixel. For single channel gray scale, the total size is 8 bits per pixel.

**BM\_5CHANNEL**

40 bits per pixel. 8 bits apiece are used for each channel.

**BM\_6CHANNEL**

48 bits per pixel. 8 bits apiece are used for each channel.

**BM\_7CHANNEL**

56 bits per pixel. 8 bits apiece are used for each channel.

**BM\_8CHANNEL**

64 bits per pixel. 8 bits apiece are used for each channel.

**BM\_GRAY**

32 bits per pixel. Only the 8 bit gray-scale value is used.

**BM\_xRGBQUADS**

Value: 0x0008

32 bits per pixel. 8 bits are used for each color channel. The most significant byte is ignored.

**BM\_xBGRQUADS**

Value: 0x0010

32 bits per pixel. 8 bits are used for each color channel. The most significant byte is ignored.

**BM\_xG3CHQUADS**

Value: 0x0304

32 bits per pixel. 8 bits are used for each color channel. The most significant byte is ignored.

**BM\_KYMCQUADS**

32 bits per pixel. 8 bits are used for each color channel.

**BM\_CMYKQUADS**

Value: 0x0020

32 bits per pixel. 8 bits are used for each color channel.

<p><b>BM_10b_RGB</b> Value: 0x0009 32 bits per pixel. 10 bits are used for each color channel. The 2 most significant bits are ignored.</p>
<p><b>BM_10b_XYZ</b> Value: 0x0401 32 bits per pixel. 10 bits are used for each color channel. The 2 most significant bits are ignored.</p>
<p><b>BM_10b_Yxy</b> 32 bits per pixel. 10 bits are used for each color channel. The 2 most significant bits are ignored.</p>
<p><b>BM_10b_Lab</b> 32 bits per pixel. 10 bits are used for each color channel. The 2 most significant bits are ignored.</p>
<p><b>BM_10b_G3CH</b> 32 bits per pixel. 10 bits are used for each color channel. The 2 most significant bits are ignored.</p>
<p><b>BM_NAMED_INDEX</b> 32 bits per pixel. Named color indices. Index numbering begins at 1.</p>
<p><b>BM_16b_RGB</b> Value: 0x000A 48 bits per pixel. Each channel uses 16 bits.</p>
<p><b>BM_16b_XYZ</b> Value: 0x0501 48 bits per pixel. Each channel uses 16 bits.</p>
<p><b>BM_16b_Yxy</b> 48 bits per pixel. Each channel uses 16 bits.</p>
<p><b>BM_16b_Lab</b> 48 bits per pixel. Each channel uses 16 bits.</p>
<p><b>BM_16b_G3CH</b> 48 bits per pixel. Each channel uses 16 bits.</p>
<p><b>BM_16b_GRAY</b> 16 bits per pixel.</p>
<p><b>BM_565RGB</b> Value: 0x0001 16 bits per pixel. 5 bits are used for red, 6 for green, and 5 for blue.</p>
<p><b>BM_32b_scRGB</b> Value: 0x0601 96 bits per pixel, 32 bit per channel IEEE floating point.</p>
<p><b>BM_32b_scARGB</b> Value: 0x0602 128 bits per pixel, 32 bit per channel IEEE floating point.</p>
<p><b>BM_S2DOT13FIXED_scRGB</b> Value: 0x0603 48 bits per pixel, Fixed point integer ranging from -4 to +4 with a sign bit and 2 bit exponent and 13 bit mantissa.</p>
<p><b>BM_S2DOT13FIXED_scARGB</b> Value: 0x0604 64 bits per pixel, Fixed point integer ranging from -4 to +4 with a sign bit and 2 bit exponent and 13 bit mantissa.</p>
<p><b>BM_R10G10B10A2</b> Value: 0x0701 32 bits per pixel. 10 bits are used for each color channel. The two most significant bits are alpha.</p>
<p><b>BM_R10G10B10A2_XR</b> Value: 0x0702 32 bits per pixel. 10 bits are used for each color channel. The 10 bits of each color channel are 2.8 fixed point with a -0.75 bias, giving a range of [-0.76 .. 1.25]. This range corresponds to [-0.5 .. 1.5] in a gamma = 1 space. The two most significant bits are preserved for alpha.</p>
<p>This uses an extended range (XR) sRGB color space. It has the same RGB primaries, white point, and gamma as sRGB.</p>
<p><b>BM_R16G16B16A16_FLOAT</b> Value: 0x0703 64 bits per pixel. Each channel is a 16-bit float. The last WORD is alpha.</p>

## Remarks

### Table of bitmap formats

The follow table shows, for each of the formats, the number of bits per pixel, the number of channels, the order of the channels, and the bit-by-bit structure of each byte. You might have to scroll to the right to see all the columns of the table.

Format	Bits Per Pixel	Number of Channels	Channel Ordering	Byte 0	Byte 1	Byte 2
BM_GRAY	8	1		K7K6K5K4K3K2K1K0		
BM_565RGB	16	3	BGR	G2G1G0B4B3B2B1B0	R4R3R2R1R0G5G4G3	
BM_x555RGB	16	3	BGR	G2G1G0B4B3B2B1B0	xR4R3R2R1R0G4G3	
BM_x555XYZ	16	3	ZYX	Y2Y1Y0Z4Z3Z2Z1Z0	xX4X3X2X1X0Y4Y3	
BM_x555Yxy	16	3	yxY	x2x1x0y4y3y2y1y0	xY4Y3Y2Y1Y0x4x3	
BM_x555Lab	16	3	baL	a2a1a0b4b3b2b1b0	xL4L3L2L1L0a4a3	
BM_x555G3CH	16	3	123	xC14C13C12C11C10C24C23	C22C21C20C34C33C32C31C30	
BM_16b_GRAY	16	1	K	K7K6K5K4K3K2K1K0	K15K14K13K12K11K10K9K8	
BM_RGBTRIPLETS	24	3	BGR	B7B6B5B4B3B2B1B0	G7G6G5G4G3G2G1G0	R7R6R5R4R3R2R1R0
BM_BGRTRIPLETS	24	3	RGB	R7R6R5R4R3R2R1R0	G7G6G5G4G3G2G1G0	B7B6B5B4B3B2B1B0
BM_XYZTRIPLETS	24	3	XYZ	X7X6X5X4X3X2X1X0	Y7Y6Y5Y4Y3Y2Y1Y0	Z7Z6Z5Z4Z3Z2Z1Z0
BM_YxyTRIPLETS	24	3	Yxy	Y7Y6Y5Y4Y3Y2Y1Y0	x7x6x5x4x3x2x1x0	y7y6y5y4y3y2y1y0
BM_LabTRIPLETS	24	3	Lab	L7L6L5L4L3L2L1L0	a7a6a5a4a3a2a1a0	b7b6b5b4b3b2b1b0
BM_G3CHTRIPLETS	24	3	123	C17C16C15C14C13C12C11C10	C27C26C25C24C23C22C21C20	C37C36C35C34C33C32C31C
BM_xRGBQUADS	32	3	BGRx	B7B6B5B4B3B2B1B0	G7G6G5G4G3G2G1G0	R7R6R5R4R3R2R1R0
BM_xBGRQUADS	32	3	RGBx	R7R6R5R4R3R2R1R0	G7G6G5G4G3G2G1G0	B7B6B5B4B3B2B1B0
BM_xG3CHQUADS	32	3	123x	C17C16C15C14C13C12C11C10	C27C26C25C24C23C22C21C20	C37C36C35C34C33C32C31C
BM_CMYKQUADS	32	4	KYMC	K7K6K5K4K3K2K1K0	Y7Y6Y5Y4Y3Y2Y1Y0	M7M6M5M4M3M2M1M0
BM_KYMCQUADS	32	4	CMYK	C7C6C5C4C3C2C1C0	M7M6M5M4M3M2M1M0	Y7Y6Y5Y4Y3Y2Y1Y0
BM_10b_RGB	32	3	BGR	B7B6B5B4B3B2B1B0	G5G4G3G2G1G0B9B8	R3R2R1R0G9G8G7G6
BM_10b_XYZ	32	3	ZYX	Z7Z6Z5Z4Z3Z2Z1Z0	Y5Y4Y3Y2Y1Y0Z9Z8	X3X2X1X0Y9Y8Y7Y6
BM_10b_Yxy	32	3	yxY	y7y6y5y4y3y2y1y0	x5x4x3x2x1x0y9y8	Y3Y2Y1Y0x9x8x7x6
BM_10b_Lab	32	3	baL	b7b6b5b4b3b2b1b0	a5a4a3a2a1a0b9b8	L3L2L1L0a9a8a7a6
BM_10b_G3CH	32	3	321	C37C36C35C34C33C32C31C30	C25C24C23C22C21C20C39C38	C13C12C11C10C29C28C27C
BM_NAMED_INDEX	32			n7n6n5n4n3n2n1n0	n15n14n13n12n11n10n9n8	n23n22n21n20n19n18n17n1
BM_5CHANNEL	40	5	12345	C17C16C15C14C13C12C11C10	C27C26C25C24C23C22C21C20	C37C36C35C34C33C32C31C
BM_6CHANNEL	48	6	123456	C17C16C15C14C13C12C11C10	C27C26C25C24C23C22C21C20	C37C36C35C34C33C32C31C
BM_16b_RGB	48	3	RGB	R7R6R5R4R3R2R1R0	R15R14R13R12R11R10R9R8	G7G6G5G4G3G2G1G0
BM_16b_XYZ	48	3	XYZ	X7X6X5X4X3X2X1X0	X15X14X13X12X11X10X9X8	Y7Y6Y5Y4Y3Y2Y1Y0
BM_16b_Lab	48	3	Lab	L7L6L5L4L3L2L1L0	L15L14L13L12L11L10L9L8	a7a6a5a4a3a2a1a0
BM_16b_G3CH	48	3	321	C37C36C35C34C33C32C31C30	C315C314C313C312C311C310C39C38	C27C26C25C24C23C22C21C
BM_16b_Yxy	48	3	Yxy	Y7Y6Y5Y4Y3Y2Y1Y0	Y15Y14Y13Y12Y11Y10Y9Y8	x7x6x5x4x3x2x1x0
BM_7CHANNEL	56	7	1234567	C17C16C15C14C13C12C11C10	C27C26C25C24C23C22C21C20	C37C36C35C34C33C32C31C
BM_8CHANNEL	64	8	12345678	C17C16C15C14C13C12C11C10	C27C26C25C24C23C22C21C20	C37C36C35C34C33C32C31C

Format	Bits Per Pixel	Number of Channels	Channel Ordering	Byte 0	Byte 1	Byte 2
BM_32b_scRGB	96	3	BGR			
BM_32b_scARGB	128	3	BGRA			
BM_S2DOT13FIXED_scRGB	48	3	BGR			
BM_S2DOT13FIXED_scARGB	64	3	BGRA			
BM_R10G10B10A2	32	3	ABGR	A7A6B5B4B3B2B1B0	B7B6B5B4G3G2G1G0	G7G6G5G4G3G2R1R0
BM_R10G10B10A2_XR	32	3	ABGR	A7A6B5B4B3B2B1B0	B7B6B5B4G3G2G1G0	G7G6G5G4G3G2R1R0
BM_R16G16B16A16_FLOAT	64	3	RGBA	R7R6R5R4R3R2R1R0	R7R6R5R4R3R2R1R0	G7G6G5G4G3G2G1G0

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h

## Feedback

Was this page helpful? [!\[\]\(647e44ea77c89a016b9d36ad68afc84b\_img.jpg\) Yes](#) [!\[\]\(fe3d391417e8013a34f523acc99a48a2\_img.jpg\) No](#)

[Get help at Microsoft Q&A](#)

# CheckBitmapBits function (icm.h)

Article 10/05/2021

Checks whether the pixels in a specified bitmap lie within the output [gamut](#) of a specified transform.

## Syntax

C++

```
BOOL CheckBitmapBits(
    HTRANSFORM    hColorTransform,
    PVOID         pSrcBits,
    BMFORMAT      bmInput,
    DWORD         dwWidth,
    DWORD         dwHeight,
    DWORD         dwStride,
    PBYTE         paResult,
    PBMCALLBACKFN pfnCallback,
    LPARAM        lpCallbackData
);
```

## Parameters

`hColorTransform`

Handle to the color transform to use.

`pSrcBits`

Pointer to the bitmap to check against the output gamut.

`bmInput`

Specifies the format of the bitmap. Must be set to one of the values of the [BMFORMAT](#) enumerated type.

`dwWidth`

Specifies the number of pixels per scan line of the bitmap.

`dwHeight`

Specifies the number of scan lines of the bitmap.

`dwStride`

Specifies the number of bytes from the beginning one scan line to the beginning of the next one. If set to zero, the bitmap scan lines are assumed to be padded so as to be **DWORD**-aligned.

`paResult`

Pointer to an array of bytes where the test results are to be placed. This results buffer must contain at least as many bytes as there are pixels in the bitmap.

`pfnCallback`

Pointer to a callback function called periodically by **CheckBitmapBits** to report progress and allow the calling process to cancel the bitmap test. (See [ICMProgressProcCallback](#)).

`lpCallbackData`

Data passed back to the callback function, for example, to identify the bitmap test about which progress is being reported.

## Return value

If this function succeeds, the return value is a nonzero value.

If this function fails, the return value is zero. For extended error information, call **GetLastError**.

## Remarks

If the input format is not compatible with the color transform, the **CheckBitmapBits** function fails.

This function places results of the tests in the buffer pointed to by *paResult*. Each byte in the buffer corresponds to a pixel in the bitmap, and has an unsigned value between 0 and 255. The value 0 denotes that the color is in gamut, while a nonzero value denotes that it is out of gamut. For any integer *n* such that  $0 < n < 255$ , a result value of *n* + 1 indicates that the corresponding color is at least as far out of gamut as would be indicated by a result value of *n*.

When either of the floating point BMFORMATs, **BM\_32b\_scARGB** or **BM\_32b\_scRGB** is used, the color data being checked should not contain NaN or infinity. NaN and infinity are not considered to represent legitimate color component values, and the result of

checking pixels containing NaN or infinity is meaningless in color terms. NaN or infinity values in the color data being processed will be handled silently, and an error will not be returned.

The out-of-gamut information in the gamut tags created in WCS use the perceptual color distance in CIECAM02, which is the mean square root in CIECAM02 Jab space. The distance in the legacy ICC profile gamut tags is the mean square root in CIELAB space. We recommend that you use the CIECAM02 space when it is available because it provides more perceptually accurate distance metrics.

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [ICMProgressProcCallback](#)
- [BMFORMAT](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# CheckColors function (icm.h)

Article 02/22/2024

Determines whether the colors in an array lie within the output [gamut](#) of a specified transform.

## Syntax

C++

```
BOOL CheckColors(
    HTRANSFORM hColorTransform,
    PCOLOR     paInputColors,
    DWORD      nColors,
    COLORTYPE   ctInput,
    PBYTE      paResult
);
```

## Parameters

`hColorTransform`

Handle to the color transform to use.

`paInputColors`

Pointer to an array of *nColors* [COLOR](#) structures to translate.

`nColors`

Contains the number of elements in the arrays pointed to by *paInputColors* and *paResult*.

`ctInput`

Specifies the input color type.

`paResult`

Pointer to an array of *nColors* bytes that receives the results of the test.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Remarks

If the input color type is not compatible with the color transform, [CheckColors](#) fails.

The function places results of the tests in the array pointed to by *paResult*. Each byte in the array corresponds to a **COLOR** element in the array pointed to by *palnputColors* and has an unsigned value between 0 and 255. The value 0 denotes that the color is in gamut, while a nonzero value denotes that it is out of gamut. For any integer *n* such that  $0 < n < 255$ , a result value of *n* + 1 indicates that the corresponding color is at least as far out of gamut as would be indicated by a result value of *n*.

The out-of-gamut information in the gamut tags created in WCS use the perceptual color distance in CIECAM02, which is the mean square root in CIECAM02 Jab space. The distance in the legacy ICC profile gamut tags is the mean square root in CIELAB space. We recommend that you use the CIECAM02 space when it is available because it provides more perceptually accurate distance metrics.

## Requirements

[+] [Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [COLOR structure](#)

---

# Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# CloseColorProfile function (icm.h)

Article02/22/2024

Closes an open profile handle.

## Syntax

C++

```
BOOL CloseColorProfile(  
    HPROFILE hProfile  
) ;
```

## Parameters

`hProfile`

Handle to the profile to be closed. The function determines whether the HPROFILE contains ICC or WCS profile information.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Requirements

[+] Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib

Requirement	Value
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?



[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# CMCheckColors function (icm.h)

Determines whether given colors lie within the output [gamut](#) of a specified transform.

## Syntax

C++

```
BOOL CMCheckColors(
    HCMTRANSFORM hcmTransform,
    LPCOLOR     lpaInputColors,
    DWORD       nColors,
    COLORTYPE   ctInput,
    LPBYTE      lpaResult
);
```

## Parameters

`hcmTransform`

Handle to the color transform to use.

`lpaInputColors`

Pointer to an array of [COLOR](#) structures to check against the output gamut.

`nColors`

Specifies the number of elements in the array.

`ctInput`

Specifies the input color type.

`lpaResult`

Pointer to a buffer in which to place an array of bytes containing the test results. Each byte in the buffer corresponds to a [COLOR](#) structure, and on exit has been set to an unsigned value between 0 and 255. The value 0 denotes that the color is in gamut, while a nonzero value indicates that it is out of gamut. For any integer  $n$  such that  $0 < n < 255$ , a result value of  $n + 1$  indicates that the corresponding color is at least as far out of gamut as would be indicated by a result value of  $n$ . These values are usually generated from the *gamutTag* in the ICC profile.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. If the function is not successful, the CMM should call [SetLastError](#) to set the last error to a valid error value defined in Winerror.h.

## Remarks

Every CMM is required to export this function.

If the input color type is not compatible with the color transform **CMCheckColors** fails.

## Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Icm32.Lib

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

Last updated on 10/31/2025

# CMCheckColorsInGamut function (icm.h)

Article02/22/2024

[**CMCheckColorsInGamut** is no longer available for use as of Windows Vista.]

Determines whether specified RGB triples lie in the output [gamut](#) of a specified transform.

## Syntax

C++

```
BOOL CMCheckColorsInGamut(
    HCMTRANSFORM hcmTransform,
    RGBTRIPLE    *lpaRGBTriple,
    LPBYTE        lpaResult,
    UINT          nCount
);
```

## Parameters

`hcmTransform`

Specifies the transform to use.

`lpaRGBTriple`

Points to an array of RGB triples to check.

`lpaResult`

Points to the buffer in which to put results.

The results are represented by an array of bytes. Each byte in the array corresponds to an RGB triple and has an unsigned value between 0 and 255. The value 0 denotes that the color is in gamut, while a nonzero value denotes that it is out of gamut. For any integer  $n$  in the range  $0 < n < 255$ , a result value of  $n + 1$  indicates that the corresponding color is at least as far out of gamut as would be indicated by a result value of  $n$ .

`nCount`

Specifies the number of elements in the array.

## Return value

Beginning with Windows Vista, the default CMM (Icm32.dll) will return **FALSE** and [GetLastError](#) will report ERROR\_NOT\_SUPPORTED.

**Windows Server 2003, Windows XP and Windows 2000:**

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. Call [GetLastError](#) to retrieve the error.

## Remarks

Beginning with Windows Vista, CMM Implementors are no longer required to implement this method.

**Windows Server 2003, Windows XP and Windows 2000:**

CMM Implementors are required to implement this method.

Every CMM is required to export this function.

If the function is not successful, custom CMMs should call [SetLastError](#) to set the last error to a valid error value defined in Winerror.h.

## Requirements

[ ] [Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

# Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# CMCheckRGBs function (icm.h)

Checks bitmap colors against an output gamut.

## Syntax

C++

```
BOOL CMCheckRGBs(
    HCMTRANSFORM hcmTransform,
    LPVOID        lpSrcBits,
    BMFORMAT      bmInput,
    DWORD         dwWidth,
    DWORD         dwHeight,
    DWORD         dwStride,
    LPBYTE        lpaResult,
    PBMCALLBACKFN pfnCallback,
    LPARAM        ulCallbackData
);
```

## Parameters

hcmTransform

lpSrcBits

bmInput

dwWidth

dwHeight

dwStride

lpaResult

pfnCallback

ulCallbackData

## Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 10 Build 20348
Minimum supported server	Windows 10 Build 20348
Header	icm.h
Library	Icm32.Lib

---

Last updated on 10/31/2025

# CMConvertColorNameToIndex function (icm.h)

Converts color names in a named color space to index numbers in a color profile.

## Syntax

C++

```
BOOL CMConvertColorNameToIndex(
    HPROFILE     hProfile,
    PCOLOR_NAME  paColorName,
    PDWORD       paIndex,
    DWORD        dwCount
);
```

## Parameters

`hProfile`

The handle to a named color profile.

`paColorName`

Pointer to an array of color name structures.

`paIndex`

Pointer to an array of **DWORDS** that this function fills with the indices.

`dwCount`

The number of color names to convert.

## Return value

If this function succeeds with the conversion, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. When this occurs, the CMM should call **SetLastError** to set the last error to a valid error value defined in **Winerror.h**.

## Remarks

This function is required in the default CMM. It is optional for all other CMMs.

# Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Icm32.Lib

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

Last updated on 10/31/2025

# CMConvertIndexToColorName function (icm.h)

Transforms indices in a color space to an array of names in a named color space.

## Syntax

C++

```
BOOL CMConvertIndexToColorName(
    HPROFILE     hProfile,
    PDWORD       paIndex,
    PCOLOR_NAME  paColorName,
    DWORD        dwCount
);
```

## Parameters

`hProfile`

The handle to a color space profile.

`paIndex`

Pointer to an array of color-space index numbers.

`paColorName`

Pointer to an array of color name structures.

`dwCount`

The number of indices to convert.

## Return value

If this conversion function succeeds, the return value is TRUE.

If this function fails, the return value is FALSE. When this occurs, the CMM should call `SetLastError` to set the last error to a valid error value defined in Winerror.h.

## Remarks

This function is required in the default CMM. It is optional for all other CMMs.

# Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Icm32.Lib

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

Last updated on 10/31/2025

# CMCreateDeviceLinkProfile function (icm.h)

Creates a [device link profile](#) in the format specified by the International Color Consortium in its ICC Profile Format Specification.

## Syntax

C++

```
BOOL CMCreateDeviceLinkProfile(
    PHPROFILE pahProfiles,
    DWORD      nProfiles,
    PDWORD     padwIntents,
    DWORD      nIntents,
    DWORD      dwFlags,
    LPBYTE     *lpProfileData
);
```

## Parameters

`pahProfiles`

Pointer to an array of profile handles.

`nProfiles`

Specifies the number of profiles in the array.

`padwIntents`

An array of rendering intents.

`nIntents`

The number of elements in the array of intents.

`dwFlags`

Specifies flags to used control creation of the transform. For details, see [CMM Transform Creation Flags](#).

`lpProfileData`

Pointer to a pointer to a buffer. If successful the function allocates and fills this buffer. The calling application must free this buffer when it is no longer needed. Use the [GlobalFree](#)

function to free this buffer.

## Return value

If the function succeeds, the return value is a nonzero value.

If this function fails, the return value is zero. If the function is not successful, the CMM should call [SetLastError](#) to set the last error to a valid error value defined in Winerror.h.

## Remarks

Only the Windows default CMM is required to export this function; it is optional for all other CMMs.

If a CMM does not support [CMCreateDeviceLinkProfile](#), Windows uses the default CMM to create a device link profile.

The first and the last profiles in the array must be [device profiles](#). The other profiles can be [color space](#) or abstract profiles. Each profile's output color space must be the next profile's input color space.

The calling application must free the buffer allocated by this function and pointed to by the *lpProfileData* parameter. Use the [GlobalFree](#) function to free the buffer.

## Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Icm32.Lib

## See also

- [Basic color management concepts](#)
- [Functions](#)

- GlobalFree
- 

Last updated on 10/31/2025

# CMCreateMultiProfileTransform function (icm.h)

Accepts an array of profiles or a single [device link profile](#) and creates a color transform. This transform is a mapping from the color space specified by the first profile to that of the second profile and so on to the last one.

## Syntax

C++

```
HCMTRANSFORM CMCreateMultiProfileTransform(  
    PHPROFILE pahProfiles,  
    DWORD nProfiles,  
    PDWORD padwIntents,  
    DWORD nIntents,  
    DWORD dwFlags  
) ;
```

## Parameters

`pahProfiles`

Points to an array of profile handles.

`nProfiles`

Specifies the number of profiles in the array.

`padwIntents`

Points to an array of rendering intents. Each rendering intent is represented by one of the following values:

`INTENT_PERCEPTUAL`

`INTENT_SATURATION`

`INTENT_RELATIVE_COLORIMETRIC`

`INTENT_ABSOLUTE_COLORIMETRIC`

For more information, see [Rendering intents](#).

`nIntents`

Specifies the number of intents in the intent array. Can be 1, or the same value as `nProfiles`.

## dwFlags

Specifies flags to used control creation of the transform. For details, see [CMM Transform Creation Flags](#).

## Return value

If this function succeeds, the return value is a color transform in the range 256 to 65,535. Since only the low **WORD** of the transform is retained, valid transforms cannot exceed this range.

If this function fails, the return value is an error code having a value less than 256. When the return value is less than 256, signaling an error, the CMM should use **SetLastError** to set the last error to a valid error value as defined in Winerror.h.

## Remarks

Every CMM is required to export this function.

The array of intents specifies how profiles should be combined. The *n*th intent is used for combining the *n*th profile in the array. If only one intent is specified, it is used for the first profile, and all other profiles are combined using Match intent.

The profile handles used to create the color transform can be closed after the call to **CmCreateMultiProfileTransform** completes.

## Requirements

[+] [Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Icm32.Lib

## See also

- [Basic color management concepts](#)

- Functions

---

Last updated on 10/31/2025

# CMCreateProfile function (icm.h)

[CMCreateProfile is no longer available for use as of Windows Vista.]

Creates a display color profile from a [LOGCOLORSPACEA](#) structure.

## Syntax

C++

```
BOOL CMCreateProfile(
    LPLOGCOLORSPACEA lpColorSpace,
    LPDEVCHARACTER   *lpProfileData
);
```

## Parameters

`lpColorSpace`

Pointer to a color logical space, of which the `IcsFilename` member will be **NULL**.

`lpProfileData`

Pointer to a pointer to a buffer. If successful the function allocates and fills this buffer. It is the calling application's responsibility to free this buffer when it is no longer needed.

## Return value

Beginning with Windows Vista, the default CMM (Icm32.dll) will return **FALSE** and [GetLastError](#) will report **ERROR\_NOT\_SUPPORTED**.

**Windows Server 2003, Windows XP and Windows 2000:**

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. Call [GetLastError](#) to retrieve the error.

## Remarks

Beginning with Windows Vista, CMM Implementors are no longer required to implement this method.

**Windows Server 2003, Windows XP and Windows 2000:**

The Unicode version of this function is [CMCreateProfileW](#).

Only the Windows default CMM is required to export this function; it is optional for all other CMMs.

If a CMM does not support [CMCreateProfile](#), Windows uses the default CMM to create the profile.

The CMM should set all header fields to sensible defaults. This profile should be usable as the input profile in a transform.

The calling application must free the buffer allocated by this function and pointed to by the *lpProfileData* parameter. Use [GlobalFree](#) to free the buffer.

## Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Icm32.Lib

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [CMCreateProfileW](#)

# CMCreateProfileW function (icm.h)

[CMCreateProfileW is no longer available for use as of Windows Vista.]

Creates a display color profile from a [LOGCOLORSPACEW](#) structure.

## Syntax

C++

```
BOOL CMCreateProfileW(  
    LPLOGCOLORSPACEW lpColorSpace,  
    LPDEVCHARACTER    *lpProfileData  
)
```

## Parameters

`lpColorSpace`

Pointer to a color logical space, of which the `IcsFilename` member will be **NULL**.

`lpProfileData`

Pointer to a pointer to a buffer. If successful the function allocates and fills this buffer. It is the calling application's responsibility to free this buffer when it is no longer needed.

## Return value

Beginning with Windows Vista, the default CMM (Icm32.dll) will return **FALSE** and [GetLastError](#) will report **ERROR\_NOT\_SUPPORTED**.

**Windows Server 2003, Windows XP and Windows 2000:**

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. Call [GetLastError](#) to retrieve the error.

## Remarks

Beginning with Windows Vista, CMM Implementors are no longer required to implement this method.

**Windows Server 2003, Windows XP and Windows 2000:**

The Unicode version of this function is [CMCreateProfileW](#).

Only the Windows default CMM is required to export this function; it is optional for all other CMMs.

If a CMM does not support [CMCreateProfileW](#), Windows uses the default CMM to create the profile.

The CMM should set all header fields to sensible defaults. This profile should be usable as the input profile in a transform.

The calling application must free the buffer allocated by this function and pointed to by the *lpProfileData* parameter. Use [GlobalFree](#) to free the buffer.

## Requirements

 [Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Icm32.Lib

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

Last updated on 10/31/2025

# CMCreateTransform function (icm.h)

Deprecated. There is no replacement API because this one was no longer being used.  
Developers of alternate CMM modules are not required to implement it.

## Syntax

C++

```
HCMTRANSFORM CMCreateTransform(
    LPLOGCOLORSPACEA lpColorSpace,
    LPDEVCHARACTER   lpDevCharacter,
    LPDEVCHARACTER   lpTargetDevCharacter
);
```

## Parameters

lpColorSpace

lpDevCharacter

lpTargetDevCharacter

## Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 10 Build 20348
Minimum supported server	Windows 10 Build 20348
Header	icm.h
Library	Icm32.Lib

Last updated on 10/31/2025

# CMCreateTransformExt function (icm.h)

Article 08/23/2022

Creates a color transform that maps from an input [LOGCOLORSPACEA](#) to an optional target space and then to an output device, using a set of flags that define how the transform should be created.

## Syntax

C++

```
HCMTRANSFORM CMCreateTransformExt(
    LPLOGCOLORSPACEA lpColorSpace,
    LPDEVCHARACTER   lpDevCharacter,
    LPDEVCHARACTER   lpTargetDevCharacter,
    DWORD             dwFlags
);
```

## Parameters

`lpColorSpace`

Pointer to an input logical color space structure.

`lpDevCharacter`

Pointer to a memory-mapped device profile.

`lpTargetDevCharacter`

Pointer to a memory-mapped target profile.

`dwFlags`

Specifies flags to used control creation of the transform. For details, see [CMM transform creation flags](#).

## Return value

If this function succeeds, the return value is a color transform in the range 256 to 65,535. Since only the low **WORD** of the transform is retained, valid transforms cannot exceed this range.

If this function fails, the return value is an error code having a value less than 256. When the return value is less than 256, signaling an error, the CMM should use **SetLastError** to set the last error to a valid error value as defined in Winerror.h.

## Remarks

The Unicode equivalent of **CMCreateTransformExt** is [CMCreateTransformExtW](#).

Every CMM is required to export this function.

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [CMCreateTransformExtW](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# CMCreateTransformExtW function (icm.h)

Creates a color transform that maps from an input [LOGCOLORSPACEW](#) to an optional target space and then to an output device, using a set of flags that define how the transform should be created.

## Syntax

C++

```
HCMTRANSFORM CMCreateTransformExtW(
    LPLOGCOLORSPACEW lpColorSpace,
    LPDEVCHARACTER   lpDevCharacter,
    LPDEVCHARACTER   lpTargetDevCharacter,
    DWORD            dwFlags
);
```

## Parameters

`lpColorSpace`

Pointer to an input logical color space structure.

`lpDevCharacter`

Pointer to a memory-mapped device profile.

`lpTargetDevCharacter`

Pointer to a memory-mapped target profile.

`dwFlags`

Specifies flags to used control creation of the transform. For details, see [CMM transform creation flags](#).

## Return value

If this function succeeds, the return value is a color transform in the range 256 to 65,535. Since only the low **WORD** of the transform is retained, valid transforms cannot exceed this range.

If this function fails, the return value is an error code having a value less than 256. When the return value is less than 256, signaling an error, the CMM should use [SetLastError](#) to set the

last error to a valid error value as defined in Winerror.h.

## Remarks

Every CMM is required to export this function.

## Requirements

 [Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Icm32.Lib

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [CMCreateTransformExtW](#)

---

Last updated on 10/31/2025

# CMCreateTransformW function (icm.h)

Deprecated. There is no replacement API because this one was no longer being used.  
Developers of alternate CMM modules are not required to implement it.

## Syntax

C++

```
HCMTRANSFORM CMCreateTransformW(
    LPLOGCOLORSPACEW lpColorSpace,
    LPDEVCHARACTER   lpDevCharacter,
    LPDEVCHARACTER   lpTargetDevCharacter
);
```

## Parameters

lpColorSpace

lpDevCharacter

lpTargetDevCharacter

## Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 10 Build 20348
Minimum supported server	Windows 10 Build 20348
Header	icm.h
Library	Icm32.Lib

Last updated on 10/31/2025

# CMDeleteTransform function (icm.h)

Deletes a specified color transform, and frees any memory associated with it.

## Syntax

C++

```
BOOL CMDeleteTransform(  
    HCMTRANSFORM hcmTransform  
)
```

## Parameters

`hcmTransform`

Identifies the color transform to be deleted.

## Return value

If this function succeeds, the return value is **TRUE**.

If the function fails, the return value is **FALSE**. If the **CMDeleteTransform** function is not successful, the CMM should call **SetLastError** to set the last error to a valid error value defined in Winerror.h.

## Remarks

Every CMM is required to export this function.

## Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h

Requirement	Value
Library	Icm32.Lib

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

Last updated on 10/31/2025

# CMGetInfo function (icm.h)

Retrieves various information about the color management module (CMM).

Every CMM is required to export this function.

## Syntax

C++

```
DWORD CMGetInfo(  
    DWORD dwInfo  
) ;
```

## Parameters

`dwInfo`

Specifies what information should be retrieved. This parameter can take one of the following constant values.

 Expand table

Constant	Significance of the function's return value
CMM_DESCRIPTION	A text string that describes the color management module.
CMM_DLL_VERSION	Version number of the CMM DLL.
CMM_DRIVER_LEVEL	Driver compatibility information.
CMM_IDENT	The CMM identification signature registered with the International Color Consortium (ICC).
CMM_LOGOICON	The logo icon for this CMM.
CMM_VERSION	Version of Windows supported.
CMM_WIN_VERSION	Backward compatibility with Windows 95.

## Return value

If this function succeeds, the return value is the same nonzero value that was passed in through the `dwInfo` parameter. If the function fails, the return value is zero.

## Remarks

The **CMGetInfo** function can be called by applications directly to obtain information about the CMM. Applications should not call other CMM functions directly. To obtain CMM information, get the path to the CMM from the registry. Invoke the Windows API function [GetModuleHandle](#) and pass the file name of the CMM as the value of its parameter. Call the **CMGetInfo** function and pass it the constant **CMM\_DESCRIPTION** as the value of its parameter. Call the [LoadString](#) function. Pass the module handle as the first parameter, and the return value of the **CMGetInfo** function as the value of the second parameter.

CMMs that do not run on Windows 95 should return 0x0050000 for **CMM\_WIN\_VERSION**.

## Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Icm32.Lib

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

Last updated on 10/31/2025

# CMGetNamedProfileInfo function (icm.h)

Retrieves information about the specified named color profile.

## Syntax

C++

```
BOOL CMGetNamedProfileInfo(  
    HPROFILE           hProfile,  
    PNAMED_PROFILE_INFO pNamedProfileInfo  
) ;
```

## Parameters

`hProfile`

The handle to the profile from which the information will be retrieved.

`pNamedProfileInfo`

A pointer to a **NAMED\_PROFILE\_INFO** structure.

## Return value

If this function succeeds, the return value is TRUE.

If this function fails, the return value is FALSE. When this occurs, the CMM should call **SetLastError** to set the last error to a valid error value defined in Winerror.h.

## Remarks

This function is required in the default CMM. It is optional for all other CMMs.

## Requirements

[ ] [Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]

Requirement	Value
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Icm32.Lib

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [NAMED\\_PROFILE\\_INFO](#)

---

Last updated on 10/31/2025

# CMGetPS2ColorRenderingDictionary function (icm.h)

Article02/22/2024

## Syntax

C++

```
BOOL CMGetPS2ColorRenderingDictionary(
    HPROFILE hProfile,
    DWORD     dwIntent,
    LPBYTE    lpBuffer,
    LPDWORD   lpcbSize,
    LPBOOL    lpbBinary
);
```

## Parameters

hProfile

dwIntent

lpBuffer

lpcbSize

lpbBinary

## Requirements

[+] Expand table

Requirement	Value
Minimum supported client	Windows 10 Build 20348
Minimum supported server	Windows 10 Build 20348
Header	icm.h

# Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# CMGetPS2ColorRenderingIntent function (icm.h)

Article 02/22/2024

Retrieves the PostScript Level 2 color rendering intent from a profile.

## Syntax

C++

```
BOOL CMGetPS2ColorRenderingIntent(
    HPROFILE hProfile,
    DWORD     dwIntent,
    LPBYTE    lpBuffer,
    LPDWORD   lpcbSize
);
```

## Parameters

`hProfile`

Specifies the profile to use.

`dwIntent`

Specifies the desired rendering intent to retrieve. Can be one of the following values:

- INTENT\_PERCEPTUAL
- INTENT\_SATURATION
- INTENT\_RELATIVE\_COLORIMETRIC
- INTENT\_ABSOLUTE\_COLORIMETRIC

For more information, see [Rendering Intents](#).

`lpBuffer`

Points to a buffer in which the color rendering intent is to be placed. If the pointer is NULL, the function returns the size required for this buffer in `*lpcbSize`.

`lpcbSize`

Points to a variable specifying the size of the buffer. On return, the variable contains has the number of bytes actually copied to the buffer.

## Return value

If this function succeeds, the return value is TRUE. It also returns TRUE if it is called with *lpBuffer* set to NULL and the size of the required buffer is copied into *lpcbSize*.

If this function fails, the return value is FALSE. When this occurs, the CMM should call **SetLastError** to set the last error to a valid error value defined in Winerror.h.

## Remarks

This function is optional for all CMMs.

If a CMM does not support this function, Windows uses the default CMM to get the color rendering intent.

If the tag is not present in the profile indicated by *hProfile*, the CMM creates it. The resulting rendering intent can be used as the operand for the PostScript Level 2 **findcolorrendering** operator.

## Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# CMGetPS2ColorSpaceArray function (icm.h)

Article02/22/2024

## Syntax

C++

```
BOOL CMGetPS2ColorSpaceArray(
    HPROFILE hProfile,
    DWORD     dwIntent,
    DWORD     dwCSAType,
    LPBYTE    lpBuffer,
    LPDWORD   lpcbSize,
    LPBOOL    lpbBinary
);
```

## Parameters

hProfile

dwIntent

dwCSAType

lpBuffer

lpcbSize

lpbBinary

## Requirements

[ ] Expand table

Requirement	Value
Minimum supported client	Windows 10 Build 20348
Minimum supported server	Windows 10 Build 20348

Requirement	Value
Header	icm.h

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# CMIsProfileValid function (icm.h)

Reports whether the given profile is a valid ICC profile that can be used for color management.

## Syntax

C++

```
BOOL CMIsProfileValid(
    HPROFILE hProfile,
    LPBOOL    lpbValid
);
```

## Parameters

`hProfile`

Specifies the profile to check.

`lpbValid`

Pointer to a variable that is set on exit to TRUE if the profile is a valid ICC profile, or FALSE if not.

## Return value

If this function succeeds, the return value is TRUE.

If this function fails, the return value is FALSE. If the function fails, the CMM should call `SetLastError` to set the last error to a valid error value defined in Winerror.h.

## Remarks

Only the Windows default CMM is required to export this function; it is optional for all other CMMs.

If a CMM does not support this function, Windows uses the default CMM to validate the profile.

## Requirements

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Icm32.Lib

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

Last updated on 10/31/2025

# CMTranslateColors function (icm.h)

Translates an array of colors from a source [color space](#) to a destination color space using a color transform.

## Syntax

C++

```
BOOL CMTranslateColors(
    HCMTRANSFORM hcmTransform,
    LPCOLOR     lpaInputColors,
    DWORD       nColors,
    COLORTYPE   ctInput,
    LPCOLOR     lpaOutputColors,
    COLORTYPE   ctOutput
);
```

## Parameters

`hcmTransform`

Specifies the color transform to use.

`lpaInputColors`

Points to an array of [COLOR](#) structures to translate.

`nColors`

Specifies the number of elements in the array.

`ctInput`

Specifies the color type of the input.

`lpaOutputColors`

Points to a buffer in which an array of translated [COLOR](#) structures is to be placed.

`ctOutput`

Specifies the output color type.

## Return value

If this function succeeds, the return value is TRUE.

If this function fails, the return value is FALSE. The CMM should call **SetLastError** to set the last error to a valid error value defined in Winerror.h.

## Remarks

Every CMM is required to export this function.

If the input and the output color types are not compatible with the color transform, this function should fail.

Note that this function must support in-place translation. That is, whenever the memory footprint of the output is less than or equal to the memory footprint of the input, this function must be able to translate the bitmap colors even if the source and destination buffers are the same.

## Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Icm32.Lib

## See also

- [Basic color management concepts](#)
- [Functions](#)

# CMTranslateRGB function (icm.h)

Translates an application-supplied RGBQuad into the device [color space](#).

## Syntax

C++

```
BOOL CMTranslateRGB(
    HCMTRANSFORM hcmTransform,
    COLORREF     ColorRef,
    LPCOLORREF   lpColorRef,
    DWORD        dwFlags
);
```

## Parameters

`hcmTransform`

Specifies the transform to be used.

`ColorRef`

The RGBQuad to translate.

`lpColorRef`

Points to a buffer in which to place the translation.

`dwFlags`

Specifies how the transform should be used to make the translation. This parameter can take one of the following meanings.

 [Expand table](#)

Value	Meaning
<code>CMS_FORWARD</code>	Use forward transform
<code>CMS_BACKWARD</code>	Use reverse transform

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. The CMM should call **SetLastError** to set the last error to a valid error value defined in **Winerror.h**.

## Remarks

Every CMM is required to export this function.

## Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Icm32.Lib

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

Last updated on 10/31/2025

# CMTranslateRGBs function (icm.h)

Article10/21/2021

[**CMTranslateRGBs** is no longer available for use as of Windows Vista.]

Translates a bitmap from one [color space](#) to another using a color transform.

## Syntax

C++

```
BOOL CMTranslateRGBs(
    HCMTRANSFORM hcmTransform,
    LPVOID        lpSrcBits,
    BMFORMAT      bmInput,
    DWORD         dwWidth,
    DWORD         dwHeight,
    DWORD         dwStride,
    LPVOID        lpDestBits,
    BMFORMAT      bmOutput,
    DWORD         dwTranslateDirection
);
```

## Parameters

**hcmTransform**

Specifies the color transform to use.

**lpSrcBits**

Points to the bitmap to translate.

**bmInput**

Specifies the input bitmap format.

**dwWidth**

Specifies the number of pixels per scan line in the input bitmap.

**dwHeight**

Specifies the number of scan lines in the input bitmap.

`dwStride`

Specifies the number of bytes from the beginning of one scan line to the beginning of the next in the input bitmap. If `dwStride` is set to zero, the CMM should assume that scan lines are padded so as to be **DWORD**-aligned.

`lpDestBits`

Points to a destination buffer in which to place the translated bitmap.

`bmOutput`

Specifies the output bitmap format.

`dwTranslateDirection`

Specifies the direction of the transform being used for the translation. This parameter must take one of the following values.

<b>Value</b>	<b>Meaning</b>
<code>CMS_FORWARD</code>	Use forward transform
<code>CMS_BACKWARD</code>	Use reverse transform

## Return value

Beginning with Windows Vista, the default CMM (`Icm32.dll`) will return **FALSE** and [GetLastError](#) will report `ERROR_NOT_SUPPORTED`.

**Windows Server 2003, Windows XP and Windows 2000:**

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. If the function is not successful, the CMM should call [SetLastError](#) to set the last error to a valid error value defined in `Winerror.h`.

## Remarks

Beginning with Windows Vista, CMM Implementors are no longer required to implement this method.

**Windows Server 2003, Windows XP and Windows 2000:**

Every CMM is required to export this function.

When writing into the destination buffer, the CMM should make sure that scan lines are padded to be **DWORD**-aligned.

If the input and output formats are not compatible with the color transform, this function fails.

If both input and output bitmap formats are 3-channel, 4 bytes-per-pixel as in the case of BM\_xRGBQUADS, the 4th byte should be preserved and copied to the output buffer.

Note that this function must support in-place translation. That is, whenever the memory footprint of the output is less than or equal to the memory footprint of the input, this function must be able to translate the bitmap colors even if the source and destination buffers are the same.

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?



[Get help at Microsoft Q&A](#)

# CMTranslateRBGsExt function (icm.h)

Article 08/23/2022

Translates a bitmap from one defined format into a different defined format and calls a callback function periodically, if one is specified, to report progress and permit the calling application to terminate the translation.

## Syntax

C++

```
BOOL CMTranslateRBGsExt(
    HCMTRANSFORM    hcmTransform,
    LPVOID          lpSrcBits,
    BMFORMAT        bmInput,
    DWORD           dwWidth,
    DWORD           dwHeight,
    DWORD           dwInputStride,
    LPVOID          lpDestBits,
    BMFORMAT        bmOutput,
    DWORD           dwOutputStride,
    LPBMCALLBACKFN lpfnCallback,
    LPARAM          ulCallbackData
);
```

## Parameters

`hcmTransform`

Specifies the color transform to use.

`lpSrcBits`

Pointer to the bitmap to translate.

`bmInput`

Specifies the input bitmap format.

`dwWidth`

Specifies the number of pixels per scan line in the input bitmap.

`dwHeight`

Specifies the number of scan lines in the input bitmap.

`dwInputStride`

Specifies the number of bytes from the beginning of one scan line to the beginning of the next in the input bitmap. If `dwInputStride` is set to zero, the CMM should assume that scan lines are padded so as to be **DWORD**-aligned.

`lpDestBits`

Points to a destination buffer in which to place the translated bitmap.

`bmOutput`

Specifies the output bitmap format.

`dwOutputStride`

Specifies the number of bytes from the beginning of one scan line to the beginning of the next in the input bitmap. If `dwOutputStride` is set to zero, the CMM should pad scan lines so that they are **DWORD**-aligned.

`lpfnCallback`

Pointer to an application-supplied callback function called periodically by **CMTranslateRBGsExt** to report progress and allow the calling process to cancel the translation. (See [ICMProgressProcCallback](#).)

`ulCallbackData`

Data passed back to the callback function, for example to identify the translation that is reporting progress.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE** and the CMM should call **SetLastError** to set the last error to a valid error value defined in **Winerror.h**.

## Remarks

Every CMM is required to export this function.

When writing into the destination buffer, the CMM should make sure that scan lines are padded to be **DWORD**-aligned.

If the input and output formats are not compatible with the color transform, this function fails.

If both input and output bitmap formats are 3 channel, 4 bytes per pixel as in the case of BM\_xRGBQUADS, the fourth bytes should be preserved and copied to the output buffer.

If the callback function returns zero, processing should be cancelled and **CMTranslateRBGsExt** should return zero to indicate failure; the output buffer may be partially filled.

Note that this function must support in-place translation. That is, whenever the memory footprint of the output is less than or equal to the memory footprint of the input, this function must be able to translate the bitmap colors even if the source and destination buffers are the same.

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h

## See also

- [Basic color management concepts](#))
- [Functions](#)

## Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# CMYKCOLOR structure (icm.h)

Article02/22/2024

Description of the CMYKCOLOR structure.

## Syntax

C++

```
struct CMYKCOLOR {
    WORD cyan;
    WORD magenta;
    WORD yellow;
    WORD black;
};
```

## Members

cyan

TBD

magenta

TBD

yellow

TBD

black

TBD

## Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]

Requirement	Value
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# COLOR union (icm.h)

Article02/22/2024

Description of the COLOR union.

## Syntax

C++

```
typedef union tagCOLOR {
    struct GRAYCOLOR      gray;
    struct RGBCOLOR       rgb;
    struct CMYKCOLOR      cmyk;
    struct XYZCOLOR       XYZ;
    struct YxyCOLOR       Yxy;
    struct LabCOLOR       Lab;
    struct GENERIC3CHANNEL gen3ch;
    struct NAMEDCOLOR     named;
    struct HiFiCOLOR      hifi;
    struct {
        DWORD reserved1;
        VOID  *reserved2;
    };
} COLOR;
```

## Members

gray

TBD

rgb

TBD

cmyk

TBD

XYZ

TBD

Yxy

TBD

Lab

TBD

gen3ch

TBD

named

TBD

hifi

TBD

reserved1

TBD

reserved2

TBD

## Remarks

A variable of type COLOR may be accessed as any of the supported color space colors by accessing the appropriate member of the union. For instance, given the following variable declaration:

```
COLOR aColor;
```

the red, green and blue values could be set in the following manner:

```
aColor.rgb.red=100;
```

```
aColor.rgb.green=50;
```

```
aColor.rgb.blue=2;
```

## Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) | [Get help at Microsoft Q&A](#)

# COLORDATATYPE enumeration (icm.h)

Article 02/22/2024

Used by WCS functions to indicate the data type of vector content.

## Syntax

C++

```
typedef enum {
    COLOR_BYTE = 1,
    COLOR_WORD,
    COLOR_FLOAT,
    COLOR_S2DOT13FIXED,
    COLOR_10b_R10G10B10A2,
    COLOR_10b_R10G10B10A2_XR,
    COLOR_FLOAT16
} COLORDATATYPE;
```

## Constants

  Expand table

**COLOR\_BYTE**

Value: 1

Color data is stored as 8 bits per channel, with a value from 0 to 255, inclusive.

**COLOR\_WORD**

Color data is stored as 16 bits per channel, with a value from 0 to 65535, inclusive.

**COLOR\_FLOAT**

Color data is stored as 32 bits value per channel, as defined by the IEEE 32-bit floating point standard.

**COLOR\_S2DOT13FIXED**

Color data is stored as 16 bits per channel, with a fixed range of -4 to +4, inclusive. A signed format is used, with 1 bit for the sign, 2 bits for the integer portion, and 13 bits for the fractional portion.

**COLOR\_10b\_R10G10B10A2**

Color data is stored as 10 bits per channel. The two most significant bits are alpha.

#### COLOR\_10b\_R10G10B10A2\_XR

Color data is stored as 10 bits per channel, 32 bits per pixel. The 10 bits of each color channel are 2.8 fixed point with a -0.75 bias, giving a range of [-0.76 .. 1.25]. This range corresponds to [-0.5 .. 1.5] in a gamma = 1 space. The two most significant bits are preserved for alpha.

This uses an extended range (XR) sRGB color space. It has the same RGB primaries, white point, and gamma as sRGB.

#### COLOR\_FLOAT16

Color data is stored as 16 bits value per channel, as defined by the IEEE 16-bit floating point standard.

## Remarks

The PCOLORDATATYPE and LPCOLORDATATYPE data types are defined as pointers to the COLORDATATYPE enumeration:

```
typedef COLORDATATYPE *PCOLORDATATYPE, *LPCOLORDATATYPE;
```

## Requirements

[\[+\] Expand table](#)

Requirement	Value
Minimum supported client	Windows Vista [desktop apps only]
Minimum supported server	Windows Server 2008 [desktop apps only]
Header	icm.h

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# COLORMATCHSETUPA structure (icm.h)

Article 07/27/2022

The **COLORMATCHSETUP** structure contains information that the [SetupColorMatchingW](#) function uses to initialize the **ColorManagement** dialog box.

After the user closes the dialog box, **SetupColorMatching** returns information about the user's selection in this structure.

## Syntax

C++

```
typedef struct _tagCOLORMATCHSETUPA {
    DWORD          dwSize;
    DWORD          dwVersion;
    DWORD          dwFlags;
    HWND           hwndOwner;
    PCSTR          pSourceName;
    PCSTR          pDisplayName;
    PCSTR          pPrinterName;
    DWORD          dwRenderIntent;
    DWORD          dwProofingIntent;
    PSTR           pMonitorProfile;
    DWORD          ccMonitorProfile;
    PSTR           pPrinterProfile;
    DWORD          ccPrinterProfile;
    PSTR           pTargetProfile;
    DWORD          ccTargetProfile;
    DLGPROC        lpfnHook;
    LPARAM         lParam;
    PCMSCALLBACKA lpfnApplyCallback;
    LPARAM         lParamApplyCallback;
} COLORMATCHSETUPA, *PCOLORMATCHSETUPA, *LPCOLORMATCHSETUPA;
```

## Members

**dwSize**

Size of the structure. Should be set to **sizeof ( COLORMATCHSETUP )**.

**dwVersion**

Version of the **COLORMATCHSETUP** structure. This should be set to **COLOR\_MATCH\_VERSION**.

## **dwFlags**

A set of bit flags used to initialize the dialog box. If set to 0 on entry, all controls assume their default states.

When the dialog box returns, these flags are set to indicate the user's input.

This member can be set using a combination of the following flags.

<b>Flag</b>	<b>Meaning</b>
CMS_DISABLEICM	If set on entry, this flag indicates that the "Enable Color Management" check box is cleared, disabling all other controls. If set on exit, it means that the user does not wish color management performed.
CMS_ENABLEPROOFING	If set on entry, this flag indicates that the Proofing controls are to be enabled, and the Proofing check box is checked. If set on exit, it means that the user wishes to perform color management for a different target device than the selected printer.
CMS_SETRENDERINTENT	If set on entry, this flag indicates that the <b>dwRenderIntent</b> member contains the value to use to initialize the Rendering Intent control. Otherwise, the control defaults to Picture rendering. This flag is set on exit if WCS is enabled.
CMS_SETPROOFINTENT	Ignored unless CMS_ENABLEPROOFING is also set. If set on entry, and CMS_ENABLEPROOFING is also set, this flag indicates that the <b>dwProofingIntent</b> member is to be used to initialize the Target Rendering Intent control. Otherwise, the control defaults to Picture rendering. This flag is set on exit if proofing is enabled.
CMS_SETMONITORPROFILE	If set on entry, this flag indicates that the color management profile named in the <b>pMonitorProfile</b> member is to be the initial selection in the monitor profile control. If the specified profile is not associated with the monitor, this flag is ignored, and the default profile for the monitor is used.
CMS_SETPRINTERPROFILE	If set on entry, this flag indicates that the color management profile named in the <b>pPrinterProfile</b> member is to be the initial selection in the printer profile control. If the specified profile is not associated with the printer, this flag is ignored, and the default profile for the printer is used.
CMS_SETTARGETPROFILE	If set on entry, this flag indicates that the color profile named in the <b>pTargetProfile</b> member is to be the initial selection in the target profile control. If the specified profile is not installed, this flag is ignored, and the default profile for the printer is used. If the printer has no default profile, then the first profile in alphabetical order will be displayed.

Flag	Meaning
CMS_USEHOOK	This flag specifies that the <i>lpfnHook</i> member contains the address of a hook procedure, and the <i>IParam</i> member contains a value to be passed to the hook procedure when the WM_INITDIALOG message is sent.
CMS_MONITOROVERFLOW	This flag is set on exit if color management is to be enabled and the buffer size given in <i>ccMonitorProfile</i> is insufficient for the selected profile name. <b>GetLastError</b> returns ERROR_INSUFFICIENT_BUFFER in such a case.
CMS_PRINTEROVERFLOW	This flag is set on exit if color management is to be enabled and the buffer size given in <i>ccPrinterProfile</i> is insufficient for the selected profile name. <b>GetLastError</b> returns ERROR_INSUFFICIENT_BUFFER in such a case.
CMS_TARGETOVERFLOW	This flag is set on exit if proofing is to be enabled and the buffer size given in <i>ccTargetProfile</i> is insufficient for the selected profile name. <b>GetLastError</b> returns ERROR_INSUFFICIENT_BUFFER in such a case.
CMS_USEAPPLYCALLBACK	If set on entry, this flag indicates that the <b>SetupColorMatching</b> function should call the function <b>PCMSCALLBACKW</b> . The address of the callback function is contained in <i>lpfnApplyCallback</i> .
CMS_USEDESCRIPTION	If set on entry, this flag instructs the <b>SetupColorMatching</b> function to retrieve the profile description contained in the profile description tags (See ICC Profile Format Specification v3.4). It will insert them into the <b>Monitor Profile</b> , <b>Printer Profile</b> , <b>Emulated Device Profile</b> edit boxes in the <b>Color Management</b> common dialog box.

**hwndOwner**

The window handle to the owner of the dialog box, or **NULL** if the dialog box has no owner.

**pSourceName**

Pointer to an application-specified string which describes the source profile of the item for which color management is to be performed. If this is **NULL**, the Image Source control displays the name of the Windows default color profile.

**pDisplayName**

Points to a string naming the monitor to be used for color management. If this is not the name of a valid monitor, the first enumerated monitor is used.

`pPrinterName`

Points to a string naming the printer on which the image is to be rendered. If this is not a valid printer name, the default printer is used and named in the dialog.

`dwRenderIntent`

The type of color management desired. Valid values are:

`INTENT_PERCEPTUAL`

`INTENT_SATURATION`

`INTENT_RELATIVE_COLORIMETRIC`

`INTENT_ABSOLUTE_COLORIMETRIC`

For more information, see [Rendering intents](#).

`dwProofingIntent`

The type of color management desired for the proofed image. Valid values are:

`INTENT_PERCEPTUAL`

`INTENT_SATURATION`

`INTENT_RELATIVE_COLORIMETRIC`

`INTENT_ABSOLUTE_COLORIMETRIC`

For more information, see [Rendering intents](#).

`pMonitorProfile`

Pointer to a buffer in which to place the name of the user-selected monitor profile. If the CMS\_SETMONITORPROFILE flag is used, this flag can also be used to select a profile other than the monitor default when the dialog is first displayed.

`ccMonitorProfile`

The size of the buffer pointed to by the `pMonitorProfile` member, in characters. If the buffer is not large enough to hold the selected name, the name is truncated to this size, and `ERROR_INSUFFICIENT_BUFFER` is returned. A buffer of `MAX_PATH` size always works.

`pPrinterProfile`

Points to a buffer in which to place the name of the user-selected printer profile. If the CMS\_SETPRINTERPROFILE flag is used, this flag can also be used to select a profile other than the printer default when the dialog is first displayed.

#### `ccPrinterProfile`

The size of the buffer pointed to by the **pPrinterProfile** member, in characters. If the buffer is not large enough to hold the selected name, the name is truncated to this size, and ERROR\_INSUFFICIENT\_BUFFER is returned. A buffer of MAX\_PATH size always works.

#### `pTargetProfile`

Points to a buffer in which to place the name of the user-selected target profile for proofing. If the CMS\_SETTARGETPROFILE flag is used, this flag can also be used to select a profile other than the printer default when the dialog is first displayed.

#### `ccTargetProfile`

The size of the buffer pointed to by the **pTargetProfile** member, in characters. If the buffer is not large enough to hold the selected name, the name is truncated to this size, and ERROR\_INSUFFICIENT\_BUFFER is returned. A buffer of MAX\_PATH size always works.

#### `lpfnHook`

If the CMS\_USEHOOK flag is set, this member is the address of a dialog procedure (see [DialogProc](#)) that can filter or handle messages for the dialog. The hook procedure receives no messages issued before WM\_INITDIALOG. It is called on the WM\_INITDIALOG message after the system-provided dialog procedure has processed the message. On all other messages, the hook procedure receives the message before the system-provided procedure. If the hook procedure returns **TRUE** to these messages, the system-provided procedure is not called.

The hook procedure may call the [EndDialog](#) function.

#### `lParam`

If the CMS\_USEHOOK flag is set, this member is passed to the application-provided hook procedure as the *lParam* parameter when the WM\_INITDIALOG message is processed.

#### `lpfnApplyCallback`

Contains a pointer to a callback function that is invoked when the **Apply** button of the Color Management dialog box is selected. If no callback function is provided, this member should be set to **NULL**. See [PCMSCALLBACKW](#).

## `IParamApplyCallback`

Contains a value that will be passed to the function `ApplyCallbackFunction` through its `IParam` parameter. The meaning and content of the value is specified by the application.

# Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h

## See also

- [A common dialog for color management](#)
- [DialogProc ↗](#)
- [EndDialog ↗](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# COLORMATCHSETUPW structure (icm.h)

Article07/27/2022

The **COLORMATCHSETUP** structure contains information that the [SetupColorMatchingW](#) function uses to initialize the **ColorManagement** dialog box.

After the user closes the dialog box, [SetupColorMatching](#) returns information about the user's selection in this structure.

## Syntax

C++

```
typedef struct _tagCOLORMATCHSETUPW {
    DWORD          dwSize;
    DWORD          dwVersion;
    DWORD          dwFlags;
    HWND           hwndOwner;
    PCWSTR         pSourceName;
    PCWSTR         pDisplayName;
    PCWSTR         pPrinterName;
    DWORD          dwRenderIntent;
    DWORD          dwProofingIntent;
    PWSTR          pMonitorProfile;
    DWORD          ccMonitorProfile;
    PWSTR          pPrinterProfile;
    DWORD          ccPrinterProfile;
    PWSTR          pTargetProfile;
    DWORD          ccTargetProfile;
    DLGPROC        lpfnHook;
    LPARAM         lParam;
    PCMSCALLBACKW lpfnApplyCallback;
    LPARAM         lParamApplyCallback;
} COLORMATCHSETUPW, *PCOLORMATCHSETUPW, *LPCOLORMATCHSETUPW;
```

## Members

**dwSize**

Size of the structure. Should be set to **sizeof ( COLORMATCHSETUP )**.

**dwVersion**

Version of the **COLORMATCHSETUP** structure. This should be set to **COLOR\_MATCH\_VERSION**.

## **dwFlags**

A set of bit flags used to initialize the dialog box. If set to 0 on entry, all controls assume their default states.

When the dialog box returns, these flags are set to indicate the user's input.

This member can be set using a combination of the following flags.

<b>Flag</b>	<b>Meaning</b>
CMS_DISABLEICM	If set on entry, this flag indicates that the "Enable Color Management" check box is cleared, disabling all other controls. If set on exit, it means that the user does not wish color management performed.
CMS_ENABLEPROOFING	If set on entry, this flag indicates that the Proofing controls are to be enabled, and the Proofing check box is checked. If set on exit, it means that the user wishes to perform color management for a different target device than the selected printer.
CMS_SETRENDERINTENT	If set on entry, this flag indicates that the <b>dwRenderIntent</b> member contains the value to use to initialize the Rendering Intent control. Otherwise, the control defaults to Picture rendering. This flag is set on exit if WCS is enabled.
CMS_SETPROOFINTENT	Ignored unless CMS_ENABLEPROOFING is also set. If set on entry, and CMS_ENABLEPROOFING is also set, this flag indicates that the <b>dwProofingIntent</b> member is to be used to initialize the Target Rendering Intent control. Otherwise, the control defaults to Picture rendering. This flag is set on exit if proofing is enabled.
CMS_SETMONITORPROFILE	If set on entry, this flag indicates that the color management profile named in the <b>pMonitorProfile</b> member is to be the initial selection in the monitor profile control. If the specified profile is not associated with the monitor, this flag is ignored, and the default profile for the monitor is used.
CMS_SETPRINTERPROFILE	If set on entry, this flag indicates that the color management profile named in the <b>pPrinterProfile</b> member is to be the initial selection in the printer profile control. If the specified profile is not associated with the printer, this flag is ignored, and the default profile for the printer is used.
CMS_SETTARGETPROFILE	If set on entry, this flag indicates that the color profile named in the <b>pTargetProfile</b> member is to be the initial selection in the target profile control. If the specified profile is not installed, this flag is ignored, and the default profile for the printer is used. If the printer has no default profile, then the first profile in alphabetical order will be displayed.

Flag	Meaning
CMS_USEHOOK	This flag specifies that the <i>lpfnHook</i> member contains the address of a hook procedure, and the <i>IParam</i> member contains a value to be passed to the hook procedure when the WM_INITDIALOG message is sent.
CMS_MONITOROVERFLOW	This flag is set on exit if color management is to be enabled and the buffer size given in <i>ccMonitorProfile</i> is insufficient for the selected profile name. <b>GetLastError</b> returns ERROR_INSUFFICIENT_BUFFER in such a case.
CMS_PRINTEROVERFLOW	This flag is set on exit if color management is to be enabled and the buffer size given in <i>ccPrinterProfile</i> is insufficient for the selected profile name. <b>GetLastError</b> returns ERROR_INSUFFICIENT_BUFFER in such a case.
CMS_TARGETOVERFLOW	This flag is set on exit if proofing is to be enabled and the buffer size given in <i>ccTargetProfile</i> is insufficient for the selected profile name. <b>GetLastError</b> returns ERROR_INSUFFICIENT_BUFFER in such a case.
CMS_USEAPPLYCALLBACK	If set on entry, this flag indicates that the <b>SetupColorMatching</b> function should call the function <b>PCMSCALLBACKW</b> . The address of the callback function is contained in <i>lpfnApplyCallback</i> .
CMS_USEDESCRIPTION	If set on entry, this flag instructs the <b>SetupColorMatching</b> function to retrieve the profile description contained in the profile description tags (See ICC Profile Format Specification v3.4). It will insert them into the <b>Monitor Profile</b> , <b>Printer Profile</b> , <b>Emulated Device Profile</b> edit boxes in the <b>Color Management</b> common dialog box.

**hwndOwner**

The window handle to the owner of the dialog box, or **NULL** if the dialog box has no owner.

**pSourceName**

Pointer to an application-specified string which describes the source profile of the item for which color management is to be performed. If this is **NULL**, the Image Source control displays the name of the Windows default color profile.

**pDisplayName**

Points to a string naming the monitor to be used for color management. If this is not the name of a valid monitor, the first enumerated monitor is used.

`pPrinterName`

Points to a string naming the printer on which the image is to be rendered. If this is not a valid printer name, the default printer is used and named in the dialog.

`dwRenderIntent`

The type of color management desired. Valid values are:

`INTENT_PERCEPTUAL`

`INTENT_SATURATION`

`INTENT_RELATIVE_COLORIMETRIC`

`INTENT_ABSOLUTE_COLORIMETRIC`

For more information, see [Rendering intents](#).

`dwProofingIntent`

The type of color management desired for the proofed image. Valid values are:

`INTENT_PERCEPTUAL`

`INTENT_SATURATION`

`INTENT_RELATIVE_COLORIMETRIC`

`INTENT_ABSOLUTE_COLORIMETRIC`

For more information, see [Rendering intents](#).

`pMonitorProfile`

Pointer to a buffer in which to place the name of the user-selected monitor profile. If the CMS\_SETMONITORPROFILE flag is used, this flag can also be used to select a profile other than the monitor default when the dialog is first displayed.

`ccMonitorProfile`

The size of the buffer pointed to by the `pMonitorProfile` member, in characters. If the buffer is not large enough to hold the selected name, the name is truncated to this size, and `ERROR_INSUFFICIENT_BUFFER` is returned. A buffer of `MAX_PATH` size always works.

`pPrinterProfile`

Points to a buffer in which to place the name of the user-selected printer profile. If the CMS\_SETPRINTERPROFILE flag is used, this flag can also be used to select a profile other than the printer default when the dialog is first displayed.

#### `ccPrinterProfile`

The size of the buffer pointed to by the **pPrinterProfile** member, in characters. If the buffer is not large enough to hold the selected name, the name is truncated to this size, and ERROR\_INSUFFICIENT\_BUFFER is returned. A buffer of MAX\_PATH size always works.

#### `pTargetProfile`

Points to a buffer in which to place the name of the user-selected target profile for proofing. If the CMS\_SETTARGETPROFILE flag is used, this flag can also be used to select a profile other than the printer default when the dialog is first displayed.

#### `ccTargetProfile`

The size of the buffer pointed to by the **pTargetProfile** member, in characters. If the buffer is not large enough to hold the selected name, the name is truncated to this size, and ERROR\_INSUFFICIENT\_BUFFER is returned. A buffer of MAX\_PATH size always works.

#### `lpfnHook`

If the CMS\_USEHOOK flag is set, this member is the address of a dialog procedure (see [DialogProc](#)) that can filter or handle messages for the dialog. The hook procedure receives no messages issued before WM\_INITDIALOG. It is called on the WM\_INITDIALOG message after the system-provided dialog procedure has processed the message. On all other messages, the hook procedure receives the message before the system-provided procedure. If the hook procedure returns **TRUE** to these messages, the system-provided procedure is not called.

The hook procedure may call the [EndDialog](#) function.

#### `lParam`

If the CMS\_USEHOOK flag is set, this member is passed to the application-provided hook procedure as the *lParam* parameter when the WM\_INITDIALOG message is processed.

#### `lpfnApplyCallback`

Contains a pointer to a callback function that is invoked when the **Apply** button of the Color Management dialog box is selected. If no callback function is provided, this member should be set to **NULL**. See [PCMSCALLBACKW](#).

## `IParamApplyCallback`

Contains a value that will be passed to the function `ApplyCallbackFunction` through its `IParam` parameter. The meaning and content of the value is specified by the application.

# Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h

## See also

- [A common dialog for color management](#)
- [DialogProc ↗](#)
- [EndDialog ↗](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# ColorProfileAddDisplayAssociation function (icm.h)

Associates an installed color profile with a specified display in the given scope.

## Syntax

C++

```
HRESULT ColorProfileAddDisplayAssociation(
    WCS_PROFILE_MANAGEMENT_SCOPE scope,
    PCWSTR                     profileName,
    LUID                        targetAdapterID,
    UINT32                      sourceID,
    BOOL                        setAsDefault,
    BOOL                        associateAsAdvancedColor
);
```

## Parameters

scope

Specifies the association as system-wide or the current user.

profileName

Identifies the installed profile to associate.

targetAdapterID

An identifier assigned to the adapter (e.g. GPU) of the target display. See [Remarks](#) for more details.

sourceID

An identifier assigned to the source of the display. See [Remarks](#) for more details.

setAsDefault

Whether or not to set the newly associated profile as the default.

associateAsAdvancedColor

Specifies to which association list the new profile is added.

# Return value

S\_OK for success, or a failure HRESULT value

## Remarks

See [connecting and configuring displays](#) for information on display adapter IDs and source IDs.

## Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 10 Build 20348
Minimum supported server	Windows 10 Build 20348
Header	icm.h
Library	Mscms.Lib

## See also

[Connecting and configuring displays](#)

---

Last updated on 10/31/2025

# ColorProfileGetDisplayDefault function (icm.h)

Gets the default color profile for a given display in the specified scope.

## Syntax

C++

```
HRESULT ColorProfileGetDisplayDefault(
    WCS_PROFILE_MANAGEMENT_SCOPE scope,
    LUID targetAdapterID,
    UINT32 sourceID,
    COLORPROFILETYPE profileType,
    COLORPROFILESUBTYPE profileSubType,
    LPWSTR *profileName
);
```

## Parameters

scope

Specifies the association as system-wide or the current user.

targetAdapterID

An identifier assigned to the adapter (e.g. GPU) of the target display. See [Remarks](#) for more details.

sourceID

An identifier assigned to the source of the display. See [Remarks](#) for more details.

profileType

The type of color profile to return (currently only CPT\_ICC is supported).

profileSubType

The subtype of the color profile to return.

profileName

Receives a pointer to the default color profile name, which must be freed with [LocalFree](#).

## Remarks

See [connecting and configuring displays](#) for information on display adapter IDs and source IDs.

## Requirements

 [Expand table](#)

Requirement	Value
Minimum supported client	Windows 10 Build 20348
Minimum supported server	Windows 10 Build 20348
Header	icm.h
Library	Mscms.Lib

## See also

[Connecting and configuring displays](#)

---

Last updated on 10/31/2025

# ColorProfileGetDisplayList function (icm.h)

Retrieves the list of profiles associated with a given display in the specified scope.

## Syntax

C++

```
HRESULT ColorProfileGetDisplayList(
    WCS_PROFILE_MANAGEMENT_SCOPE scope,
    LUID                         targetAdapterID,
    UINT32                        sourceID,
    LPWSTR                         **profileList,
    PDWORD                         profileCount
);
```

## Parameters

scope

Specifies the association as system-wide or the current user.

targetAdapterID

An identifier assigned to the adapter (e.g. GPU) of the target display. See [Remarks](#) for more details.

sourceID

An identifier assigned to the source of the display. See [Remarks](#) for more details.

profileList

Pointer to a buffer where the profile names are placed, must be freed with [LocalFree](#).

profileCount

Receives the number of profiles names copied into profileList.

## Return value

[S\\_OK](#) for success, or a failure [HRESULT](#) value

## Remarks

See [connecting and configuring displays](#) for information on display adapter IDs and source IDs.

## Requirements

 [Expand table](#)

Requirement	Value
Minimum supported client	Windows 10 Build 20348
Minimum supported server	Windows 10 Build 20348
Header	icm.h
Library	Mscms.Lib

## See also

[Connecting and configuring displays](#)

---

Last updated on 10/31/2025

# ColorProfileGetDisplayUserScope function (icm.h)

Gets the currently selected color profile scope of the provided display - either user or system.

## Syntax

C++

```
HRESULT ColorProfileGetDisplayUserScope(
    LUID targetAdapterID,
    UINT32 sourceID,
    WCS_PROFILE_MANAGEMENT_SCOPE *scope
);
```

## Parameters

`targetAdapterID`

An identifier assigned to the adapter (e.g. GPU) of the target display. See [Remarks](#) for more details.

`sourceID`

An identifier assigned to the source of the display. See [Remarks](#) for more details.

`scope`

Returns the scope of the currently selected color profile - either the current user or system.

## Return value

`S_OK` for success, or a failure `HRESULT` value

## Remarks

See [Connecting and configuring displays](#) for information on display adapter IDs and source IDs.

## Requirements

Requirement	Value
Minimum supported client	Windows 10 Build 20348
Minimum supported server	Windows 10 Build 20348
Header	icm.h
Library	Mscms.Lib

## See also

[Connecting and configuring displays](#)

---

Last updated on 10/31/2025

# ColorProfileRemoveDisplayAssociation function (icm.h)

Disassociates an installed color profile from a specified display in the given scope.

## Syntax

C++

```
HRESULT ColorProfileRemoveDisplayAssociation(
    WCS_PROFILE_MANAGEMENT_SCOPE scope,
    PCWSTR                 profileName,
    LUID                   targetAdapterID,
    UINT32                 sourceID,
    BOOL                   dissociateAdvancedColor
);
```

## Parameters

scope

Specifies the association as system-wide or the current user.

profileName

Identifies the installed profile to associate.

targetAdapterID

An identifier assigned to the adapter (e.g. GPU) of the target display. See [Remarks](#) for more details.

sourceID

An identifier assigned to the source of the display. See [Remarks](#) for more details.

dissociateAdvancedColor

Specifies to which association list the new profile is added.

## Return value

S\_OK for success, or a failure HRESULT value

## Remarks

See [connecting and configuring displays](#) for information on display adapter IDs and source IDs.

## Requirements

 [Expand table](#)

Requirement	Value
Minimum supported client	Windows 10 Build 20348
Minimum supported server	Windows 10 Build 20348
Header	icm.h
Library	Mscms.Lib

## See also

[Connecting and configuring displays](#)

---

Last updated on 10/31/2025

# ColorProfileSetDisplayDefaultAssociation function (icm.h)

Sets an installed color profile as the default profile for a specified display in the given scope.

## Syntax

C++

```
HRESULT ColorProfileSetDisplayDefaultAssociation(
    WCS_PROFILE_MANAGEMENT_SCOPE scope,
    PCWSTR                     profileName,
    COLORPROFILETYPE           profileType,
    COLORPROFILESUBTYPE         profileSubType,
    LUID                        targetAdapterID,
    UINT32                      sourceID
);
```

## Parameters

scope

Specifies the association as system-wide or the current user.

profileName

Identifies the installed profile to associate.

profileType

The type of color profile to set as default (currently only CPT\_ICC is supported).

profileSubType

The subtype of the color profile to set as default.

targetAdapterID

An identifier assigned to the adapter (e.g. GPU) of the target display. See [Remarks](#) for more details.

sourceID

An identifier assigned to the source of the display. See [Remarks](#) for more details.

# Return value

S\_OK for success, or a failure HRESULT value

## Remarks

See [connecting and configuring displays](#) for information on display adapter IDs and source IDs.

## Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 10 Build 20348
Minimum supported server	Windows 10 Build 20348
Header	icm.h
Library	Mscms.Lib

## See also

[Connecting and configuring displays](#)

---

Last updated on 10/31/2025

# COLORPROFILESUBTYPE enumeration (icm.h)

Article 02/22/2024

Specifies the subtype of the color profile.

## Syntax

C++

```
typedef enum {
    CPST_PERCEPTUAL,
    CPST_RELATIVE_COLORIMETRIC,
    CPST_SATURATION,
    CPST_ABSOLUTE_COLORIMETRIC,
    CPST_NONE,
    CPST_RGB_WORKING_SPACE,
    CPST_CUSTOM_WORKING_SPACE,
    CPST_STANDARD_DISPLAY_COLOR_MODE,
    CPST_EXTENDED_DISPLAY_COLOR_MODE
} COLORPROFILESUBTYPE;
```

## Constants

[+] Expand table

<b>CPST_PERCEPTUAL</b> A perceptual rendering intent for gamut map model profiles (GMMPs) defined in WCS.
<b>CPST_RELATIVE_COLORIMETRIC</b> A relative colorimetric rendering intent for GMMPs defined in WCS.
<b>CPST_SATURATION</b> A saturation rendering intent for GMMPs defined in WCS.
<b>CPST_ABSOLUTE_COLORIMETRIC</b> An absolute colorimetric rendering intent for GMMPs defined in WCS.
<b>CPST_NONE</b> The color profile subtype is not applicable to the selected color profile type.

<code>CPST_RGB_WORKING_SPACE</code>	The RGB color working space for International Color Consortium (ICC) profiles or device model profiles (DMPs) defined in WCS.
<code>CPST_CUSTOM_WORKING_SPACE</code>	A custom color working space.
<code>CPST_STANDARD_DISPLAY_COLOR_MODE</code>	TBD
<code>CPST_EXTENDED_DISPLAY_COLOR_MODE</code>	TBD

## Remarks

For a description of rendering intents, see [Rendering Intents](#).

The PCOLORPROFILESUBTYPE and LPCOLORPROFILESUBTYPE data types are defined as pointers to the **COLORPROFILESUBTYPE** enumeration:

```
typedef COLORPROFILESUBTYPE *PCOLORPROFILESUBTYPE, *LPCOLORPROFILESUBTYPE;
```

The valid profile type/subtype combinations are

\${ROWSpan3}\$ COLORPROFILETYPE  
 \${REMOVE}\$  
 Valid COLORPROFILESUBTYPE

Valid COLORPROFILESUBTYPE

\${ROWSpan3}\$ Notes  
 \${REMOVE}\$  
 Valid COLORPROFILESUBTYPE

default for a device

global default

Intended Usage

Intended Usage

CPT\_ICC

CPST\_NONE

Get/Set default ICC profile associated with a device

CPST\_RGBWorkingSpace or CPST\_CustomWorkingSpace

Get/Set ICC profile as global RGB or custom working space

CPT\_DMP

CPST\_NONE

Get/Set default DMP profile associated with a device

CPST\_RGBWorkingSpace or CPST\_CustomWorkingSpace

Get/Set DMP as global RGB or custom working space

CPT\_CAMP

CPST\_NONE

Get/Set default CAMP profile associated with a device

CPST\_NONE

Get/Set CAMP profile as global color appearance profile

CPT\_GMMP

CPST\_NONE

Get/Set default GMMP profile associated with a device

CPST\_Perceptual or

CPST\_Absolute\_colorimetric or

CPST\_Relative\_colorimetric or

CPTS\_Saturation

Get/Set GMMP as global gamut map model profile for a specific rendering intent as described by that subtype to be used in CreateMultiProfileTransform API when resolving the rendering intent array in WCS transform.

COLORPROFILESUBTYPE Global default can be or'd with WCS\_DEFAULT to set this GMMP as the global default for use in OpenColorProfile or WcsOpenColorProfile where GMMP is NULL.

## Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 10 Build 20348
Minimum supported server	Windows 10 Build 20348
Header	icm.h

## See also

- [COLORPROFILETYPE](#)
- [WcsSetDefaultColorProfile](#)

---

## Feedback

Was this page helpful?



[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# COLORPROFILETYPE enumeration (icm.h)

Article 02/22/2024

Specifies the type of color profile.

## Syntax

C++

```
typedef enum {
    CPT_ICC,
    CPT_DMP,
    CPT_CAMP,
    CPT_GMMP
} COLORPROFILETYPE;
```

## Constants

[ ] [Expand table](#)

**CPT\_ICC**

An International Color Consortium (ICC) profile. If you specify this value, only the CPST\_RGB\_WORKING\_SPACE and CPST\_CUSTOM\_WORKING\_SPACE values of [COLORPROFILESUBTYPE](#) are valid.

**CPT\_DMP**

A device model profile (DMP) defined in WCS. If you specify this value, only the CPST\_RGB\_WORKING\_SPACE and CPST\_CUSTOM\_WORKING\_SPACE values of [COLORPROFILESUBTYPE](#) are valid.

**CPT\_CAMP**

A color appearance model profile (CAMP) defined in WCS. If you specify this value, only the CPST\_NONE value of [COLORPROFILESUBTYPE](#) is valid.

**CPT\_GMMP**

Specifies a WCS gamut map model profile (GMMP). If this value is specified, only the CPST\_PERCEPTUAL, CPST\_SATURATION, CPST\_RELATIVE\_COLORIMETRIC, and CPST\_ABSOLUTE\_COLORIMETRIC values of [COLORPROFILESUBTYPE](#) are valid. Any of these values may optionally be combined (in a bitwise **OR** operation) with CPST\_DEFAULT.

# Remarks

The PCOLORPROFILETYPE and LPCOLORPROFILETYPE data types are defined as pointers to this enumeration:

```
typedef COLORPROFILETYPE *PCOLORPROFILETYPE, *LPCOLORPROFILETYPE;
```

# Requirements

[\[+\] Expand table](#)

Requirement	Value
Minimum supported client	Windows Vista [desktop apps only]
Minimum supported server	Windows Server 2008 [desktop apps only]
Header	icm.h

## See also

[COLORPROFILESUBTYPE](#)

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# COLORTYPE enumeration (icm.h)

Article 02/22/2024

The values of the **COLORTYPE** enumeration are used by several WCS functions. Variables of type **COLOR** are defined in the color spaces enumerated by the **COLORTYPE** enumeration.

## Syntax

C++

```
typedef enum {
    COLOR_GRAY = 1,
    COLOR_RGB,
    COLOR_XYZ,
    COLOR_Yxy,
    COLOR_Lab,
    COLOR_3_CHANNEL,
    COLOR_CMYK,
    COLOR_5_CHANNEL,
    COLOR_6_CHANNEL,
    COLOR_7_CHANNEL,
    COLOR_8_CHANNEL,
    COLOR_NAMED
} COLORTYPE;
```

## Constants

[ ] Expand table

**COLOR\_GRAY**

Value: 1

The **COLOR** is in the GRAYCOLOR color space.

**COLOR\_RGB**

The **COLOR** is in the RGBCOLOR color space.

**COLOR\_XYZ**

The **COLOR** is in the XYZCOLOR color space.

**COLOR\_Yxy**

The **COLOR** is in the YxyCOLOR color space.

#### COLOR\_Lab

The COLOR is in the LabCOLOR color space.

#### COLOR\_3\_CHANNEL

The COLOR is in the GENERIC3CHANNEL color space.

#### COLOR\_CMYK

The COLOR is in the CMYKCOLOR color space.

#### COLOR\_5\_CHANNEL

The COLOR is in a five channel color space.

#### COLOR\_6\_CHANNEL

The COLOR is in a six channel color space.

#### COLOR\_7\_CHANNEL

The COLOR is in a seven channel color space.

#### COLOR\_8\_CHANNEL

The COLOR is in an eight channel color space.

#### COLOR\_NAMED

The COLOR is in a named color space.

## Remarks

In addition to managing the common two, three, and four channel color spaces, WCS 1.0 is able to perform color management with device profiles that contain five through eight [color channels](#). It is also able to use named color spaces. When five, six, seven, or eight color channels are used, the provider of the device profile is free to determine what the color channels represent. The same is true of named color spaces. WCS 1.0 is able to manage these color spaces as long as there is a mapping in the device profile that maps the channels or the name space into the [PCS](#). The device profile must also contain a mapping from the PCS into the five, size, seven, or eight channel space, or into the named color space.

## Requirements

[\[+\] Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]

Requirement	Value
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# ConvertColorNameToIndex function (icm.h)

Article 02/22/2024

Converts color names in a named color space to index numbers in an International Color Consortium (ICC) color profile.

## Syntax

C++

```
BOOL ConvertColorNameToIndex(
    HPROFILE     hProfile,
    PCOLOR_NAME  paColorName,
    PDWORD       paIndex,
    DWORD        dwCount
);
```

## Parameters

`hProfile`

The handle to an ICC named color profile.

`paColorName`

Pointer to an array of color name structures.

`paIndex`

Pointer to an array of **DWORDs** that this function fills with the indices. The indices begin with one, not zero.

`dwCount`

The number of color names to convert.

## Return value

If this function succeeds with the conversion, the return value is **TRUE**.

If the conversion function fails, the return value is **FALSE**.

## Remarks

This function will fail if *hProfile* is not a valid ICC profile.

This function does not support Windows Color System (WCS) profiles CAMP, DMP, and GMMP; because named profiles are explicit ICC profile types.

## Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# ConvertIndexToColorName function (icm.h)

Article02/22/2024

Transforms indices in a color space to an array of names in a named color space.

## Syntax

C++

```
BOOL ConvertIndexToColorName(
    HPROFILE     hProfile,
    PDWORD       paIndex,
    PCOLOR_NAME  paColorName,
    DWORD        dwCount
);
```

## Parameters

`hProfile`

The handle to an International Color Consortium (ICC) color space profile.

`paIndex`

Pointer to an array of color-space index numbers. The indices begin with one, not zero.

`paColorName`

Pointer to an array of color name structures.

`dwCount`

The number of indices to convert.

## Return value

If this conversion function succeeds, the return value is **TRUE**.

If this conversion function fails, the return value is **FALSE**.

# Remarks

This function will fail if *hProfile* is not a valid ICC profile.

This function does not support Windows Color System (WCS) profiles CAMP, DMP, and GMMP; because named profiles are explicit ICC profile types.

# Requirements

[ ] [Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# CreateColorTransformA function (icm.h)

Article 08/23/2022

Creates a color transform that applications can use to perform color management.

## Syntax

C++

```
HTRANSFORM CreateColorTransformA(  
    LPLOGCOLORSPACEA pLogColorSpace,  
    HPROFILE          hDestProfile,  
    HPROFILE          hTargetProfile,  
    DWORD             dwFlags  
) ;
```

## Parameters

`pLogColorSpace`

Pointer to the input [LOGCOLORSPACEA](#).

`hDestProfile`

Handle to the profile of the destination device. The function determines whether the HPROFILE contains International Color Consortium (ICC) or Windows Color System (WCS) profile information.

`hTargetProfile`

Handle to the profile of the target device. The function determines whether the HPROFILE contains ICC or WCS profile information.

`dwFlags`

Specifies flags to used control creation of the transform. See Remarks.

## Return value

If this function succeeds, the return value is a handle to the color transform.

If this function fails, the return value is **NULL**. For extended error information, call **GetLastError**.

## Remarks

If the target profile is **NULL**, the transform goes from the source logical color space to the destination profile. If the target profile is given, the transform goes from the source logical color space to the target profile and then to the destination profile. This allows previewing output meant for the target device on the destination device.

The values in *dwFlags* are intended as hints only. The color management module must determine the best way to use them.

**Windows Vista:** Three new flags have been added that can be used with *dwFlags*:

Flag	Description
<b>PRESERVEBLACK</b>	If this bit is set, the transform engine inserts the appropriate black generation GMMP as the last GMMP in the transform sequence. This flag only works in a pure WCS transform.
<b>SEQUENTIAL_TRANSFORM</b>	If this bit is set, each step in the WCS processing pipeline is performed for every pixel in the image and no optimized color transform is built. This flag only works in a pure WCS transform. <b>Restrictions:</b> A transform created with the <b>SEQUENTIAL_TRANSFORM</b> flag set may only be used in the thread on which it was created and only for one color translation call at a time. COM must be initialized prior to creating the sequential transform and must remain initialized for the lifetime of the transform object.
<b>WCS_ALWAYS</b>	If this bit is set, even all-ICC transforms will use the WCS code path.

### ⓘ Note

**SEQUENTIAL\_TRANSFORM** was inadvertently omitted from the icm.h header in the Windows Vista SDK. If you wish to use the **SEQUENTIAL\_TRANSFORM** flag, define it in your application as follows:`#define SEQUENTIAL_TRANSFORM 0x80800000`

For details, see [CMM Transform Creation Flags](#). All of the flags mentioned there are supported for all types of transforms, except for **FAST\_TRANSLATE**, which only works in a pure ICC-to-ICC transform.

The `CreateColorTransform` function is used outside of a device context. Colors may shift when transforming from a color profile to the same color profile. This is due to precision errors. Therefore, a color transform should not be performed under these circumstances.

The B2Ax tags are required for any profile that is the target of a transform.

WCS transform support for ICC ColorSpace profiles is limited to RGB colorspace profiles. The following ICC profile types cannot be used in a CITE-processed transform, either a mixed WCS/ICC transform or an all-ICC transform with `WCS_ALWAYS` set:

- Non-RGB ColorSpace profiles
- NamedColor profiles
- n-channel profiles (where  $n > 8$ )
- DeviceLink profiles
- Abstract profiles

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# CreateColorTransformW function (icm.h)

Article 08/23/2022

Creates a color transform that applications can use to perform color management.

## Syntax

C++

```
HTRANSFORM CreateColorTransformW(
    LPLOGCOLORSPACEW pLogColorSpace,
    HPROFILE          hDestProfile,
    HPROFILE          hTargetProfile,
    DWORD             dwFlags
);
```

## Parameters

`pLogColorSpace`

Pointer to the input [LOGCOLORSPACEA](#).

`hDestProfile`

Handle to the profile of the destination device. The function determines whether the HPROFILE contains International Color Consortium (ICC) or Windows Color System (WCS) profile information.

`hTargetProfile`

Handle to the profile of the target device. The function determines whether the HPROFILE contains ICC or WCS profile information.

`dwFlags`

Specifies flags to used control creation of the transform. See Remarks.

## Return value

If this function succeeds, the return value is a handle to the color transform.

If this function fails, the return value is **NULL**. For extended error information, call **GetLastError**.

## Remarks

If the target profile is **NULL**, the transform goes from the source logical color space to the destination profile. If the target profile is given, the transform goes from the source logical color space to the target profile and then to the destination profile. This allows previewing output meant for the target device on the destination device.

The values in *dwFlags* are intended as hints only. The color management module must determine the best way to use them.

**Windows Vista:** Three new flags have been added that can be used with *dwFlags*:

Flag	Description
<b>PRESERVEBLACK</b>	If this bit is set, the transform engine inserts the appropriate black generation GMMP as the last GMMP in the transform sequence. This flag only works in a pure WCS transform.
<b>SEQUENTIAL_TRANSFORM</b>	If this bit is set, each step in the WCS processing pipeline is performed for every pixel in the image and no optimized color transform is built. This flag only works in a pure WCS transform. <b>Restrictions:</b> A transform created with the <b>SEQUENTIAL_TRANSFORM</b> flag set may only be used in the thread on which it was created and only for one color translation call at a time. COM must be initialized prior to creating the sequential transform and must remain initialized for the lifetime of the transform object.
<b>WCS_ALWAYS</b>	If this bit is set, even all-ICC transforms will use the WCS code path.

### ⓘ Note

**SEQUENTIAL\_TRANSFORM** was inadvertently omitted from the icm.h header in the Windows Vista SDK. If you wish to use the **SEQUENTIAL\_TRANSFORM** flag, define it in your application as follows:`#define SEQUENTIAL_TRANSFORM 0x80800000`

For details, see [CMM Transform Creation Flags](#). All of the flags mentioned there are supported for all types of transforms, except for **FAST\_TRANSLATE**, which only works in a pure ICC-to-ICC transform.

The `CreateColorTransform` function is used outside of a device context. Colors may shift when transforming from a color profile to the same color profile. This is due to precision errors. Therefore, a color transform should not be performed under these circumstances.

The B2Ax tags are required for any profile that is the target of a transform.

WCS transform support for ICC ColorSpace profiles is limited to RGB colorspace profiles. The following ICC profile types cannot be used in a CITE-processed transform, either a mixed WCS/ICC transform or an all-ICC transform with `WCS_ALWAYS` set:

- Non-RGB ColorSpace profiles
- NamedColor profiles
- n-channel profiles (where  $n > 8$ )
- DeviceLink profiles
- Abstract profiles

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# CreateDeviceLinkProfile function (icm.h)

Article 10/05/2021

Creates an International Color Consortium (ICC) *device link profile* from a set of color profiles, using the specified intents.

## Syntax

C++

```
BOOL CreateDeviceLinkProfile(
    PHPROFILE hProfile,
    DWORD     nProfiles,
    PDWORD    padwIntent,
    DWORD     nIntents,
    DWORD     dwFlags,
    PBYTE    *pProfileData,
    DWORD     indexPreferredCMM
);
```

## Parameters

`hProfile`

Pointer to an array of handles of the color profiles to be used. The function determines whether the HPROFILEs contain ICC profile information and, if so, it processes them appropriately.

`nProfiles`

Specifies the number of profiles in the array pointed to by *hProfile*.

`padwIntent`

Pointer to an array of **DWORDS** containing the intents to be used. See [Rendering intents](#).

`nIntents`

The number of intents in the array pointed to by *padwIntent*.

`dwFlags`

Specifies flags to used control creation of the transform. For details, see [CMM Transform Creation Flags](#).

`pProfileData`

Pointer to a pointer to a buffer. If successful, this function allocates the buffer, places its address in `*pProfileData`, and fills it with a device link profile. If the function succeeds, the calling application must free the buffer after it is no longer needed.

`indexPreferredCMM`

Specifies the one-based index of the color profile that indicates what color management module (CMM) to use. The application developer may allow Windows to choose the CMM by setting this parameter to INDEX\_DONT\_CARE. See [Using Color Management Modules \(CMM\)](#).

## Return value

If this function succeeds, the return value is a nonzero value.

If this function fails, the return value is zero. For extended error information, call `GetLastError`.

## Remarks

For HPROFILEs that contain WCS profile information, the HPROFILEs are converted into valid ICC profile handles and then these ICC profile handles are used in creating the device link profile.

The first and the last profiles in the array must be device profiles. The other profiles can be color space or abstract profiles.

Each profile's output color space must be the next profile's input color space.

The calling application must free the buffer allocated by this function and pointed to by the `pProfileData` parameter. The `GlobalFree` function should be used to free the buffer.

## Requirements

Minimum supported client

Windows 2000 Professional [desktop apps only]

Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [GlobalFree](#)

---

## Feedback

Was this page helpful?



[Get help at Microsoft Q&A](#)

# CreateMultiProfileTransform function (icm.h)

Article 08/23/2022

Accepts an array of profiles or a single [device link profile](#) and creates a color transform that applications can use to perform color mapping.

## Syntax

C++

```
HTRANSFORM CreateMultiProfileTransform(
    PHPROFILE pahProfiles,
    DWORD      nProfiles,
    PDWORD     padwIntent,
    DWORD      nIntents,
    DWORD      dwFlags,
    DWORD      indexPreferredCMM
);
```

## Parameters

**pahProfiles**

Pointer to an array of handles to the profiles to be used. The function determines whether the HPROFILEs contain International Color Consortium (ICC) or Windows Color System (WCS) profile information and processes them appropriately. When valid WCS profiles are returned by [OpenColorProfileW](#) and [WcsOpenColorProfileW](#), these profile handles contain the combination of DMP, CAMP, and GMMP profiles.

**nProfiles**

Specifies the number of profiles in the array. The maximum is 10.

**padwIntent**

Pointer to an array of intents to use. Each intent is one of the following values:

**INTENT\_PERCEPTUAL**

**INTENT\_SATURATION**

## `INTENT_RELATIVE_COLORIMETRIC`

## `INTENT_ABSOLUTE_COLORIMETRIC`

GMMPs are a generalization of intents. There are two possible sources of intents: the "destination" profile and the intent list parameter to [CreateMultiProfileTransform](#). The term "destination" is not used since all but two of the profiles in the profile list parameter will serve as first destination and then source.

For more information, see [Rendering Intents](#).

### `nIntents`

Specifies the number of elements in the intents array: can either be 1 or the same value as `nProfiles`. For profile arrays that contain any WCS profiles, the first rendering intent is ignored and only `nProfiles` - 1 elements are used for these profile arrays. The maximum number of `nIntents` is 10.

### `dwFlags`

Specifies flags used to control creation of the transform. See Remarks.

### `indexPreferredCMM`

Specifies the one-based index of the color profile that indicates what color management module (CMM) to use. The application developer may allow Windows to choose the CMM by setting this parameter to INDEX\_DONT\_CARE. See [Using Color Management Modules \(CMM\)](#) Third party CMMs are only available for ICC workflows. Profile arrays containing WCS profiles will ignore this flag. It is also ignored when only ICC profiles are used and when the WCS\_ALWAYS flag is used.

## Return value

If this function succeeds, the return value is a handle to the color transform.

If this function fails, the return value is `NULL`. For extended error information, call [GetLastError](#).

## Remarks

If a device link profile is being used, the function will fail if `nProfiles` is not set to 1.

The array of intents specifies how profiles should be combined. The `nth` intent is used for combining the `nth` profile in the array. If only one intent is specified, it is used for the

first profile, and all other profiles are combined using [Match intent](#).

The values in *dwFlags* are intended as hints only. The color management module must determine the best way to use them.

**Windows Vista:** Three new flags have been added that can be used with *dwFlags*:

Flag	Description
PRESERVEBLACK	If this bit is set, the transform engine inserts the appropriate black generation GMMP as the last GMMP in the transform sequence. This flag only works in a pure WCS transform.
SEQUENTIAL_TRANSFORM	If this bit is set, each step in the WCS processing pipeline is performed for every pixel in the image and no optimized color transform is built. This flag only works in a pure WCS transform. <b>Restrictions:</b> A transform created with the SEQUENTIAL_TRANSFORM flag set may only be used in the thread on which it was created and only for one color translation call at a time. COM must be initialized prior to creating the sequential transform and must remain initialized for the lifetime of the transform object.
WCS_ALWAYS	If this bit is set, even all-ICC transforms will use the WCS code path.

#### ① Note

SEQUENTIAL\_TRANSFORM was inadvertently omitted from the icm.h header in the Windows Vista SDK. If you wish to use the SEQUENTIAL\_TRANSFORM flag, define it in your application as follows:

```
#define SEQUENTIAL_TRANSFORM 0x80800000
```

For details, see [CMM Transform Creation Flags](#). All of the flags mentioned there are supported for all types of transforms, except for FAST\_TRANSLATE and USE\_RELATIVE\_COLORIMETRIC, which only work in a pure ICC-to-ICC transform.

The **CreateMultiProfileTransform** function is used outside of a device context. Colors may shift when transforming from a color profile to the same color profile. This is due to precision errors. Therefore, a color transform should not be performed under these circumstances.

We recommend that there be only one GMMP between a source and destination DMP. Gamut boundary descriptions (GBDs) are created from the DMP/CAMP combinations. The subsequent GMMPs use the GDBs prior to them in the processing chain until there

exists a DMP/CAMP GBD next in the sequence to be used. For example, assume a sequence DMP1, CAMP1, GMMP1, GMMP2, GMMP3, DMP2, CAMP2, GMMP4, GMMP5, CAMP3, DMP3. Then GMMP1, GMMP2 use GBD1 as their source and destination. Then GMMP3 uses GBD1 as source and GBD2 as destination. Then GMMP4 uses GBD2 as source and destination. Finally GMMP5 uses GBD2 as source and GBD3 as destination. This assumes no GMMP is identical to one next to it.

For WCS profiles, we recommend that the rendering intents be set to DWORD\_MAX in order to use the GMMP within the WCS profile handle. This is because the array of rendering intents takes precedence over the rendering intents or gamut mapping models specified or contained in the profiles specified by the HPROFILEs. The array of rendering intents references the default GMMP for those rendering intents. Ideally, only one gamut mapping is performed between a source and destination device by setting one or the other GMMP to **NULL** when creating the HPROFILE with WCS profile information. Any legacy application that uses a WCS DMP will invoke a sequence of GMMPs. GDBs are chosen based on DMPs and CAMPs. For intermediate GMMP gamut boundaries, the source and destination GBDs are used.

In summary, if *nIntents* == 1, then the first GMM is set based on the GMMP that is set as default\* for the *padwIntent* value, unless that value is DWORD\_MAX, in which case the embedded GMM information from the second profile is used (The embedded GMM information is either a GMMP or, in the case of an ICC profile, the baseline GMM corresponding to\*\* the intent from the profile header). The remainder of the GMMs are set based on the GMMP that is set as default\* for RelativeColorimetric.

If *nIntents* = *nProfiles* -1, then each GMM is set based on the GMMP that is set as default\* for the value in the *padwIntent* array at the corresponding index, except where *padwIntent* values are DWORD\_MAX. For values in the *padwIntent* array that are DWORD\_MAX, the GMMs at corresponding positions are set based on the embedded GMM information from the second of the two profiles whose gamuts are mapped by the GMM. (Again, the embedded GMM information is either a GMMP or, in the case of an ICC profile, the baseline GMM corresponding to\*\* the intent from the profile header).

If *nIntents* = *nProfiles*, then first intent is ignored and function behaves as it does in the case when *nIntents* = *nProfiles* -1.

Any other combination of *padwIntents* and *nIntents* will return an error.

\* "set as default" means that the default GMMP is queried using **WcsGetDefaultColorProfile** with its *profileManagementScope* parameter set to **WCS\_PROFILE\_MANAGEMENT\_SCOPE\_CURRENT\_USER**. This may return either current-user or system-wide defaults as described in the documentation for **WcsGetDefaultColorProfile**.

\*\* "GMM corresponding to" does not mean "GMM from the GMMP set as default for". Instead it means "a constant association between ICC profile intents and baseline GMM algorithms."

WCS transform support for ICC ColorSpace profiles is limited to RGB colorspace profiles. The following ICC profile types cannot be used in a CITE-processed transform, either a mixed WCS/ICC transform or an all-ICC transform with **WCS\_ALWAYS** set:

- Non-RGB ColorSpace profiles
- NamedColor profiles
- n-channel profiles (where  $n > 8$ )
- DeviceLink profiles
- Abstract profiles

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [COLOR Structure\\*\\*](#)
- [DeleteColorTransform](#)

## Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# CreateProfileFromLogColorSpaceA function (icm.h)

Article 02/22/2024

Converts a logical [color space](#) to a [device profile](#).

## Syntax

C++

```
BOOL CreateProfileFromLogColorSpaceA(
    LPLOGCOLORSPACEA pLogColorSpace,
    PBYTE             *pProfile
);
```

## Parameters

`pLogColorSpace`

A pointer to a logical color space structure. See [LOGCOLORSPACEA](#) for details. The `IcsFilename` [0] member of the structure must be set to the `null` character ('\0') or this function call will fail with the return value of `INVALID_PARAMETER`.

`pProfile`

A pointer to a pointer to a buffer where the device profile will be created. This function allocates the buffer and fills it with profile information if it is successful. If not, the pointer is set to `NULL`. The caller is responsible for freeing this buffer when it is no longer needed.

## Return value

If this function succeeds, the return value is `TRUE`.

If this function fails, the return value is `FALSE`.

If the `IcsFilename` [0] member if the [LOGCOLORSPACEA](#) structure pointed to by `pLogColorSpace` is not '\0', this function returns `INVALID_PARAMETER`.

## Remarks

This function can be used with ASCII or Unicode strings. The buffer created by this function must be freed by the caller when it is no longer needed or there will be a memory leak. The [GlobalFree](#) function should be used to free this buffer.

This function does not support Windows Color System (WCS) profiles CAMP, DMP, and GMMP.

## Requirements

  [Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [GlobalFree](#)

---

## Feedback

Was this page helpful?



Yes



No

[Provide product feedback](#) | [Get help at Microsoft Q&A](#)

# CreateProfileFromLogColorSpaceW function (icm.h)

Article 02/22/2024

Converts a logical [color space](#) to a [device profile](#).

## Syntax

C++

```
BOOL CreateProfileFromLogColorSpaceW(
    LPLOGCOLORSPACEW pLogColorSpace,
    PBYTE           *pProfile
);
```

## Parameters

`pLogColorSpace`

A pointer to a logical color space structure. See [LOGCOLORSPACEA](#) for details. The `IcsFilename` [0] member of the structure must be set to the `null` character ('\0') or this function call will fail with the return value of `INVALID_PARAMETER`.

`pProfile`

A pointer to a pointer to a buffer where the device profile will be created. This function allocates the buffer and fills it with profile information if it is successful. If not, the pointer is set to `NULL`. The caller is responsible for freeing this buffer when it is no longer needed.

## Return value

If this function succeeds, the return value is `TRUE`.

If this function fails, the return value is `FALSE`.

If the `IcsFilename` [0] member if the [LOGCOLORSPACEA](#) structure pointed to by `pLogColorSpace` is not '\0', this function returns `INVALID_PARAMETER`.

## Remarks

This function can be used with ASCII or Unicode strings. The buffer created by this function must be freed by the caller when it is no longer needed or there will be a memory leak. The [GlobalFree](#) function should be used to free this buffer.

This function does not support Windows Color System (WCS) profiles CAMP, DMP, and GMMP.

## Requirements

  [Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [GlobalFree](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) | [Get help at Microsoft Q&A](#)

# DeleteColorTransform function (icm.h)

Article02/22/2024

Deletes a given color transform.

## Syntax

C++

```
BOOL DeleteColorTransform(  
    HTRANSFORM hxform  
) ;
```

## Parameters

`hxform`

Identifies the color transform to delete.

## Return value

If this function succeeds, the return value is TRUE.

If this function fails, the return value is FALSE. For extended error information, call [GetLastError](#).

## Requirements

[Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
  - [Functions](#)
  - [COLOR structure](#)
  - [CreateMultiProfileTransform](#)
- 

## Feedback

Was this page helpful?



[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# DisassociateColorProfileFromDeviceA function (icm.h)

Article07/27/2022

Disassociates a specified color profile with a specified device on a specified computer.

## ⓘ Note

This API does not support "advanced color" profiles for HDR monitors. Use [ColorProfileRemoveDisplayAssociation](#) for managing advanced color profiles.

## Syntax

C++

```
BOOL DisassociateColorProfileFromDeviceA(  
    PCSTR pMachineName,  
    PCSTR pProfileName,  
    PCSTR pDeviceName  
) ;
```

## Parameters

pMachineName

Reserved. Must be **NULL**. This parameter is intended to point to the name of the computer on which to disassociate the specified profile and device. A **NULL** pointer indicates the local computer.

pProfileName

Pointer to the file name of the profile to disassociate.

pDeviceName

Pointer to the name of the device to disassociate.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call **GetLastError**.

## Remarks

If more than one profile is associated with a device, WCS uses the last one associated as the default. That is, if your application sequentially associates three profiles with a device, WCS will use the last one associated as the default. If your application then calls the **DisassociateColorProfileFromDevice** function to disassociate the third profile (which is the default in this example), the WCS will use the second profile as the default.

If your application disassociates all profiles from a device, WCS uses the sRGB profile as the default.

**DisassociateColorProfileFromDevice** always removes the specified profile from the current user's per-user profile association list for the specified device. Before removing the profile from the list, **DisassociateColorProfileFromDevice** determines whether the user has previously expressed the desire to use a per-user profile association list for the device. If so, then **DisassociateColorProfileFromDevice** simply removes the specified profile from the existing per-user profile association list for the device. If not, then **DisassociateColorProfileFromDevice** creates a new per-user profile association list for the device by copying the system-wide association list for that device. It then removes the specified profile from the per-user list. From that point on, the current user will be using a per-user profile association list for the specified device, as if **WcsSetUsePerUserProfiles** had been called for *pDevice* with the *usePerUserProfiles* parameter set to **TRUE**.

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- Basic color management concepts
  - Functions
  - AssociateColorProfileWithDeviceA
- 

## Feedback

Was this page helpful?



[Get help at Microsoft Q&A](#)

# DisassociateColorProfileFromDeviceW function (icm.h)

Article07/27/2022

Disassociates a specified color profile with a specified device on a specified computer.

## ⓘ Note

This API does not support "advanced color" profiles for HDR monitors. Use [ColorProfileRemoveDisplayAssociation](#) for managing advanced color profiles.

## Syntax

C++

```
BOOL DisassociateColorProfileFromDeviceW(
    PCWSTR pMachineName,
    PCWSTR pProfileName,
    PCWSTR pDeviceName
);
```

## Parameters

pMachineName

Reserved. Must be **NULL**. This parameter is intended to point to the name of the computer on which to disassociate the specified profile and device. A **NULL** pointer indicates the local computer.

pProfileName

Pointer to the file name of the profile to disassociate.

pDeviceName

Pointer to the name of the device to disassociate.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call **GetLastError**.

## Remarks

If more than one profile is associated with a device, WCS uses the last one associated as the default. That is, if your application sequentially associates three profiles with a device, WCS will use the last one associated as the default. If your application then calls the **DisassociateColorProfileFromDevice** function to disassociate the third profile (which is the default in this example), the WCS will use the second profile as the default.

If your application disassociates all profiles from a device, WCS uses the sRGB profile as the default.

**DisassociateColorProfileFromDevice** always removes the specified profile from the current user's per-user profile association list for the specified device. Before removing the profile from the list, **DisassociateColorProfileFromDevice** determines whether the user has previously expressed the desire to use a per-user profile association list for the device. If so, then **DisassociateColorProfileFromDevice** simply removes the specified profile from the existing per-user profile association list for the device. If not, then **DisassociateColorProfileFromDevice** creates a new per-user profile association list for the device by copying the system-wide association list for that device. It then removes the specified profile from the per-user list. From that point on, the current user will be using a per-user profile association list for the specified device, as if **WcsSetUsePerUserProfiles** had been called for *pDevice* with the *usePerUserProfiles* parameter set to **TRUE**.

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- Basic color management concepts
  - Functions
  - AssociateColorProfileWithDeviceW
- 

## Feedback

Was this page helpful?



[Get help at Microsoft Q&A](#)

# EnumColorProfilesA function (icm.h)

Enumerates all the profiles satisfying the given enumeration criteria.

## Syntax

C++

```
BOOL EnumColorProfilesA(
    PCSTR      pMachineName,
    PENUMTYPEA  pEnumRecord,
    PBYTE       pEnumerationBuffer,
    PDWORD     pdwSizeOfEnumerationBuffer,
    PDWORD     pnProfiles
);
```

## Parameters

`pMachineName`

Reserved. Must be **NULL**. This parameter is intended to point to the name of the computer on which to enumerate profiles. A **NULL** pointer indicates the local computer.

`pEnumRecord`

Pointer to a structure specifying the enumeration criteria.

`pEnumerationBuffer`

Pointer to a buffer in which the profiles are to be enumerated. A `MULTI_SZ` string of profile names satisfying the criteria specified in `*pEnumRecord` will be placed in this buffer.

`pdwSizeOfEnumerationBuffer`

Pointer to a variable containing the size of the buffer pointed to by `pBuffer`. On return, `*pdwSize` contains the size of buffer actually used or needed.

`pnProfiles`

Pointer to a variable that will contain, on return, the number of profile names actually copied to the buffer.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Remarks

Several profiles are typically associated with printers, based on the paper and ink types. There is a default profile for each device. For International Color Consortium (ICC) profiles, GDI selects the best one from the ICC-associated profiles when your application creates a device context (DC).

Do not attempt to use **EnumColorProfiles** to determine the default profile for a device. Instead, create a device context for the device and then invoke the [GetICMProfile](#) function. On Windows Vista and Windows 7, the [WcsGetDefaultColorProfile](#) function can also be used to determine a device's default color profile.

If the **dwFields** member of the structure of type **ENUMTYPE** that is pointed to by the *pEnumRecord* parameter is set to **ET\_DEVICENAME**, this function will enumerate all of the color profiles associated with all types of devices attached to the user's computer, regardless of the device class. If the **dwFields** member of the structure pointed to by the *pEnumRecord* parameter is set to **ET\_DEVICENAME** or **ET\_DEVICECLASS** and a device class is specified in the **dwDeviceClass** member of the structure, this function will only enumerate the profiles associated with the specified device class. If the **dwFields** member is set only to **ET\_DEVICECLASS**, the **EnumColorProfiles** function will enumerate all profiles that can be associated with that type of device.

Whenever **EnumColorProfiles** is examining the profiles associated with a specific device, the results depend on whether the user has chosen to use the system-wide list of profiles associated with that device, or his or her own ("per-user") list. Calling [WcsSetUsePerUserProfiles](#) with its *usePerUserProfiles* parameter set to **TRUE** causes future calls to **EnumColorProfiles** to look at only the current user's per-user list of profile associations for the specified device. Calling [WcsSetUsePerUserProfiles](#) with its *usePerUserProfiles* parameter set to **FALSE** causes future calls to **EnumColorProfiles** to look at the system-wide list of profile associations for the specified device. If [WcsSetUsePerUserProfiles](#) has never been called for the current user, **EnumColorProfiles** examines the system-wide list.

Your application can use **EnumColorProfiles** to obtain the size of the buffer in which the profiles are enumerated. It should call the **EnumColorProfiles** function with the *pBuffer* parameter set to **NULL**. When the function returns, the *pdwSize* parameter will contain the required buffer size in bytes. Your program can use that information to allocate the enumeration buffer. It can then invoke **EnumColorProfiles** again with the *pBuffer* parameter set to the address of the buffer.

This function will provide the information for converting WCS-specific DMP information to the legacy EnumType record in enable consistent profile enumeration. The defaults will be the same as ICC if this information is not present.

## Per-user/LUA support

The enumeration is specific to current user. Both system wide and current user device associations are considered. For default profile configuration, current user settings override system wide ones.

# Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.Lib

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [GetICMProfile](#)
- [ENUMTYPEW](#)

---

Last updated on 10/31/2025

# EnumColorProfilesW function (icm.h)

Enumerates all the profiles satisfying the given enumeration criteria.

## Syntax

C++

```
BOOL EnumColorProfilesW(
    PCWSTR      pMachineName,
    PENUMTYPEW   pEnumRecord,
    PBYTE       pEnumerationBuffer,
    PDWORD     pdwSizeOfEnumerationBuffer,
    PDWORD     pnProfiles
);
```

## Parameters

`pMachineName`

Reserved. Must be **NULL**. This parameter is intended to point to the name of the computer on which to enumerate profiles. A **NULL** pointer indicates the local computer.

`pEnumRecord`

Pointer to a structure specifying the enumeration criteria.

`pEnumerationBuffer`

Pointer to a buffer in which the profiles are to be enumerated. A **MULTI\_SZ** string of profile names satisfying the criteria specified in `*pEnumRecord` will be placed in this buffer.

`pdwSizeOfEnumerationBuffer`

Pointer to a variable containing the size of the buffer pointed to by `pBuffer`. On return, `*pdwSize` contains the size of buffer actually used or needed.

`pnProfiles`

Pointer to a variable that will contain, on return, the number of profile names actually copied to the buffer.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Remarks

Several profiles are typically associated with printers, based on the paper and ink types. There is a default profile for each device. For International Color Consortium (ICC) profiles, GDI selects the best one from the ICC-associated profiles when your application creates a device context (DC).

Do not attempt to use **EnumColorProfiles** to determine the default profile for a device. Instead, create a device context for the device and then invoke the [GetICMProfile](#) function. On Windows Vista and Windows 7, the [WcsGetDefaultColorProfile](#) function can also be used to determine a device's default color profile.

If the **dwFields** member of the structure of type **ENUMTYPE** that is pointed to by the *pEnumRecord* parameter is set to **ET\_DEVICENAME**, this function will enumerate all of the color profiles associated with all types of devices attached to the user's computer, regardless of the device class. If the **dwFields** member of the structure pointed to by the *pEnumRecord* parameter is set to **ET\_DEVICENAME** or **ET\_DEVICECLASS** and a device class is specified in the **dwDeviceClass** member of the structure, this function will only enumerate the profiles associated with the specified device class. If the **dwFields** member is set only to **ET\_DEVICECLASS**, the **EnumColorProfiles** function will enumerate all profiles that can be associated with that type of device.

Whenever **EnumColorProfiles** is examining the profiles associated with a specific device, the results depend on whether the user has chosen to use the system-wide list of profiles associated with that device, or his or her own ("per-user") list. Calling [WcsSetUsePerUserProfiles](#) with its *usePerUserProfiles* parameter set to **TRUE** causes future calls to **EnumColorProfiles** to look at only the current user's per-user list of profile associations for the specified device. Calling [WcsSetUsePerUserProfiles](#) with its *usePerUserProfiles* parameter set to **FALSE** causes future calls to **EnumColorProfiles** to look at the system-wide list of profile associations for the specified device. If [WcsSetUsePerUserProfiles](#) has never been called for the current user, **EnumColorProfiles** examines the system-wide list.

Your application can use **EnumColorProfiles** to obtain the size of the buffer in which the profiles are enumerated. It should call the **EnumColorProfiles** function with the *pBuffer* parameter set to **NULL**. When the function returns, the *pdwSize* parameter will contain the required buffer size in bytes. Your program can use that information to allocate the enumeration buffer. It can then invoke **EnumColorProfiles** again with the *pBuffer* parameter set to the address of the buffer.

This function will provide the information for converting WCS-specific DMP information to the legacy EnumType record in enable consistent profile enumeration. The defaults will be the same as ICC if this information is not present.

## Per-user/LUA support

The enumeration is specific to current user. Both system wide and current user device associations are considered. For default profile configuration, current user settings override system wide ones.

# Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.Lib

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [GetICMProfile](#)
- [ENUMTYPEW](#)

---

Last updated on 10/31/2025

# ENUMTYPEA structure (icm.h)

Article09/01/2022

Contains information that defines the profile enumeration constraints.

## Syntax

C++

```
typedef struct tagENUMTYPEA {
    DWORD dwSize;
    DWORD dwVersion;
    DWORD dwFields;
    PCSTR pDeviceName;
    DWORD dwMediaType;
    DWORD dwDitheringMode;
    DWORD dwResolution[2];
    DWORD dwCMMType;
    DWORD dwClass;
    DWORD dwDataColorSpace;
    DWORD dwConnectionSpace;
    DWORD dwSignature;
    DWORD dwPlatform;
    DWORD dwProfileFlags;
    DWORD dwManufacturer;
    DWORD dwModel;
    DWORD dwAttributes[2];
    DWORD dwRenderingIntent;
    DWORD dwCreator;
    DWORD dwDeviceClass;
} ENUMTYPEA, *PENUMTYPEA, *LPENUMTYPEA;
```

## Members

`dwSize`

The size of this structure in bytes.

`dwVersion`

The version number of the **ENUMTYPE** structure. Should be set to **ENUM\_TYPE\_VERSION**.

`dwFields`

Indicates which fields in this structure are being used. Can be set to any combination of the following constant values.

ET\_DEVICENAME

ET\_MEDIATYPE

ET\_DITHERMODE

ET\_RESOLUTION

ET\_CMMTYPE

ET\_CLASS

ET\_DATACOLORSPACE

ET\_CONNECTIONSPACE

ET\_SIGNATURE

ET\_PLATFORM

ET\_PROFILEFLAGS

ET\_MANUFACTURER

ET\_MODEL

ET\_ATTRIBUTES

ET\_RENDERINGINTENT

ET\_CREATOR

ET\_DEVICECLASS

pDeviceName

User friendly name of the device.

dwMediaType

Indicates which type of media is associated with the profile, such as a printer or screen.

dwDitheringMode

Indicates the style of dithering that will be used when an image is displayed.

**dwResolution[2]**

The horizontal (x) and vertical (y) resolution in pixels of the device on which the image will be displayed. The x resolution is stored in **dwResolution[0]**, and the y resolution is kept in **dwResolution[1]**.

**dwCMMType**

The identification number of the CMM that is used in the profile. Identification numbers are registered with the ICC.

**dwClass**

Indicates the profile class. For a description of profile classes, see [Using Device Profiles with WCS](#). A profile class may have any of the following values.

<b>Profile Class</b>	<b>Signature</b>
Input Device Profile	CLASS_SCANNER
Display Device Profile	CLASS_MONITOR
Output Device Profile	CLASS_PRINTER
Device Link Profile	CLASS_LINK
Color Space Conversion Profile	CLASS_COLORSPACE
Abstract Profile	CLASS_ABSTRACT
Named Color Profile	CLASS_NAMED
Color Appearance Model Profile	CLASS_CAMP
Color Gamut Map Model Profile	CLASS_GMMP

**dwDataColorSpace**

A signature value that indicates the color space in which the profile data is defined. Can be any value from the [Color Space Constants](#).

**dwConnectionSpace**

A signature value that indicates the color space in which the profile connection space (PCS) is defined. Can be any of the following values.

<b>Profile Class</b>	<b>Signature</b>

<b>Profile Class</b>	<b>Signature</b>
XYZ	SPACE_XYZ
Lab	SPACE_Lab

When the **dwClass** member is set to CLASS\_LINK, the PCS is taken from the **dwDataColorSpace** member.

#### **dwSignature**

Reserved for internal use.

#### **dwPlatform**

The primary platform for which the profile was created. The member can be set to any of the following values.

<b>Platform</b>	<b>Value</b>
Apple Computer, Inc.	'APPL'
Microsoft Corp.	'MSFT'
Silicon Graphics, Inc.	'SGI'
Sun Microsystems, Inc.	'SUNW'
Taligent	'TGNT'

#### **dwProfileFlags**

Bit flags containing hints that the CMM uses to interpret the profile data and can be set to one of the following values.

<b>Constant</b>	<b>Meaning</b>
FLAG_EMBEDDEDPROFILE	The profile is embedded in a bitmap file.
FLAG_DEPENDENTONDATA	The profile can't be used independently of the embedded color data. Used for profiles that are embedded in bitmap files.

#### **dwManufacturer**

The identification number of the device profile manufacturer. All manufacturer identification numbers are registered with the ICC.

#### **dwModel**

The device manufacturer's device model number. All model identification numbers are registered with the ICC.

`dwAttributes[2]`

Attributes of profile that can be any of the following values.

<b>Constant</b>	<b>Meaning</b>
ATTRIB_TRANSPARENCY	Turns transparency on. If this flag is not used, the attribute is reflective by default.
ATTRIB_MATTE	Turns matte display on. If this flag is not used, the attribute is glossy by default.

`dwRenderingIntent`

The profile rendering intent that can be set to one of the following values:

`INTENT_PERCEPTUAL`

`INTENT_SATURATION`

`INTENT_RELATIVE_COLORIMETRIC`

`INTENT_ABSOLUTE_COLORIMETRIC`

For more information, see [Rendering intents](#).

`dwCreator`

Signature of the software that created the profile. Signatures are registered with the ICC.

`dwDeviceClass`

Indicates the device class. A device class may have one of the following values.

<b>Profile Class</b>	<b>Signature</b>
Input Device Profile	CLASS_SCANNER
Display Device Profile	CLASS_MONITOR
Output Device Profile	CLASS_PRINTER

## Requirements

<b>Minimum supported client</b>	Windows 2000 Professional [desktop apps only]
<b>Minimum supported server</b>	Windows 2000 Server [desktop apps only]
<b>Header</b>	icm.h

## See also

- [Further information](#)
- [Using device profiles with WCS](#)
- [Rendering intents](#)

---

## Feedback

Was this page helpful?



[Get help at Microsoft Q&A](#)

# ENUMTYPEW structure (icm.h)

Article09/01/2022

Contains information that defines the profile enumeration constraints.

## Syntax

C++

```
typedef struct tagENUMTYPEW {
    DWORD  dwSize;
    DWORD  dwVersion;
    DWORD  dwFields;
    PCWSTR pDeviceName;
    DWORD  dwMediaType;
    DWORD  dwDitheringMode;
    DWORD  dwResolution[2];
    DWORD  dwCMMType;
    DWORD  dwClass;
    DWORD  dwDataColorSpace;
    DWORD  dwConnectionSpace;
    DWORD  dwSignature;
    DWORD  dwPlatform;
    DWORD  dwProfileFlags;
    DWORD  dwManufacturer;
    DWORD  dwModel;
    DWORD  dwAttributes[2];
    DWORD  dwRenderingIntent;
    DWORD  dwCreator;
    DWORD  dwDeviceClass;
} ENUMTYPEW, *PENUMTYPEW, *LPENUMTYPEW;
```

## Members

`dwSize`

The size of this structure in bytes.

`dwVersion`

The version number of the **ENUMTYPE** structure. Should be set to **ENUM\_TYPE\_VERSION**.

`dwFields`

Indicates which fields in this structure are being used. Can be set to any combination of the following constant values.

ET\_DEVICENAME

ET\_MEDIATYPE

ET\_DITHERMODE

ET\_RESOLUTION

ET\_CMMTYPE

ET\_CLASS

ET\_DATACOLORSPACE

ET\_CONNECTIONSPACE

ET\_SIGNATURE

ET\_PLATFORM

ET\_PROFILEFLAGS

ET\_MANUFACTURER

ET\_MODEL

ET\_ATTRIBUTES

ET\_RENDERINGINTENT

ET\_CREATOR

ET\_DEVICECLASS

pDeviceName

User friendly name of the device.

dwMediaType

Indicates which type of media is associated with the profile, such as a printer or screen.

dwDitheringMode

Indicates the style of dithering that will be used when an image is displayed.

### **dwResolution[2]**

The horizontal (x) and vertical (y) resolution in pixels of the device on which the image will be displayed. The x resolution is stored in **dwResolution[0]**, and the y resolution is kept in **dwResolution[1]**.

### **dwCMMType**

The identification number of the CMM that is used in the profile. Identification numbers are registered with the ICC.

### **dwClass**

Indicates the profile class. For a description of profile classes, see [Using Device Profiles with WCS](#). A profile class may have any of the following values.

<b>Profile Class</b>	<b>Signature</b>
Input Device Profile	CLASS_SCANNER
Display Device Profile	CLASS_MONITOR
Output Device Profile	CLASS_PRINTER
Device Link Profile	CLASS_LINK
Color Space Conversion Profile	CLASS_COLORSPACE
Abstract Profile	CLASS_ABSTRACT
Named Color Profile	CLASS_NAMED
Color Appearance Model Profile	CLASS_CAMP
Color Gamut Map Model Profile	CLASS_GMMP

### **dwDataColorSpace**

A signature value that indicates the color space in which the profile data is defined. Can be any value from the [Color Space Constants](#).

### **dwConnectionSpace**

A signature value that indicates the color space in which the profile connection space (PCS) is defined. Can be any of the following values.

<b>Profile Class</b>	<b>Signature</b>
----------------------	------------------

<b>Profile Class</b>	<b>Signature</b>
XYZ	SPACE_XYZ
Lab	SPACE_Lab

When the dwClass member is set to CLASS\_LINK, the PCS is taken from the dwDataColorSpace member.

#### `dwSignature`

Reserved for internal use.

#### `dwPlatform`

The primary platform for which the profile was created. The member can be set to any of the following values.

<b>Platform</b>	<b>Value</b>
Apple Computer, Inc.	'APPL'
Microsoft Corp.	'MSFT'
Silicon Graphics, Inc.	'SGI'
Sun Microsystems, Inc.	'SUNW'
Taligent	'TGNT'

#### `dwProfileFlags`

Bit flags containing hints that the CMM uses to interpret the profile data and can be set to one of the following values.

<b>Constant</b>	<b>Meaning</b>
FLAG_EMBEDDEDPROFILE	The profile is embedded in a bitmap file.
FLAG_DEPENDENTONDATA	The profile can't be used independently of the embedded color data. Used for profiles that are embedded in bitmap files.

#### `dwManufacturer`

The identification number of the device profile manufacturer. All manufacturer identification numbers are registered with the ICC.

#### `dwModel`

The device manufacturer's device model number. All model identification numbers are registered with the ICC.

`dwAttributes[2]`

Attributes of profile that can be any of the following values.

<b>Constant</b>	<b>Meaning</b>
ATTRIB_TRANSPARENCY	Turns transparency on. If this flag is not used, the attribute is reflective by default.
ATTRIB_MATTE	Turns matte display on. If this flag is not used, the attribute is glossy by default.

`dwRenderingIntent`

The profile rendering intent that can be set to one of the following values:

`INTENT_PERCEPTUAL`

`INTENT_SATURATION`

`INTENT_RELATIVE_COLORIMETRIC`

`INTENT_ABSOLUTE_COLORIMETRIC`

For more information, see [Rendering intents](#).

`dwCreator`

Signature of the software that created the profile. Signatures are registered with the ICC.

`dwDeviceClass`

Indicates the device class. A device class may have one of the following values.

<b>Profile Class</b>	<b>Signature</b>
Input Device Profile	CLASS_SCANNER
Display Device Profile	CLASS_MONITOR
Output Device Profile	CLASS_PRINTER

## Requirements

<b>Minimum supported client</b>	Windows 2000 Professional [desktop apps only]
<b>Minimum supported server</b>	Windows 2000 Server [desktop apps only]
<b>Header</b>	icm.h

## See also

- [Further information](#)
- [Using device profiles with WCS](#)
- [Rendering intents](#)

---

## Feedback

Was this page helpful?



[Get help at Microsoft Q&A](#)

# GENERIC3CHANNEL structure (icm.h)

Article02/22/2024

TBD

## Syntax

C++

```
struct GENERIC3CHANNEL {
    WORD ch1;
    WORD ch2;
    WORD ch3;
};
```

## Members

ch1

TBD

ch2

TBD

ch3

TBD

## Requirements

[+] Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h

# Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# GetCMMInfo function (icm.h)

Article02/22/2024

Retrieves various information about the color management module (CMM) that created the specified color transform.

## Syntax

C++

```
DWORD GetCMMInfo(
    HTRANSFORM hColorTransform,
    DWORD       unnamedParam2
);
```

## Parameters

`hColorTransform`

Identifies the transform for which to find CMM information.

`unnamedParam2`

Specifies the information to be retrieved. This parameter can take one of the following constant values.

[ ] Expand table

Value	Meaning
<code>CMM_WIN_VERSION</code>	Retrieves the version of Windows targeted by the color management module (CMM).
<code>CMM_DLL_VERSION</code>	Retrieves the version number of the CMM.
<code>CMM_IDENT</code>	Retrieves the CMM signature registered with the International Color Consortium (ICC).

## Return value

If this function succeeds, the return value is the information specified in `dwInfo`.

If this function fails, the return value is zero.

# Requirements

[\[+\] Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# GetColorDirectoryA function (icm.h)

Article02/22/2024

## ⓘ Note

This API may be unavailable in future releases. We encourage new and existing software to use other APIs for color profile interactions. Please refer to the below table for some examples.

  Expand table

Scenario	Mechanism
Enumerating all installed profiles	Use <a href="#">WcsEnumColorProfilesSize</a> and <a href="#">WcsEnumColorProfiles</a> , or <a href="#">EnumColorProfilesA</a>
Installing/Uninstalling color profiles	Use <a href="#">InstallColorProfileA</a> / <a href="#">UninstallColorProfileA</a>
Opening a color profile file directly	Use <a href="#">OpenColorProfileA</a> with dwType=PROFILE_FILENAME in the PROFILE struct parameter. Or use <a href="#">WcsOpenColorProfileA</a> . <a href="#">Icm.h</a> contains many APIs that accept the returned HPROFILE for color profile manipulation

Retrieves the path of the Windows COLOR directory on a specified machine.

## Syntax

C++

```
BOOL GetColorDirectoryA(
    PCSTR pMachineName,
    PSTR pBuffer,
    PDWORD pdwSize
);
```

## Parameters

pMachineName

Reserved; must be **NULL**. This parameter is intended to point to the name of the machine on which the profile is to be installed. A **NULL** pointer indicates the local machine.

`pBuffer`

Points to the buffer in which the color directory path is to be placed.

`pdwSize`

Points to a variable containing the size in bytes of the buffer pointed to by *pBuffer*. On return, the variable contains the size of the buffer actually used or needed.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call **GetLastError**.

## Remarks

Per-user/LUA support

Color directory is still system-wide. This function is executable in LUA context.

## Requirements

[ ] [Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- Basic color management concepts
  - Functions
- 

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# GetColorDirectoryW function (icm.h)

Article02/22/2024

## ⓘ Note

This API may be unavailable in future releases. We encourage new and existing software to use other APIs for color profile interactions. Please refer to the below table for some examples.

[+] Expand table

Scenario	Mechanism
Enumerating all installed profiles	Use <a href="#">WcsEnumColorProfilesSize</a> and <a href="#">WcsEnumColorProfiles</a> , or <a href="#">EnumColorProfilesW</a>
Installing/Uninstalling color profiles	Use <a href="#">InstallColorProfileW</a> / <a href="#">UninstallColorProfileW</a>
Opening a color profile file directly	Use <a href="#">OpenColorProfileW</a> with dwType=PROFILE_FILENAME in the PROFILE struct parameter. Or use <a href="#">WcsOpenColorProfileW</a> . <a href="#">Icm.h</a> contains many APIs that accept the returned HPROFILE for color profile manipulation

Retrieves the path of the Windows COLOR directory on a specified machine.

## Syntax

C++

```
BOOL GetColorDirectoryW(
    PCWSTR pMachineName,
    PWSTR pBuffer,
    PDWORD pdwSize
);
```

## Parameters

pMachineName

Reserved; must be **NULL**. This parameter is intended to point to the name of the machine on which the profile is to be installed. A **NULL** pointer indicates the local machine.

`pBuffer`

Points to the buffer in which the color directory path is to be placed.

`pdwSize`

Points to a variable containing the size in bytes of the buffer pointed to by *pBuffer*. On return, the variable contains the size of the buffer actually used or needed.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call **GetLastError**.

## Remarks

Per-user/LUA support

Color directory is still system-wide. This function is executable in LUA context.

## Requirements

[ ] Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- Basic color management concepts
  - Functions
- 

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# GetColorProfileElement function (icm.h)

Article 10/05/2021

Copies data from a specified tagged profile element of a specified color profile into a buffer.

## Syntax

C++

```
BOOL GetColorProfileElement(
    HPROFILE hProfile,
    TAGTYPE tag,
    DWORD dwOffset,
    PDWORD pcbElement,
    PVOID pElement,
    PBOOL pbReference
);
```

## Parameters

`hProfile`

Specifies a handle to the International Color Consortium (ICC) color profile in question.

`tag`

Identifies the tagged element from which to copy.

`dwOffset`

Specifies the offset from the first byte of the tagged element data at which to begin copying.

`pcbElement`

Pointer to a variable specifying the number of bytes to copy. On return, the variable contains the number of bytes actually copied.

`pElement`

Pointer to a buffer into which the tagged element data is to be copied. The buffer must contain at least as many bytes as are specified by the variable pointed to by `pcbSize`. If

If the *pBuffer* pointer is set to **NULL**, the size of the entire tagged element data in bytes is returned in the memory location pointed to by *pcbSize*, and *dwOffset* is ignored. In this case, the function will return **FALSE**.

#### pbReference

Points to a Boolean value that is set to **TRUE** if more than one tag in the color profile refers to the same data as the specified tag refers to, or **FALSE** if not.

## Return value

If this function succeeds, the return value is nonzero.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Remarks

This function will fail if *hProfile* is not a valid International Color Consortium (ICC) profile.

If the *pBuffer* pointer is set to **NULL**, the size of the entire tagged element data in bytes is returned in the variable pointed to by *pcbSize*, and *dwOffset* is ignored.

This function does not support Windows Color System (WCS) profiles CAMP, DMP, and GMMP; because profile elements are implicitly associated with, and hard coded to, ICC tag types and there exist many robust XML parsing libraries.

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- Basic color management concepts
  - Functions
- 

## Feedback

Was this page helpful?



[Get help at Microsoft Q&A](#)

# GetColorProfileElementTag function (icm.h)

Article02/22/2024

Retrieves the tag name specified by *dwIndex* in the tag table of a given International Color Consortium (ICC) color profile, where *dwIndex* is a one-based index into that table.

## Syntax

C++

```
BOOL GetColorProfileElementTag(  
    HPROFILE hProfile,  
    DWORD     dwIndex,  
    PTAGTYPE  pTag  
) ;
```

## Parameters

`hProfile`

Specifies a handle to the ICC color profile in question.

`dwIndex`

Specifies the one-based index of the tag to retrieve.

`pTag`

Pointer to a variable in which the tag name is to be placed.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call **GetLastError**.

## Remarks

This function will fail if *hProfile* is not a valid ICC profile.

**GetColorProfileElementTag** can be used to enumerate all tags in a profile after getting the number of tags in the profile using [GetCountColorProfileElements](#).

This function does not support Windows Color System (WCS) profiles CAMP, DMP, and GMMP; because profile elements are implicitly associated with, and hard coded to, ICC tag types and there exist many robust XML parsing libraries.

## Requirements

[] [Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# GetColorProfileFromHandle function (icm.h)

Article02/22/2024

Given a handle to an open color profile, the **GetColorProfileFromHandle** function copies the contents of the profile into a buffer supplied by the application. If the handle is a Windows Color System (WCS) handle, then the DMP is returned and the CAMP and GMMP associated with the HPROFILE are ignored.

## Syntax

C++

```
BOOL GetColorProfileFromHandle(
    HPROFILE hProfile,
    PBYTE    pProfile,
    PDWORD   pcbProfile
);
```

## Parameters

`hProfile`

Handle to an open color profile. The function determines whether the HPROFILE contains ICC or WCS profile information.

`pProfile`

Pointer to buffer to receive raw ICC or DMP profile data. Can be **NULL**. If it is, the size required for the buffer will be stored in the memory location pointed to by *pcbSize*. The buffer can be allocated to the appropriate size, and this function called again with *pBuffer* containing the address of the buffer.

`pcbProfile`

Pointer to a **DWORD** that holds the size of buffer pointed at by *pBuffer*. On return it is filled with size of buffer that was actually used if the function succeeds. If this function is called with *pBuffer* set to **NULL**, this parameter will contain the size of the buffer required.

# Return value

If this function succeeds, the return value is **TRUE**. It returns **FALSE** if the *pBuffer* parameter is **NULL** and the size required for the buffer is copied into *pcbSize*.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Requirements

[ ] [Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# GetColorProfileHeader function (icm.h)

Article02/22/2024

Retrieves or derives ICC header structure from either ICC color profile or WCS XML profile. Drivers and applications should assume returning **TRUE** only indicates that a properly structured header is returned. Each tag will still need to be validated independently using either legacy ICM2 APIs or XML schema APIs.

## Syntax

C++

```
BOOL GetColorProfileHeader(
    HPROFILE      hProfile,
    PPROFILEHEADER pHeader
);
```

## Parameters

`hProfile`

Specifies a handle to the color profile in question.

`pHeader`

Points to a variable in which the ICC header structure is to be placed.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. This function will fail if an invalid ICC or WCS XML profile is referenced in the `hProfile` parameter. For extended error information, call `GetLastError`.

## Remarks

To determine whether the header is derived from an ICC or DMP profile handle, check the header signature (header bytes 36-39). If the signature is "acsp" (big endian) then an ICC profile was used. If the signature is "cdmp" (big-endian) then a DMP was used.

The distinguishing features that identify a header as having been "synthesized" for a WCS DMP are:

plcmProfileHeader->phSignature = 'pmdc' (little endian = big endian 'cdmp')

plcmProfileHeader->phCMMType = '1scw' (little endian = big endian 'wcs1').

## Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [PROFILEHEADER](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# GetCountColorProfileElements function (icm.h)

Article 02/22/2024

Retrieves the number of tagged elements in a given color profile.

## Syntax

C++

```
BOOL GetCountColorProfileElements(
    HPROFILE hProfile,
    PDWORD    pnElementCount
);
```

## Parameters

`hProfile`

Specifies a handle to the profile in question.

`pnElementCount`

Pointer to a variable in which to place the number of tagged elements in the profile.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Remarks

This function will fail if *hProfile* is not a valid ICC profile.

This function does not support Windows Color System (WCS) profiles CAMP, DMP, and GMMP.

# Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) | [Get help at Microsoft Q&A](#)

# GetNamedProfileInfo function (icm.h)

Article02/22/2024

Retrieves information about the International Color Consortium (ICC) named color profile that is specified in the first parameter.

## Syntax

C++

```
BOOL GetNamedProfileInfo(  
    HPROFILE           hProfile,  
    PNAMED_PROFILE_INFO pNamedProfileInfo  
) ;
```

## Parameters

`hProfile`

The handle to the ICC profile from which the information will be retrieved.

`pNamedProfileInfo`

A pointer to a `NAMED_PROFILE_INFO` structure.

## Return value

If this function succeeds, the return value is `TRUE`.

If this function fails, the return value is `FALSE`.

## Remarks

This function will fail if `hProfile` is not a valid ICC profile.

This function does not support Windows Color System (WCS) profiles CAMP, DMP, and GMMP; because named profiles are explicit ICC profile types.

## Requirements

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Gdi32.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [NAMED\\_PROFILE\\_INFO](#)

---

## Feedback

Was this page helpful?



[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# GetPS2ColorRenderingDictionary function (icm.h)

Article 02/22/2024

Retrieves the PostScript Level 2 color rendering dictionary from the specified ICC color profile.

## Syntax

C++

```
BOOL GetPS2ColorRenderingDictionary(
    HPROFILE hProfile,
    DWORD     dwIntent,
    PBYTE     pPS2ColorRenderingDictionary,
    PDWORD    pcbPS2ColorRenderingDictionary,
    PBOOL     pbBinary
);
```

## Parameters

`hProfile`

Specifies a handle to the ICC color profile in question.

`dwIntent`

Specifies the desired rendering intent for the color rendering dictionary. Valid values are:

- INTENT\_PERCEPTUAL
- INTENT\_SATURATION
- INTENT\_RELATIVE\_COLORIMETRIC
- INTENT\_ABSOLUTE\_COLORIMETRIC

For more information, see [Rendering intents](#).

`pPS2ColorRenderingDictionary`

Pointer to a buffer in which the color rendering dictionary is to be placed. If the `pBuffer` pointer is set to **NULL**, the required buffer size is returned in `*pcbSize`.

`pcbPS2ColorRenderingDictionary`

Pointer to a variable containing the size of the buffer in bytes. On return, the variable contains the number of bytes actually copied.

#### pbBinary

Pointer to a Boolean variable. If **TRUE**, the color rendering dictionary could be copied in binary form. If **FALSE**, the dictionary will be encoded in ASCII85 form. On return, this Boolean variable indicates whether the dictionary was actually binary (**TRUE**) or ASCII85 (**FALSE**).

## Return value

If this function succeeds, the return value is **TRUE**. It also returns **TRUE** if the *pBuffer* parameter is **NULL** and the size required for the buffer is copied into *pcbSize*.

If this function fails, the return value is **FALSE**.

## Remarks

If the dictionary is not available in the profile, the **GetPS2ColorRenderingDictionary** function builds one using the profile contents. This dictionary can then be used as the operand for the PostScript Level 2 **setcolorrendering** operator.

This method does not support WCS profiles.

## Requirements

[+] Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- Basic color management concepts
  - Functions
- 

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# GetPS2ColorRenderingIntent function (icm.h)

Article02/22/2024

Retrieves the PostScript Level 2 color [rendering intent](#) from an ICC color profile.

## Syntax

C++

```
BOOL GetPS2ColorRenderingIntent(
    HPROFILE hProfile,
    DWORD     dwIntent,
    PBYTE     pBuffer,
    PDWORD    pcbPS2ColorRenderingIntent
);
```

## Parameters

`hProfile`

Specifies a handle to the ICC color profile in question.

`dwIntent`

Specifies the desired rendering intent to retrieve. Valid values are:

`INTENT_PERCEPTUAL`

`INTENT_SATURATION`

`INTENT_RELATIVE_COLORIMETRIC`

`INTENT_ABSOLUTE_COLORIMETRIC`

For more information, see [Rendering Intents](#).

`pBuffer`

Points to a buffer in which the color rendering intent is to be placed. If the `pBuffer` pointer is set to `NULL`, the buffer size required is returned in `*pcbSize`.

`pcbPS2ColorRenderingIntent`

Points to a variable containing the buffer size in bytes. On return, this variable contains the number of bytes actually copied.

## Return value

If this function succeeds, the return value is **TRUE**. If this function succeeds, the return value is **TRUE**. It also returns **TRUE** if the *pBuffer* parameter is **NULL** and the size required for the buffer is copied into *pcbSize*.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Remarks

The rendering intent returned by [GetPS2ColorRenderingIntent](#) can be used as the operand for the PostScript Level 2 *findcolorrendering* operator.

This method does not support WCS profiles.

## Requirements

[Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | Get help at Microsoft Q&A

# GetPS2ColorSpaceArray function (icm.h)

Article 10/05/2021

Retrieves the PostScript Level 2 [color space](#) array from an ICC color profile.

## Syntax

C++

```
BOOL GetPS2ColorSpaceArray(
    HPROFILE hProfile,
    DWORD     dwIntent,
    DWORD     dwCSAType,
    PBYTE     pPS2ColorSpaceArray,
    PDWORD    pcbPS2ColorSpaceArray,
    PBOOL     pbBinary
);
```

## Parameters

`hProfile`

Specifies a handle to the ICC profile from which to retrieve the PostScript Level 2 color space array.

`dwIntent`

Specifies the desired rendering intent for the color space array. This field may take one of the following values:

`INTENT_PERCEPTUAL`

`INTENT_SATURATION`

`INTENT_RELATIVE_COLORIMETRIC`

`INTENT_ABSOLUTE_COLORIMETRIC`

For more information, see [Rendering Intents](#).

`dwCSAType`

Specifies the type of color space array. See [Color Space Type Identifiers](#).

`pPS2ColorSpaceArray`

Pointer to a buffer in which the color space array is to be placed. If the *pBuffer* pointer is set to **NULL**, the function returns the required size of the buffer in the memory location pointed to by *pcbSize*.

`pcbPS2ColorSpaceArray`

Pointer to a variable containing the size of the buffer in bytes. On return, it contains the number of bytes copied into the buffer.

`pbBinary`

Pointer to a Boolean variable. If set to **TRUE**, the data copied could be binary. If set to **FALSE**, data should be encoded as ASCII85. On return, the memory location pointed to by *pbBinary* indicates whether the data returned actually is binary (**TRUE**) or ASCII85 (**FALSE**).

## Return value

If this function succeeds, the return value is **TRUE**. It also returns **TRUE** if the *pBuffer* parameter is **NULL** and the size required for the buffer is copied into *pcbSize*.

If this function fails, the return value is **FALSE**. For extended error information, call **GetLastError**.

## Remarks

If the color space array is not available in the profile, the **GetPS2ColorSpaceArray** function builds a PostScript Level 2 color space array using the profile contents. This array can then be used as the operand for the PostScript Level2 `setcolorspace` operator.

This method does not support WCS profiles.

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h

Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# GetStandardColorSpaceProfileA function (icm.h)

Article07/27/2022

Retrieves the color profile registered for the specified standard [color space](#).

## Syntax

C++

```
BOOL GetStandardColorSpaceProfileA(
    PCSTR pMachineName,
    DWORD dwSCS,
    PSTR pBuffer,
    PDWORD pcbSize
);
```

## Parameters

pMachineName

Reserved. Must be **NULL**. This parameter is intended to point to the name of the computer on which to get a standard color space profile. A **NULL** pointer indicates the local machine.

dwSCS

Specifies the ID value of the standard color space for which to retrieve the profile. The only valid values for this parameter are **LCS\_sRGB** and **LCS\_WINDOWS\_COLOR\_SPACE**.

pBuffer

Pointer to the buffer in which the name of the profile is to be placed. If **NULL**, the call will return **TRUE** and the required size of the buffer is placed in *pdwSize*.

pcbSize

Pointer to a variable containing the size in bytes of the buffer pointed to by *pProfileName*. On return, the variable contains the size of the buffer actually used or needed.

# Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call **GetLastError**.

## Remarks

If the buffer pointed to by *pProfileName* is to be dynamically allocated by an application, the application can call the **GetStandardColorSpaceProfile** function to retrieve the size required for the buffer. If **GetStandardColorSpaceProfile** is called with *pProfileName* set to **NULL**, it will return **FALSE** and the **DWORD** pointed at by *pdwSize* will contain the number of bytes needed for the buffer pointed at by *pProfileName*. The application can then allocate the buffer and call **GetStandardColorSpaceProfile** again with *pProfileName* set to the address of the buffer.

This function supports Windows Color System (WCS) device model profiles (DMPs) in addition to International Color Consortium (ICC) profiles. It does not support WCS CAMP or GMMP profiles and will return an error if such profiles are used.

### Overview of Windows Vista Specific Functionality

This will support WCS DMPs in addition to ICC profiles. It will not support WCS CAMP or GMMP profiles and will return an error if such profiles are used with this API.

#### *Per-user/LUA support*

This will retrieve the color profile registered for the given standard color space for current user. If there is no such setting for the current user, it retrieves the system wide setting.

This uses **WcsGetDefaultColorProfile** with  
**WCS\_PROFILE\_MANAGEMENT\_SCOPE\_CURRENT\_USER**.

This is executable in LUA context.

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
--------------------------	---

Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [SetStandardColorSpaceProfile](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# GetStandardColorSpaceProfileW function (icm.h)

Article 07/27/2022

Retrieves the color profile registered for the specified standard [color space](#).

## Syntax

C++

```
BOOL GetStandardColorSpaceProfileW(
    PCWSTR pMachineName,
    DWORD dwSCS,
    PWSTR pBuffer,
    PDWORD pcbSize
);
```

## Parameters

`pMachineName`

Reserved. Must be **NULL**. This parameter is intended to point to the name of the computer on which to get a standard color space profile. A **NULL** pointer indicates the local machine.

`dwSCS`

Specifies the ID value of the standard color space for which to retrieve the profile. The only valid values for this parameter are `LCS_sRGB` and `LCS_WINDOWS_COLOR_SPACE`.

`pBuffer`

Pointer to the buffer in which the name of the profile is to be placed. If **NULL**, the call will return **TRUE** and the required size of the buffer is placed in `pdwSize`.

`pcbSize`

Pointer to a variable containing the size in bytes of the buffer pointed to by `pProfileName`. On return, the variable contains the size of the buffer actually used or needed.

# Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Remarks

If the buffer pointed to by *pProfileName* is to be dynamically allocated by an application, the application can call the [GetStandardColorSpaceProfile](#) function to retrieve the size required for the buffer. If [GetStandardColorSpaceProfile](#) is called with *pProfileName* set to **NULL**, it will return **FALSE** and the **DWORD** pointed at by *pdwSize* will contain the number of bytes needed for the buffer pointed at by *pProfileName*. The application can then allocate the buffer and call [GetStandardColorSpaceProfile](#) again with *pProfileName* set to the address of the buffer.

This function supports Windows Color System (WCS) device model profiles (DMPs) in addition to International Color Consortium (ICC) profiles. It does not support WCS CAMP or GMMP profiles and will return an error if such profiles are used.

### Overview of Windows Vista Specific Functionality

This will support WCS DMPs in addition to ICC profiles. It will not support WCS CAMP or GMMP profiles and will return an error if such profiles are used with this API.

#### *Per-user/LUA support*

This will retrieve the color profile registered for the given standard color space for current user. If there is no such setting for the current user, it retrieves the system wide setting.

This uses [WcsGetDefaultColorProfile](#) with  
`WCS_PROFILE_MANAGEMENT_SCOPE_CURRENT_USER`.

This is executable in LUA context.

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
--------------------------	---

Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [SetStandardColorSpaceProfile](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# GRAYCOLOR structure (icm.h)

Article02/22/2024

Description of the GRAYCOLOR structure.

## Syntax

C++

```
struct GRAYCOLOR {
    WORD gray;
};
```

## Members

gray

TBD

## Requirements

[+] Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# HiFiCOLOR structure (icm.h)

Article02/22/2024

Description of the HiFiCOLOR structure.

## Syntax

C++

```
struct HiFiCOLOR {
    BYTE channel[MAX_COLOR_CHANNELS];
};
```

## Members

channel[MAX\_COLOR\_CHANNELS]

TBD

## Requirements

[+] Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# InstallColorProfileA function (icm.h)

Article02/22/2024

Installs a given profile for use on a specified machine. The profile is also copied to the COLOR directory.

## Syntax

C++

```
BOOL InstallColorProfileA(  
    PCSTR pMachineName,  
    PCSTR pProfileName  
) ;
```

## Parameters

pMachineName

Reserved. Must be **NULL**. This parameter is intended to point to the name of the computer on which the profile is to be installed. A **NULL** pointer indicates the local computer.

pProfileName

Pointer to the fully qualified path name of the profile to install.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call **GetLastError**.

## Requirements

[+] Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# InstallColorProfileW function (icm.h)

Article02/22/2024

Installs a given profile for use on a specified machine. The profile is also copied to the COLOR directory.

## Syntax

C++

```
BOOL InstallColorProfileW(  
    PCWSTR pMachineName,  
    PCWSTR pProfileName  
) ;
```

## Parameters

pMachineName

Reserved. Must be **NULL**. This parameter is intended to point to the name of the computer on which the profile is to be installed. A **NULL** pointer indicates the local computer.

pProfileName

Pointer to the fully qualified path name of the profile to install.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call **GetLastError**.

## Requirements

[+] Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# IsColorProfileTagPresent function (icm.h)

Article 02/22/2024

Reports whether a specified International Color Consortium (ICC) tag is present in the specified color profile.

## Syntax

C++

```
BOOL IsColorProfileTagPresent(
    HPROFILE hProfile,
    TAGTYPE tag,
    PBOOL    pbPresent
);
```

## Parameters

`hProfile`

Specifies a handle to the ICC profile in question.

`tag`

Specifies the ICC tag to check.

`pbPresent`

Pointer to a variable that is set to **TRUE** on return if the specified ICC tag is present, or **FALSE** if not.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call **GetLastError**.

## Remarks

This function will fail if *hProfile* is not a valid ICC profile.

This function does not support Windows Color System (WCS) profiles CAMP, DMP, and GMMP; because profile elements are implicitly associated with and hard coded to ICC tag types and there exist many robust XML parsing libraries.

## Requirements

  Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# IsColorProfileValid function (icm.h)

Article 02/22/2024

Allows you to determine whether the specified profile is a valid International Color Consortium (ICC) profile, or a valid Windows Color System (WCS) profile handle that can be used for color management. WCS profile validation doesn't invoke the underlying device models, but instead simply validates against the XML schema and the schema element range limits.

## Syntax

C++

```
BOOL IsColorProfileValid(
    HPROFILE hProfile,
    PBOOL     pbValid
);
```

## Parameters

`hProfile`

Specifies a handle to the profile to be validated. The function determines whether the HPROFILE contains ICC or WCS profile information.

`pbValid`

Pointer to a variable that is set to **TRUE** on return if the operation succeeds and the profile is a valid ICC or WCS profile. If the operation fails or the profile is not valid the variable is **FALSE**.

## Return value

If this function succeeds and the profile is valid, the return value is **TRUE**.

If this function fails (or succeeds and the profile is not valid), the return value is **FALSE**. For extended error information, call **GetLastError**.

## Requirements

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 [Yes](#) [No](#)

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# LabCOLOR structure (icm.h)

Article02/22/2024

TBD

## Syntax

C++

```
struct LabCOLOR {
    WORD L;
    WORD a;
    WORD b;
};
```

## Members

L

TBD

a

TBD

b

TBD

## Requirements

[+] Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h

# Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# NAMED\_PROFILE\_INFO structure (icm.h)

Article 02/22/2024

The NAMED\_PROFILE\_INFO structure is used to store information about a named color profile.

## Syntax

C++

```
typedef struct tagNAMED_PROFILE_INFO {
    DWORD      dwFlags;
    DWORD      dwCount;
    DWORD      dwCountDevCoordinates;
    COLOR_NAME szPrefix;
    COLOR_NAME szSuffix;
} NAMED_PROFILE_INFO;
```

## Members

`dwFlags`

Not currently used by the default CMM.

`dwCount`

Total number of named colors in the profile.

`dwCountDevCoordinates`

Total number of device coordinates for each named color.

`szPrefix`

Pointer to a string containing the prefix for each color name.

`szSuffix`

Pointer to a string containing the suffix for each color name.

## Requirements

Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) | [Get help at Microsoft Q&A](#)

# NAMEDCOLOR structure (icm.h)

Article02/22/2024

TBD

## Syntax

C++

```
struct NAMEDCOLOR {
    DWORD dwIndex;
};
```

## Members

`dwIndex`

TBD

## Requirements

[+] Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# OpenColorProfileA function (icm.h)

Article 07/27/2022

Creates a handle to a specified color profile. The handle can then be used in other profile management functions.

## Syntax

C++

```
HPROFILE OpenColorProfileA(
    PPROFILE pProfile,
    DWORD     dwDesiredAccess,
    DWORD     dwShareMode,
    DWORD     dwCreationMode
);
```

## Parameters

**pProfile**

Pointer to a color profile structure specifying the profile. The *pProfile* pointer can be freed as soon as the handle is created.

**dwDesiredAccess**

Specifies how to access the given profile. This parameter must take one the following constant values.

Value	Meaning
PROFILE_READ	Opens the profile for read access.
PROFILE_READWRITE	Opens the profile for both read and write access. Has no effect for WCS XML profiles.

**dwShareMode**

Specifies how the profile should be shared, if the profile is contained in a file. A value of zero prevents the profile from being shared at all. The parameter can contain one or both of the following constants (combined by addition or logical OR).

Value	Meaning
FILE_SHARE_READ	Other open operations can be performed on the profile for read access.
FILE_SHARE_WRITE	Other open operations can be performed on the profile for write access. Has no effect for WCS XML profiles.

#### dwCreationMode

Specifies which actions to take on the profile while opening it, if it is contained in a file. This parameter must take one of the following constant values.

Value	Meaning
CREATE_NEW	Creates a new profile. Fails if the profile already exists.
CREATE_ALWAYS	Creates a new profile. Overwrites the profile if it exists.
OPEN_EXISTING	Opens the profile. Fails if it does not exist
OPEN_ALWAYS	Opens the profile if it exists. For ICC profiles, if the profile does not exist, creates the profile. For WCS XML profiles, if the profile does not exist, returns an error.
TRUNCATE_EXISTING	Opens the profile, and truncates it to zero bytes, returning a blank ICC profile. Fails if the profile doesn't exist.

## Return value

If this function succeeds, the return value is the handle of the color profile that is opened. For ICC and WCS profiles, a CAMP and GMMP are provided by the function based on the current default CAMP and GMMP in the registry.

When OpenColorProfile encounters an ICC profile with an embedded WCS profile, and if the dwType member within the Profile structure does not take the value DONT\_USE\_EMBEDDED\_WCS\_PROFILES, it should extract and use the WCS profile(s) contained in this WcsProfilesTag. The HPROFILE returned would be a WCS HPROFILE.

If this function fails, the return value is **NULL**. For extended error information, call **GetLastError**.

## Remarks

If the profile data is not specified using a file name, *dwShareMode* and *dwCreationMode* are ignored.

*dwCreationMode* flags CREATE\_NEW, CREATE\_ALWAYS, and TRUNCATE\_EXISTING, will always return blank ICC HPROFILEs. If other *dwCreationMode* flags are present, InternalOpenColorProfile is called (using the flags as provided by the API) to determine whether the profile is ICC or WCS XML.

Within the ICC code path, an ICC HPROFILE is returned using the requested sharing, access and creation flags as specified in the tables above.

Within the WCS path, the *dwCreationMode* flag OPEN\_ALWAYS will fail if the profile doesn't exist, since WCS profiles cannot be created or edited within the WCS architecture (they must be edited outside of it, using MSXML6). For the same reason, *dwShareMode* flag FILE\_SHARE\_WRITE, and *dwDesiredAccess* flag PROFILE\_READWRITE are ignored within the WCS path.

When the function opens the ICC profile, it will look for a *WcsProfilesTag* and, if there is one, it will extract and use the original WCS profiles contained therein. (See [WcsCreateIccProfile](#).)

An HPROFILE with WCS profile information is derived from a DMP by acquiring the default CAMP and default GMMP from the registry. An HPROFILE is a composition of a DMP, CAMP and GMMP.

Once the handle to the color profile is created, any information used to create that handle can be deleted.

Use the [CloseColorProfile](#) function to close an object handle returned by [OpenColorProfile](#).

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- Basic color management concepts
  - Functions
  - CloseColorProfile
  - PROFILE
- 

## Feedback

Was this page helpful?

 Yes

 No

Get help at Microsoft Q&A

# OpenColorProfileW function (icm.h)

Article 07/27/2022

Creates a handle to a specified color profile. The handle can then be used in other profile management functions.

## Syntax

C++

```
HPROFILE OpenColorProfileW(
    PPROFILE pProfile,
    DWORD     dwDesiredAccess,
    DWORD     dwShareMode,
    DWORD     dwCreationMode
);
```

## Parameters

pProfile

Pointer to a color profile structure specifying the profile. The *pProfile* pointer can be freed as soon as the handle is created.

dwDesiredAccess

Specifies how to access the given profile. This parameter must take one the following constant values.

Value	Meaning
PROFILE_READ	Opens the profile for read access.
PROFILE_READWRITE	Opens the profile for both read and write access. Has no effect for WCS XML profiles.

dwShareMode

Specifies how the profile should be shared, if the profile is contained in a file. A value of zero prevents the profile from being shared at all. The parameter can contain one or both of the following constants (combined by addition or logical OR).

Value	Meaning
FILE_SHARE_READ	Other open operations can be performed on the profile for read access.
FILE_SHARE_WRITE	Other open operations can be performed on the profile for write access. Has no effect for WCS XML profiles.

#### dwCreationMode

Specifies which actions to take on the profile while opening it, if it is contained in a file. This parameter must take one of the following constant values.

Value	Meaning
CREATE_NEW	Creates a new profile. Fails if the profile already exists.
CREATE_ALWAYS	Creates a new profile. Overwrites the profile if it exists.
OPEN_EXISTING	Opens the profile. Fails if it does not exist
OPEN_ALWAYS	Opens the profile if it exists. For ICC profiles, if the profile does not exist, creates the profile. For WCS XML profiles, if the profile does not exist, returns an error.
TRUNCATE_EXISTING	Opens the profile, and truncates it to zero bytes, returning a blank ICC profile. Fails if the profile doesn't exist.

## Return value

If this function succeeds, the return value is the handle of the color profile that is opened. For ICC and WCS profiles, a CAMP and GMMP are provided by the function based on the current default CAMP and GMMP in the registry.

When OpenColorProfile encounters an ICC profile with an embedded WCS profile, and if the dwType member within the Profile structure does not take the value DONT\_USE\_EMBEDDED\_WCS\_PROFILES, it should extract and use the WCS profile(s) contained in this WcsProfilesTag. The HPROFILE returned would be a WCS HPROFILE.

If this function fails, the return value is **NULL**. For extended error information, call **GetLastError**.

## Remarks

If the profile data is not specified using a file name, *dwShareMode* and *dwCreationMode* are ignored.

*dwCreationMode* flags CREATE\_NEW, CREATE\_ALWAYS, and TRUNCATE\_EXISTING, will always return blank ICC HPROFILEs. If other *dwCreationMode* flags are present, InternalOpenColorProfile is called (using the flags as provided by the API) to determine whether the profile is ICC or WCS XML.

Within the ICC code path, an ICC HPROFILE is returned using the requested sharing, access and creation flags as specified in the tables above.

Within the WCS path, the *dwCreationMode* flag OPEN\_ALWAYS will fail if the profile doesn't exist, since WCS profiles cannot be created or edited within the WCS architecture (they must be edited outside of it, using MSXML6). For the same reason, *dwShareMode* flag FILE\_SHARE\_WRITE, and *dwDesiredAccess* flag PROFILE\_READWRITE are ignored within the WCS path.

When the function opens the ICC profile, it will look for a *WcsProfilesTag* and, if there is one, it will extract and use the original WCS profiles contained therein. (See [WcsCreateIccProfile](#).)

An HPROFILE with WCS profile information is derived from a DMP by acquiring the default CAMP and default GMMP from the registry. An HPROFILE is a composition of a DMP, CAMP and GMMP.

Once the handle to the color profile is created, any information used to create that handle can be deleted.

Use the [CloseColorProfile](#) function to close an object handle returned by [OpenColorProfile](#).

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- Basic color management concepts
  - Functions
  - CloseColorProfile
  - PROFILE
- 

## Feedback

Was this page helpful?

 Yes

 No

Get help at Microsoft Q&A

# PBMCALLBACKFN callback function (icm.h)

Article 02/22/2024

TBD

## Syntax

C++

```
PBMCALLBACKFN Pbmcallbackfn;

BOOL Pbmcallbackfn(
    ULONG unnamedParam1,
    ULONG unnamedParam2,
    LPARAM unnamedParam3
)
{...}
```

## Parameters

unnamedParam1

TBD

unnamedParam2

TBD

unnamedParam3

TBD

## Requirements

[ ] Expand table

Requirement	Value
Minimum supported client	Windows 10 Build 20348
Minimum supported server	Windows 10 Build 20348

Requirement	Value
Header	icm.h

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# PCMSCALLBACKA callback function (icm.h)

Article 10/05/2021

\***PCMSCALLBACKA**\* (or **ApplyCallbackFunction**) is a callback function that you implement that updates the WCS configuration data while the dialog box displayed by the **SetupColorMatchingW** function is executing. The name **ApplyCallbackFunction** is a placeholder. The actual name of this callback function is supplied by your application using ICM.

## Syntax

C++

```
PCMSCALLBACKA Pcmcallback;  
  
BOOL Pcmcallback(  
    _tagCOLORMATCHSETUPA *unnamedParam1,  
    LPARAM unnamedParam2  
)  
{...}
```

## Parameters

unnamedParam1

Pointer to a **COLORMATCHSETUPW** structure that contains WCS configuration data.

unnamedParam2

Contains a value supplied by the application.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. The callback function can set the extended error information by calling **SetLastError**.

## Remarks

The [ApplyCallbackFunction](#) function is used to change the WCS configuration for a device while the Color Management dialog box is displayed. The Color Management dialog box is displayed by the [SetupColorMatchingW](#) function.

If the callback function is provided, an **Apply** button is displayed in the lower right of the dialog box. When you select the **Apply** button, the callback function immediately updates the configuration for the device being set up. The Color Management dialog box remains on the screen.

An application supplies a callback function to WCS by storing the address of the callback function in the [COLORMATCHSETUPW](#) structure that is passed to the [SetupColorMatchingW](#) function. The address is stored in the [IPfnApplyCallback](#) ↗ member of the [COLORMATCHSETUP](#) structure. The **dwFlags** member should be set to **CMS\_USEAPPLYCALLBACK**, or the callback function will be ignored.

A value supplied by the application may be passed to the callback function. Prior to invoking the [SetupColorMatchingW](#) function, the application can store a value in the [IParamApplyCallback](#) ↗ member of the [COLORMATCHSETUPW](#) structure. When the callback function is invoked, the value in the [IParamApplyCallback](#) structure member will be passed to the callback function in its *IParam* parameter.

The callback function is completely optional. If it is not supplied, the **Apply** button does not appear in the Color Management dialog box. Microsoft strongly recommends that your application supplies a callback function.

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [SetupColorMatchingW](#)
- [COLORMATCHSETUPW](#)

# Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# PCMSCALLBACKW callback function (icm.h)

Article 10/05/2021

\***PCMSCALLBACKW**\* (or **ApplyCallbackFunction**) is a callback function that you implement that updates the WCS configuration data while the dialog box displayed by the **SetupColorMatchingW** function is executing. The name **ApplyCallbackFunction** is a placeholder. The actual name of this callback function is supplied by your application using ICM.

## Syntax

C++

```
PCMSCALLBACKW Pcmcallbackw;

BOOL Pcmcallbackw(
    _tagCOLORMATCHSETUPW *unnamedParam1,
    LPARAM unnamedParam2
)
{...}
```

## Parameters

unnamedParam1

Pointer to a **COLORMATCHSETUPW** structure that contains WCS configuration data.

unnamedParam2

Contains a value supplied by the application.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. The callback function can set the extended error information by calling **SetLastError**.

## Remarks

The [ApplyCallbackFunction](#) function is used to change the WCS configuration for a device while the Color Management dialog box is displayed. The Color Management dialog box is displayed by the [SetupColorMatchingW](#) function.

If the callback function is provided, an **Apply** button is displayed in the lower right of the dialog box. When you select the **Apply** button, the callback function immediately updates the configuration for the device being set up. The Color Management dialog box remains on the screen.

An application supplies a callback function to WCS by storing the address of the callback function in the [COLORMATCHSETUPW](#) structure that is passed to the [SetupColorMatchingW](#) function. The address is stored in the [IPfnApplyCallback](#) ↗ member of the [COLORMATCHSETUP](#) structure. The **dwFlags** member should be set to **CMS\_USEAPPLYCALLBACK**, or the callback function will be ignored.

A value supplied by the application may be passed to the callback function. Prior to invoking the [SetupColorMatchingW](#) function, the application can store a value in the [IParamApplyCallback](#) ↗ member of the [COLORMATCHSETUPW](#) structure. When the callback function is invoked, the value in the [IParamApplyCallback](#) structure member will be passed to the callback function in its *IParam* parameter.

The callback function is completely optional. If it is not supplied, the **Apply** button does not appear in the Color Management dialog box. Microsoft strongly recommends that your application supplies a callback function.

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [SetupColorMatchingW](#)
- [COLORMATCHSETUPW](#)

# Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# PROFILE structure (icm.h)

Article02/22/2024

Contains information that defines a color profile. See [Using device profiles with WCS](#) for more information.

## Syntax

C++

```
typedef struct tagPROFILE {
    DWORD dwType;
    PVOID pProfileData;
    DWORD cbDataSize;
} PROFILE;
```

## Members

**dwType**

Must be set to one of the following values.

[+] [Expand table](#)

Value	Meaning
PROFILE_FILENAME	Indicates that the <b>pProfileData</b> member contains a null-terminated string that holds the name of a device profile file.
PROFILE_MEMBUFFER	Indicates that the <b>pProfileData</b> member contains a pointer to a device profile in a memory buffer.

**pProfileData**

The contents of this member is indicated by the **dwTYPE** member. It will either be a pointer to a null-terminated string containing the file name of the device profile, or it will be a pointer to a buffer in memory containing the device profile data.

**cbDataSize**

The size in bytes of the data buffer pointed to by the **pProfileData** member.

# Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h

## See also

- [Using device profiles with WCS](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) | [Get help at Microsoft Q&A](#)

# PROFILEHEADER structure (icm.h)

Article09/01/2022

Contains information that describes the contents of a device profile file. This header occurs at the beginning of a device profile file.

## Syntax

C++

```
typedef struct tagPROFILEHEADER {
    DWORD    phSize;
    DWORD    phCMMType;
    DWORD    phVersion;
    DWORD    phClass;
    DWORD    phDataColorSpace;
    DWORD    phConnectionSpace;
    DWORD    phDateTime[3];
    DWORD    phSignature;
    DWORD    phPlatform;
    DWORD    phProfileFlags;
    DWORD    phManufacturer;
    DWORD    phModel;
    DWORD    phAttributes[2];
    DWORD    phRenderingIntent;
    CIEXYZ  phIlluminant;
    DWORD    phCreator;
    BYTE     phReserved[44];
} PROFILEHEADER;
```

## Members

`phSize`

The size of the profile in bytes.

`phCMMType`

The identification number of the CMM that is used in the profile. Identification numbers are registered with the ICC.

`phVersion`

The version number of the profile. The version number is determined by the ICC. The current major version number is 02h. The current minor version number is 10h. The

major and minor version numbers are in binary coded decimal (BCD). They must be stored in the following format.

Byte Number	Contents
0	Major version number in BCD.
1	Minor version number in the most significant nibble of this byte. Bug fix version number in the least significant nibble.
2	Reserved. Must be set to 0.
3	Reserved. Must be set to 0.

#### phClass

Indicates the profile class. For a description of profile classes, see [Using Device Profiles with WCS](#). A profile class may have any of the following values.

Profile Class	Signature
Input Device Profile	CLASS_SCANNER
Display Device Profile	CLASS_MONITOR
Output Device Profile	CLASS_PRINTER
Device Link Profile	CLASS_LINK
Color Space Conversion Profile	CLASS_COLORSPACE
Abstract Profile	CLASS_ABSTRACT
Named Color Profile	CLASS_NAMED
Color Appearance Model Profile	CLASS_CAMP
Color Gamut Map Model Profile	CLASS_GMMP

#### phDataColorSpace

A signature value that indicates the color space in which the profile data is defined. The member can be any of value from the [Color Space Constants](#).

#### phConnectionSpace

A signature value that indicates the color space in which the profile connection space (PCS) is defined. The member can be any of the following values.

<b>Profile Class</b>	<b>Signature</b>
XYZ	SPACE_XYZ
Lab	SPACE_Lab

When the **phClass** member is set to CLASS\_LINK, the PCS is taken from the **phDataColorSpace** member.

**phDateTime[3]**

The date and time that the profile was created.

**phSignature**

Reserved for internal use.

**phPlatform**

The primary platform for which the profile was created. The primary platform can be set to any of the following values.

<b>Platform</b>	<b>Value</b>
Apple Computer, Inc.	'APPL'
Microsoft Corp.	'MSFT'
Silicon Graphics, Inc.	'SGI'
Sun Microsystems, Inc.	'SUNW'
Taligent	'TGNT'

**phProfileFlags**

Bit flags containing hints that the CMM uses to interpret the profile data. The member can be set to the following values.

<b>Constant</b>	<b>Meaning</b>
FLAG_EMBEDDEDPROFILE	The profile is embedded in a bitmap file.

Constant	Meaning
FLAG_DEPENDENTONDATA	The profile can't be used independently of the embedded color data. Used for profiles that are embedded in bitmap files.

`phManufacturer`

The identification number of the device profile manufacturer. All manufacturer identification numbers are registered with the ICC.

`phModel`

The device manufacturer's device model number. All model identification numbers are registered with the ICC.

`phAttributes[2]`

Attributes of profile. The profile attributes can be any of the following values.

Constant	Meaning
ATTRIB_TRANSPARENCY	Turns transparency on. If this flag is not used, the attribute is reflective by default.
ATTRIB_MATTE	Turns matte display on. If this flag is not used, the attribute is glossy by default.

`phRenderingIntent`

The profile rendering intent. The member can be set to one of the following values:

`INTENT_PERCEPTUAL`

`INTENT_SATURATION`

`INTENT_RELATIVE_COLORIMETRIC`

`INTENT_ABSOLUTE_COLORIMETRIC`

For more information, see [Rendering intents](#).

`phIlluminant`

Profile illuminant.

phCreator

Signature of the software that created the profile. Signatures are registered with the ICC.

phReserved[44]

Reserved.

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h

## See also

- [Further information](#)
- [Using device profiles with WCS](#)
- [Rendering intents](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# RegisterCMMA function (icm.h)

Article02/22/2024

Associates a specified identification value with the specified color management module dynamic link library (CMM DLL). When this ID appears in a color profile, Windows can then locate the corresponding CMM so as to create a transform.

## Syntax

C++

```
BOOL RegisterCMMA(
    PCSTR pMachineName,
    DWORD cmmID,
    PCSTR pCMMdll
);
```

## Parameters

pMachineName

Reserved; must currently be set to **NULL**, until non-local registration is supported. This parameter is intended to point to the name of the machine on which a CMM DLL should be registered. A **NULL** pointer indicates the local machine.

cmmID

Specifies the ID signature of the CMM registered with the International Color Consortium (ICC).

pCMMdll

Pointer to the fully qualified path name of the CMM DLL.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call **GetLastError**.

# Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) | [Get help at Microsoft Q&A](#)

# RegisterCMMW function (icm.h)

Article02/22/2024

Associates a specified identification value with the specified color management module dynamic link library (CMM DLL). When this ID appears in a color profile, Windows can then locate the corresponding CMM so as to create a transform.

## Syntax

C++

```
BOOL RegisterCMMW(
    PCWSTR pMachineName,
    DWORD cmmID,
    PCWSTR pCMMdll
);
```

## Parameters

pMachineName

Reserved; must currently be set to **NULL**, until non-local registration is supported. This parameter is intended to point to the name of the machine on which a CMM DLL should be registered. A **NULL** pointer indicates the local machine.

cmmID

Specifies the ID signature of the CMM registered with the International Color Consortium (ICC).

pCMMdll

Pointer to the fully qualified path name of the CMM DLL.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call **GetLastError**.

# Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) | [Get help at Microsoft Q&A](#)

# RGBCOLOR structure (icm.h)

Article02/22/2024

TBD

## Syntax

C++

```
struct RGBCOLOR {
    WORD red;
    WORD green;
    WORD blue;
};
```

## Members

red

TBD

green

TBD

blue

TBD

## Requirements

[+] Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h

# Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# SelectCMM function (icm.h)

Article02/22/2024

Allows you to select the preferred color management module (CMM) to use.

## Syntax

C++

```
BOOL SelectCMM(  
    DWORD dwCMMType  
) ;
```

## Parameters

`dwCMMType`

Specifies the signature of the desired CMM as registered with the International Color Consortium (ICC).

**Windows 2000 only:** Setting this parameter to **NULL** causes the WCS system to select the default CMM.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Remarks

For **SelectCMM** to succeed, the specified CMM must be registered with the system.

## Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# SetColorProfileElement function (icm.h)

Article 02/22/2024

Sets the element data for a tagged profile element in an ICC color profile.

## Syntax

C++

```
BOOL SetColorProfileElement(
    HPROFILE hProfile,
    TAGTYPE tag,
    DWORD dwOffset,
    PDWORD pcbElement,
    PVOID pElement
);
```

## Parameters

`hProfile`

Specifies a handle to the ICC profile in question.

`tag`

Identifies the tagged element.

`dwOffset`

Specifies the offset from the first byte of the tagged element data at which to start writing.

`pcbElement`

Pointer to a variable containing the number of bytes of data to write. On return, it contains the number of bytes actually written.

`pElement`

Pointer to a buffer containing the data to write to the tagged element in the color profile.

# Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Remarks

This function will fail if *hProfile* is not a valid ICC profile.

If the color profile is not opened for read/write permission, this function fails.

If *dwOffset* exceeds the size set for the specified tagged element, this function fails.

If *dwOffset* + *\*pcbSize* is greater than the size of the specified element, this function only writes as many bytes as will fit within the current size of the element.

Any existing data in the specified portion of the tagged element is overwritten when this function succeeds.

This function does not support Windows Color System (WCS) profiles CAMP, DMP, and GMMP; because profile elements are implicitly associated with and hard coded to ICC tag types and there exist many robust XML parsing libraries.

## Requirements

[+] [Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)

- Functions
- 

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# SetColorProfileElementReference function (icm.h)

Article02/22/2024

Creates in a specified ICC color profile a new tag that references the same data as an existing tag.

## Syntax

C++

```
BOOL SetColorProfileElementReference(
    HPROFILE hProfile,
    TAGTYPE newTag,
    TAGTYPE refTag
);
```

## Parameters

`hProfile`

Specifies a handle to the ICC color profile in question.

`newTag`

Identifies the new tag to create.

`refTag`

Identifies the existing tag whose data is to be referenced by the new tag.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Remarks

This function will fail if *hProfile* is not a valid ICC profile.

If *newTag* already exists or *refTag* does not exist, **SetColorProfileElementReference** fails.

If the color profile was not opened with read/write permission,  
**SetColorProfileElementReference** fails.

This function does not support Windows Color System (WCS) profiles CAMP, DMP, and GMMP; because profile elements are implicitly associated with and hard coded to ICC tag types and there exist many robust XML parsing libraries.

## Requirements

[ ] Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# SetColorProfileElementSize function (icm.h)

Article 10/05/2021

Sets the size of a tagged element in an ICC color profile.

## Syntax

C++

```
BOOL SetColorProfileElementSize(
    HPROFILE hProfile,
    TAGTYPE tagType,
    DWORD    pcbElement
);
```

## Parameters

`hProfile`

Specifies a handle to the ICC color profile in question.

`tagType`

Identifies the tagged element.

`pcbElement`

Specifies the size to set the tagged element to. If `cbSize` is zero, this function deletes the specified tagged element. If the tag is a reference, only the tag table entry is deleted, not the data.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Remarks

This function will fail if *hProfile* is not a valid ICC profile.

To create a new tagged element in a color profile, use **SetColorProfileElementSize** to set the size, then use **SetColorProfileElement** to set the element value.

If the specified tag already exists in the profile, **SetColorProfileElementSize** changes the size of the element by truncating it or adding zeroes at the end as the case may be.

If the specified tag already exists and is a reference to another tag, **SetColorProfileElementSize** creates a new data area for the tag that is not shared.

This function does not support Windows Color System (WCS) profiles CAMP, DMP, and GMMP; because profile elements are implicitly associated with and hard coded to ICC tag types and there exist many robust XML parsing libraries.

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [SetColorProfileElement](#)

---

## Feedback

Was this page helpful?



[Get help at Microsoft Q&A](#)

# SetColorProfileHeader function (icm.h)

Article02/22/2024

Sets the header data in a specified ICC color profile.

## Syntax

C++

```
BOOL SetColorProfileHeader(
    HPROFILE      hProfile,
    PPROFILEHEADER pHeader
);
```

## Parameters

`hProfile`

Specifies a handle to the ICC color profile in question.

`pHeader`

Pointer to the profile header data to write to the specified profile.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call `GetLastError`.

## Remarks

This function will fail if `hProfile` is not a valid ICC profile.

If the color profile was not opened with read/write permission, `SetColorProfileHeader` fails.

`SetColorProfileHeader` overwrites the current header in the ICC profile.

This function does not support Windows Color System (WCS) profiles CAMP, DMP, and GMMP; because profile elements are implicitly associated with and hard coded to ICC tag types and there exist many robust XML parsing libraries.

## Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [PROFILEHEADER](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# SetStandardColorSpaceProfileA function (icm.h)

Article07/27/2022

Registers a specified profile for a given standard [color space](#). The profile can be queried using [GetStandardColorSpaceProfileW](#).

## Syntax

C++

```
BOOL SetStandardColorSpaceProfileA(  
    PCSTR pMachineName,  
    DWORD dwProfileID,  
    PCSTR pProfilename  
>;
```

## Parameters

`pMachineName`

Reserved. Must be **NULL**. This parameter is intended to point to the name of the machine on which to set a standard color space profile. A **NULL** pointer indicates the local machine.

`dwProfileID`

Specifies the ID value of the standard color space that the given profile represents. This is a custom ID value used to uniquely identify the color space profile within your application.

`pProfilename`

Points to a fully qualified path to the profile file.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Remarks

The profile must already be installed on the system before it can be registered for a standard color space.

This function supports Windows Color System (WCS) device model profiles (DMPs) in addition to International Color Consortium (ICC) profiles. It does not support WCS CAMP or GMMP profiles and will return an error if such profiles are used.

### *Per-user/LUA support*

This will register a specified profile for a given standard color space only for current user.

This uses [WcsSetDefaultColorProfile](#) with  
WCS\_PROFILE\_MANAGEMENT\_SCOPE\_CURRENT\_USER.

This is executable in LUA context if the profile is already installed, fails otherwise with access denied since install is system-wide and requires administrator privileges.

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [SetStandardColorSpaceProfileW](#)

# Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# SetStandardColorSpaceProfileW function (icm.h)

Article07/27/2022

Registers a specified profile for a given standard [color space](#). The profile can be queried using [GetStandardColorSpaceProfileW](#).

## Syntax

C++

```
BOOL SetStandardColorSpaceProfileW(  
    PCWSTR pMachineName,  
    DWORD dwProfileID,  
    PCWSTR pProfileName  
) ;
```

## Parameters

`pMachineName`

Reserved. Must be **NULL**. This parameter is intended to point to the name of the machine on which to set a standard color space profile. A **NULL** pointer indicates the local machine.

`dwProfileID`

Specifies the ID value of the standard color space that the given profile represents. This is a custom ID value used to uniquely identify the color space profile within your application.

`pProfileName`

Points to a fully qualified path to the profile file.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Remarks

The profile must already be installed on the system before it can be registered for a standard color space.

This function supports Windows Color System (WCS) device model profiles (DMPs) in addition to International Color Consortium (ICC) profiles. It does not support WCS CAMP or GMMP profiles and will return an error if such profiles are used.

### *Per-user/LUA support*

This will register a specified profile for a given standard color space only for current user.

This uses [WcsSetDefaultColorProfile](#) with  
WCS\_PROFILE\_MANAGEMENT\_SCOPE\_CURRENT\_USER.

This is executable in LUA context if the profile is already installed, fails otherwise with access denied since install is system-wide and requires administrator privileges.

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [SetStandardColorSpaceProfileW](#)

# Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# SetupColorMatchingA function (icm.h)

Article02/22/2024

Creates a Color Management dialog box that lets the user choose whether to enable color management and, if so, provides control over the color profiles used and over the rendering intent.

## Syntax

C++

```
BOOL SetupColorMatchingA(  
    PCOLORMATCHSETUPA pcms  
) ;
```

## Parameters

pcms

Pointer to a [COLORMATCHSETUPW](#) structure that on entry contains information used to initialize the dialog box.

When [SetupColorMatching](#) returns, if the user clicked the OK button, this structure contains information about the user's selection. Otherwise, if an error occurred or the user canceled the dialog box, the structure is left unchanged.

## Return value

If this function succeeds, the return value is **TRUE** indicating that no errors occurred and the user clicked the OK button.

If this function fails, the return value is **FALSE** indicating that an error occurred or the dialog was canceled. For extended error information, call [GetLastError](#).

## Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Icmui.lib
DLL	Icmui.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# SetupColorMatchingW function (icm.h)

Article02/22/2024

Creates a Color Management dialog box that lets the user choose whether to enable color management and, if so, provides control over the color profiles used and over the rendering intent.

## Syntax

C++

```
BOOL SetupColorMatchingW(  
    PCOLORMATCHSETUPW pcms  
);
```

## Parameters

pcms

Pointer to a **COLORMATCHSETUPW** structure that on entry contains information used to initialize the dialog box.

When **SetupColorMatching** returns, if the user clicked the OK button, this structure contains information about the user's selection. Otherwise, if an error occurred or the user canceled the dialog box, the structure is left unchanged.

## Return value

If this function succeeds, the return value is **TRUE** indicating that no errors occurred and the user clicked the OK button.

If this function fails, the return value is **FALSE** indicating that an error occurred or the dialog was canceled. For extended error information, call **GetLastError**.

## Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Icmui.lib
DLL	Icmui.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# TranslateBitmapBits function (icm.h)

Article 10/05/2021

Translates the colors of a bitmap having a defined format so as to produce another bitmap in a requested format.

## Syntax

C++

```
BOOL TranslateBitmapBits(
    HTRANSFORM    hColorTransform,
    PVOID         pSrcBits,
    BMFORMAT      bmInput,
    DWORD         dwWidth,
    DWORD         dwHeight,
    DWORD         dwInputStride,
    PVOID         pDestBits,
    BMFORMAT      bmOutput,
    DWORD         dwOutputStride,
    PBMCALLBACKFN pfnCallBack,
    LPARAM        ulCallbackData
);
```

## Parameters

`hColorTransform`

Identifies the color transform to use.

`pSrcBits`

Pointer to the bitmap to translate.

`bmInput`

Specifies the format of the input bitmap. Must be set to one of the values of the [BMFORMAT](#) enumerated type.

### Note

This function doesn't support `BM_XYZTRIPLETS` or `BM_YxyTRIPLETS` as inputs.

`dwWidth`

Specifies the number of pixels per scan line in the input bitmap.

`dwHeight`

Specifies the number of scan lines in the input bitmap.

`dwInputStride`

Specifies the number of bytes from the beginning of one scan line to the beginning of the next in the input bitmap; if set to zero, the function assumes that scan lines are padded so as to be **DWORD**-aligned.

`pDestBits`

Pointer to the buffer in which to place the translated bitmap.

`bmOutput`

Specifies the format of the output bitmap. Must be set to one of the values of the **BMFORMAT** enumerated type.

`dwOutputStride`

Specifies the number of bytes from the beginning of one scan line to the beginning of the next in the output bitmap; if set to zero, the function assumes that scan lines should be padded to be **DWORD**-aligned.

`pfnCallBack`

Pointer to a callback function called periodically by **TranslateBitmapBits** to report progress and allow the calling process to cancel the translation. (See **ICMProgressProcCallback**)

`ulCallbackData`

Data passed back to the callback function, for example, to identify the translation that is reporting progress.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call **GetLastError**.

## Remarks

If the input and output formats are not compatible with the color transform, this function fails.

When either of the floating point BMFORMATs, BM\_32b\_scARGB or BM\_32b\_scRGB are used, the color data being translated should not contain NaN or infinity. NaN and infinity are not considered to represent legitimate color component values, and the result of translating pixels containing NaN or infinity is meaningless in color terms. NaN or infinity values in the color data being processed will be handled silently, and an error will not be returned.

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [ICMProgressProcCallback](#)
- [Windows bitmap header structures](#)
- [BMFORMAT](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# TranslateColors function (icm.h)

Article 10/05/2021

Translates an array of colors from the source [color space](#) to the destination color space as defined by a color transform.

## Syntax

C++

```
BOOL TranslateColors(
    HTRANSFORM hColorTransform,
    PCOLOR     paInputColors,
    DWORD      nColors,
    COLORTYPE   ctInput,
    PCOLOR     paOutputColors,
    COLORTYPE   ctOutput
);
```

## Parameters

`hColorTransform`

Identifies the color transform to use.

`paInputColors`

Pointer to an array of *nColors*[COLOR](#) structures to translate.

`nColors`

Contains the number of elements in the arrays pointed to by *paInputColors* and *paOutputColors*.

`ctInput`

Specifies the input color type.

`paOutputColors`

Pointer to an array of *nColors* [COLOR](#) structures that receive the translated colors.

`ctOutput`

Specifies the output color type.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Remarks

If the input and the output color types are not compatible with the color transform, this function fails.

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# UninstallColorProfileA function (icm.h)

Article 02/22/2024

Removes a specified color profile from a specified computer. Associated files are optionally deleted from the system.

## Syntax

C++

```
BOOL UninstallColorProfileA(  
    PCSTR pMachineName,  
    PCSTR pProfileName,  
    BOOL bDelete  
) ;
```

## Parameters

pMachineName

Reserved. Must be **NULL**. This parameter is intended to point to the name of the machine from which to uninstall the specified profile. A **NULL** pointer indicates the local machine.

pProfileName

Points to the file name of the profile to uninstall.

bDelete

If set to **TRUE**, the function deletes the profile from the COLOR directory. If set to **FALSE**, this function has no effect.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call **GetLastError**.

# Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) | [Get help at Microsoft Q&A](#)

# UninstallColorProfileW function (icm.h)

Article 02/22/2024

Removes a specified color profile from a specified computer. Associated files are optionally deleted from the system.

## Syntax

C++

```
BOOL UninstallColorProfileW(
    PCWSTR pMachineName,
    PCWSTR pProfileName,
    BOOL    bDelete
);
```

## Parameters

pMachineName

Reserved. Must be **NULL**. This parameter is intended to point to the name of the machine from which to uninstall the specified profile. A **NULL** pointer indicates the local machine.

pProfileName

Points to the file name of the profile to uninstall.

bDelete

If set to **TRUE**, the function deletes the profile from the COLOR directory. If set to **FALSE**, this function has no effect.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call **GetLastError**.

# Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) | [Get help at Microsoft Q&A](#)

# UnregisterCMMA function (icm.h)

Article02/22/2024

Dissociates a specified ID value from a given color management module dynamic-link library (CMM DLL).

## Syntax

C++

```
BOOL UnregisterCMMA(
    PCSTR pMachineName,
    DWORD cmmID
);
```

## Parameters

pMachineName

Reserved; must currently be set to **NULL**, until non-local registration is supported. This parameter is intended to point to the name of the computer on which a CMM DLLs registration should be removed. A **NULL** pointer indicates the local computer.

cmmID

Specifies the ID value identifying the CMM whose registration is to be removed. This is the signature of the CMM registered with the International Color Consortium (ICC).

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Remarks

If the profile for creating a transform later specifies this ID, the Windows default CMM will be used rather than this CMM DLL.

# Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) | [Get help at Microsoft Q&A](#)

# UnregisterCMMW function (icm.h)

Article02/22/2024

Dissociates a specified ID value from a given color management module dynamic-link library (CMM DLL).

## Syntax

C++

```
BOOL UnregisterCMMW(
    PCWSTR pMachineName,
    DWORD   cmmID
);
```

## Parameters

pMachineName

Reserved; must currently be set to **NULL**, until non-local registration is supported. This parameter is intended to point to the name of the computer on which a CMM DLLs registration should be removed. A **NULL** pointer indicates the local computer.

cmmID

Specifies the ID value identifying the CMM whose registration is to be removed. This is the signature of the CMM registered with the International Color Consortium (ICC).

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call **GetLastError**.

## Remarks

If the profile for creating a transform later specifies this ID, the Windows default CMM will be used rather than this CMM DLL.

# Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) | [Get help at Microsoft Q&A](#)

# WCS\_PROFILE\_MANAGEMENT\_SCOPE enumeration (icm.h)

Article02/22/2024

Specifies the scope of a profile management operation, such as associating a profile with a device.

## Syntax

C++

```
typedef enum {
    WCS_PROFILE_MANAGEMENT_SCOPE_SYSTEM_WIDE,
    WCS_PROFILE_MANAGEMENT_SCOPE_CURRENT_USER
} WCS_PROFILE_MANAGEMENT_SCOPE;
```

## Constants

[ ] Expand table

`WCS_PROFILE_MANAGEMENT_SCOPE_SYSTEM_WIDE`

Indicates that the profile management operation affects all users.

`WCS_PROFILE_MANAGEMENT_SCOPE_CURRENT_USER`

Indicates that the profile management operation affects only the current user.

## Requirements

[ ] Expand table

Requirement	Value
Minimum supported client	Windows Vista [desktop apps only]
Minimum supported server	Windows Server 2008 [desktop apps only]
Header	icm.h

# Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# WcsAssociateColorProfileWithDevice function (icm.h)

Article02/22/2024

Associates a specified WCS color profile with a specified device.

## ⓘ Note

This API does not support "advanced color" profiles for HDR monitors. Use [ColorProfileAddDisplayAssociation](#) for managing advanced color profiles.

## Syntax

C++

```
BOOL WcsAssociateColorProfileWithDevice(
    WCS_PROFILE_MANAGEMENT_SCOPE scope,
    PCWSTR pProfileName,
    PCWSTR pDeviceName
);
```

## Parameters

scope

A [WCS\\_PROFILE\\_MANAGEMENT\\_SCOPE](#) value that specifies the scope of this profile management operation, which could be system-wide or for the current user.

pProfileName

A pointer to the file name of the profile to associate.

pDeviceName

A pointer to the name of the device with which the profile is to be associated.

## Return value

If this function succeeds, the return value is TRUE.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Remarks

The [WCSAssociateColorProfileWithDevice](#) function fails if the profile has not been installed on the computer using the [InstallColorProfileW](#) function.

If the *profileManagementScope* parameter is `WCS_PROFILE_MANAGEMENT_SCOPE_SYSTEM_WIDE`, the profile association is system-wide and applies to all users. If *profileManagementScope* is `WCS_PROFILE_MANAGEMENT_SCOPE_CURRENT_USER`, the association is only for the current user.

This function is executable in Least-Privileged User Account (LUA) context if *profileManagementScope* is `WCS_PROFILE_MANAGEMENT_SCOPE_CURRENT_USER`. Otherwise, administrative privileges are required.

## Requirements

[Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [Windows Color System schemas and algorithms](#)
- [WcsDisassociateColorProfileFromDevice\\*\\*](#)

# Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# WcsCheckColors function (icm.h)

Article02/22/2024

Determines whether the colors in an array are within the output gamut of a specified WCS color transform.

## Syntax

C++

```
BOOL WcsCheckColors(
    HTRANSFORM    hColorTransform,
    DWORD         nColors,
    DWORD         nInputChannels,
    COLORDATATYPE cdtInput,
    DWORD         cbInput,
    PVOID         pInputData,
    PBYTE         paResult
);
```

## Parameters

`hColorTransform`

A handle to the specified WCS color transform.

`nColors`

The number of elements in the array pointed to by `pInputData` and `paResult`.

`nInputChannels`

The number of channels per element in the array pointed to by `pInputData`.

`cdtInput`

The input COLORDATATYPE color data type.

`cbInput`

The buffer size of `pInputData`.

`pInputData`

A pointer to an array of input colors. Colors in this array correspond to the color space of the source profile. The size of the buffer for this array will be the number of bytes indicated by *cbInput*.

paResult

A pointer to an array of *nColors* bytes that receives the results of the test.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Remarks

If the input and the output color data types are not compatible with the color transform, this function will convert the input color data as required.

This function fails if you use an International Color Consortium (ICC) transform is used.

## Requirements

[\[\] Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [Windows Color System schemas and algorithms](#)

---

# Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# WcsCreateIccProfile function (icm.h)

Article 10/05/2021

Converts a WCS profile into an International Color Consortium (ICC) profile.

## Syntax

C++

```
HPROFILE WcsCreateIccProfile(
    HPROFILE hWcsProfile,
    DWORD     dwOptions
);
```

## Parameters

`hWcsProfile`

A handle to the WCS color profile that is converted. See Remarks.

`dwOptions`

A flag value that specifies the profile conversion options.

By default, the original WCS profiles used for the conversion are embedded in the output ICC profile in a Microsoft private tag, *WcsProfilesTag* (with signature "MS000"). This produces an ICC profile that is compatible with ICC software, yet retains the original WCS profile data available to code designed to parse it.

The possible values of this parameter are as follows. Any bits not defined in this list are reserved and should be set to zero:

Value	Description
<code>WCS_DEFAULT</code>	Specifies that the new ICC profile contains the original WCS profile in a private <i>WcsProfilesTag</i> .
<code>WCS_ICCONLY</code>	Specifies that the new ICC profile does not contain either the <i>WcsProfilesTag</i> or the original WCS profile.

## Return value

If this function succeeds, the return value is the handle of the new color profile.

If this function fails, the return value is **NULL**. For extended error information, call [GetLastError](#).

## Remarks

This function can be used with ASCII or Unicode strings.

The **CloseColorProfile** function should be used to close the returned HPROFILE handle when it is no longer needed.

The DMP, CAMP, and GMMP from the HPROFILE are embedded in a private tag within the created ICC profile.

The ICC profile created using this API will have its profile description tag constructed from the ProfileName elements of the WCS profiles according to the following pattern:  
"Created by Microsoft WCS from DMP:[the DMP ProfileName], CAMP:[the CAMP ProfileName], GMMP:[the GMMP ProfileName]"

When WCS encounters this ICC profile (via [OpenColorProfileW](#) or [WcsOpenColorProfileW](#)) it extracts and uses the WCS profile(s) contained in the *WcsProfilesTag*.

The out-of-gamut information in the gamut tags created in WCS uses the perceptual color distance in CIECAM02, which is the mean square root in CIECAM02 Jab space. The distance in legacy ICC profile gamut tags is the mean square root in CIELAB space. It is recommended that you use the CIECAM02 space when it is available, to provide more perceptually accurate distance metrics.

WCS extracts and uses the original WCS profile by means of an XML profile explicitly associated with a device, or an ICC profile that has a *WcsProfilesTag*.

The *WcsProfilesTag* is a Microsoft private ICC profile tag used in profiles created by **WcsCreateIccProfile** to contain the WCS profiles input to **WcsCreateIccProfile**. This tag conforms to ICC profile requirements for profile tags. The non-XML components of the tag must be in "Big-Endian" byte ordering, which is standard for ICC profiles. Further, the tag data must be aligned on a 4-byte boundary (measured from the start of the ICC profile). The structure of the tag is defined by the **WcsProfilesTagType** below. Note that the XML components of the tag, the WCS profiles contained in the *WcsProfileTag*, are left in their native byte ordering, which may be either little-endian or big-endian since XML parsers correctly process either.

The WcsProfilesTag signature is "MS00". This is the tag signature that will appear in the ICC profiles tag table for the WcsProfilesTag.

The **WcsProfilesTagType** Structure has the following structure:

Byte Offset	Content
0-3	The MS10 type signature.
4-7	Reserved, must be set to 0 (ICC tradition).
8-11	Byte offset from the beginning of the tag to the CDMP data.
12-15	Size of the CDMP data in bytes.
16-19	Byte offset from the beginning of the tag to the CAMP data.
20-23	Size of the CAMP data in bytes.
24-27	Byte offset from the beginning of the tag to the GMMP data.
28-31	Byte offset from the beginning of the tag to the GMMP data.
31-n	A sequence of (element size -32) bytes [where element size is the tag size recorded in the ICC profile tag table entry for this tag.]

These are the WCS XML profiles that were used by **WcsCreateIccProfile** to create this ICC profile. The WCS profiles are ordered: the DMP (required) first, followed by the CAMP (if present), followed by the GMMP (if present).

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- Basic color management concepts
  - Functions
  - Windows Color System schemas and algorithms
- 

## Feedback

Was this page helpful?



[Get help at Microsoft Q&A](#)

# WcsDisassociateColorProfileFromDevice function (icm.h)

Article02/22/2024

Disassociates a specified WCS color profile from a specified device on a computer.

## ⓘ Note

This API does not support "advanced color" profiles for HDR monitors. Use [ColorProfileRemoveDisplayAssociation](#) for managing advanced color profiles.

## Syntax

C++

```
BOOL WcsDisassociateColorProfileFromDevice(
    WCS_PROFILE_MANAGEMENT_SCOPE scope,
    PCWSTR pProfileName,
    PCWSTR pDeviceName
);
```

## Parameters

scope

A [WCS\\_PROFILE\\_MANAGEMENT\\_SCOPE](#) value that specifies the scope of this profile management operation, which could be system-wide or for the current user.

pProfileName

A pointer to the file name of the profile to disassociate.

pDeviceName

A pointer to the name of the device from which to disassociate the profile.

## Return value

If this function succeeds, the return value is TRUE.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Remarks

The WCS color profile should be installed. Moreover, you must use the same *profileManagementScope* value as when the device was associated with the profile. See [WcsAssociateColorProfileWithDevice](#).

If *profileManagementScope* is **WCS\_PROFILE\_MANAGEMENT\_SCOPE\_SYSTEM\_WIDE**, the profile disassociation is systemwide and applies to all users. If *profileManagementScope* is **WCS\_PROFILE\_MANAGEMENT\_SCOPE\_CURRENT\_USER**, the disassociation is only for the current user.

If more than one color profile is associated with a device, WCS uses the last associated profile as the default. For example, if your application sequentially associates three profiles with a device, WCS uses the last profile associated as the default. If your application then calls the [WcsDisassociateColorProfileFromDevice](#) function to disassociate the third profile (which is the default in this example), WCS uses the second profile as the default.

If your application disassociates all profiles from a device, WCS uses the sRGB profile as the default.

If *profileManagementScope* is **WCS\_PROFILE\_MANAGEMENT\_SCOPE\_CURRENT\_USER**, this function is executable in Least-Privileged User Account (LUA) context. Otherwise, administrative privileges are required.

## Requirements

[Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- Basic color management concepts
  - Functions
  - Windows Color System schemas and algorithms
  - WcsAssociateColorProfileWithDevice
- 

## Feedback

Was this page helpful?



Provide product feedback | Get help at Microsoft Q&A

# WcsEnumColorProfiles function (icm.h)

Article10/05/2021

Enumerates color profiles associated with any device, in the specified scope.

## ⓘ Note

This API does not support "advanced color" profiles for HDR monitors. Use [ColorProfileGetDisplayList](#) for managing advanced color profiles.

## Syntax

C++

```
BOOL WcsEnumColorProfiles(
    WCS_PROFILE_MANAGEMENT_SCOPE scope,
    PENUMTYPEW pEnumRecord,
    PBYTE pBuffer,
    DWORD dwSize,
    PDWORD pnProfiles
);
```

## Parameters

scope

A [WCS\\_PROFILE\\_MANAGEMENT\\_SCOPE](#) value specifying the scope of this profile management operation.

pEnumRecord

A pointer to a structure specifying the enumeration criteria.

pBuffer

A pointer to a buffer in which the profile names are to be enumerated. The **WcsEnumColorProfiles** function places, in this buffer, a MULTI\_SZ string that consists of profile names that satisfy the criteria specified in *\*pEnumRecord*.

dwSize

A variable that contains the size, in bytes, of the buffer that is pointed to by *pBuffer*. See Remarks.

#### pnProfiles

An optional pointer to a variable that receives the number of profile names that are copied to the buffer to which *pBuffer* points. Can be **NULL** if this information is not needed.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Remarks

Use the [WcsEnumColorProfilesSize](#) function to retrieve the value for the *dwSize* parameter, which is the size, in bytes, of the buffer pointed to by the *pBuffer* parameter.

If the *profileManagementScope* parameter is **WCS\_PROFILE\_MANAGEMENT\_SCOPE\_SYSTEM\_WIDE**, only system-wide associations of profiles to the device are considered. If *profileManagementScope* is **WCS\_PROFILE\_MANAGEMENT\_SCOPE\_CURRENT\_USER**, only per-user associations for the current user are considered. If [WcsSetUsePerUserProfiles](#) has never been called for this user, or if [WcsSetUsePerUserProfiles](#) was most recently called for this user with its *usePerUserProfiles* parameter set to **FALSE**, then [WCSEnumColorProfiles](#) returns an empty list.

If **WCS\_PROFILE\_MANAGEMENT\_SCOPE\_CURRENT\_USER** (the current user setting) is present, it overrides the system-wide default for the *profileManagementScope* parameter.

This function is executable in Least-Privileged User Account (LUA) context.

## Requirements

Minimum supported client

Windows 2000 Professional [desktop apps only]

Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [Windows Color System schemas and algorithms](#)
- [WCS\\_PROFILE\\_MANAGEMENT\\_SCOPE](#)
- [WcsEnumColorProfilesSize](#)

---

## Feedback

Was this page helpful?



[Get help at Microsoft Q&A](#)

# WcsEnumColorProfilesSize function (icm.h)

Article02/22/2024

Returns the size, in bytes, of the buffer that is required by the [WcsEnumColorProfiles](#) function to enumerate color profiles.

## ⓘ Note

This API does not support "advanced color" profiles for HDR monitors. Use [ColorProfileGetDisplayList](#) for managing advanced color profiles.

## Syntax

C++

```
BOOL WcsEnumColorProfilesSize(
    WCS_PROFILE_MANAGEMENT_SCOPE scope,
    PENUMTYPEW pEnumRecord,
    PDWORD pdwSize
);
```

## Parameters

scope

A [WCS\\_PROFILE\\_MANAGEMENT\\_SCOPE](#) value that specifies the scope of the profile management operation that is performed by this function.

pEnumRecord

A pointer to a structure that specifies the enumeration criteria.

pdwSize

A pointer to a variable that receives the size of the buffer that is required to receive all enumerated profile names. This value is used by the *dwSize* parameter of the [WcsEnumColorProfiles](#) function.

# Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Remarks

This function is executable in Least-Privileged User Account (LUA) context.

## Requirements

[ ] [Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [Windows Color System schemas and algorithms](#)
- [WcsEnumColorProfiles](#)
- [WCS\\_PROFILE\\_MANAGEMENT\\_SCOPE](#)

---

## Feedback

Was this page helpful?



Yes



No

# WcsGetCalibrationManagementState function (icm.h)

Article02/22/2024

Determines whether system management of the display calibration state is enabled.

## Syntax

C++

```
BOOL WcsGetCalibrationManagementState(
    BOOL *pbEnabled
);
```

## Parameters

pbEnabled

**TRUE** if system management of the display calibration state is enabled; otherwise **FALSE**.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**.

## Requirements

  Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib

Requirement	Value
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# WcsGetDefaultColorProfile function (icm.h)

Article10/05/2021

Retrieves the default color profile for a device, or for a device-independent default if the device is not specified.

## ⓘ Note

This API does not support "advanced color" profiles for HDR monitors. Use [ColorProfileGetDisplayDefault](#) for managing advanced color profiles.

## Syntax

C++

```
BOOL WcsGetDefaultColorProfile(
    WCS_PROFILE_MANAGEMENT_SCOPE scope,
    PCWSTR pDeviceName,
    COLORPROFILETYPE cptColorProfileType,
    COLORPROFILESUBTYPE cpstColorProfileSubType,
    DWORD dwProfileID,
    DWORD cbProfileName,
    LPWSTR pProfileName
);
```

## Parameters

scope

A [WCS\\_PROFILE\\_MANAGEMENT\\_SCOPE](#) value specifying the scope of this profile management operation.

pDeviceName

A pointer to the name of the device for which the default color profile is obtained. If **NULL**, a device-independent default profile is obtained.

cptColorProfileType

A [COLORPROFILETYPE](#) value specifying the color profile type.

`cpstColorProfileSubType`

A [COLORPROFILESUBTYPE](#) value specifying the color profile subtype.

`dwProfileID`

The ID of the color space that the color profile represents.

`cbProfileName`

The buffer size, in bytes, of the buffer that is pointed to by *pProfileName*.

`pProfileName`

A pointer to a buffer to receive the name of the color profile. The size of the buffer, in bytes, will be the indicated by *cbProfileName*.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Remarks

Use the [WcsGetDefaultColorProfileSize](#) function to obtain the required size of the buffer that is pointed to by the *pProfileName* parameter.

If **WCS\_PROFILE\_MANAGEMENT\_SCOPE\_CURRENT\_USER** is present, it overrides the system-wide default for *profileManagementScope*.

This function is executable in Least-Privileged User Account (LUA) context.

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h

Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Windows Color System schemas and algorithms](#)
- [Functions](#)
- [COLORPROFILESUBTYPE](#)
- [COLORPROFILETYPE](#)
- [WCS\\_PROFILE\\_MANAGEMENT\\_SCOPE](#)
- [WcsGetDefaultColorProfileSize](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# WcsGetDefaultColorProfileSize function (icm.h)

Article02/22/2024

Returns the size, in bytes, of the default color profile name (including the **NUL** terminator), for a device.

## ⓘ Note

This API does not support "advanced color" profiles for HDR monitors. Use [ColorProfileGetDisplayDefault](#) for managing advanced color profiles.

## Syntax

C++

```
BOOL WcsGetDefaultColorProfileSize(
    WCS_PROFILE_MANAGEMENT_SCOPE scope,
    PCWSTR pDeviceName,
    COLORPROFILETYPE cptColorProfileType,
    COLORPROFILESUBTYPE cpstColorProfileSubType,
    DWORD dwProfileID,
    PDWORD pcbProfileName
);
```

## Parameters

scope

A [WCS\\_PROFILE\\_MANAGEMENT\\_SCOPE](#) value that specifies the scope of this profile management operation.

pDeviceName

A pointer to the name of the device for which the default color profile is to be obtained. If **NULL**, a device-independent default profile will be used.

cptColorProfileType

A [COLORPROFILETYPE](#) value specifying the color profile type.

`cpstColorProfileSubType`

A [COLORPROFILESUBTYPE](#) value specifying the color profile subtype.

`dwProfileID`

The ID of the color space that the color profile represents.

`pcbProfileName`

A pointer to a location that receives the size, in bytes, of the path name of the default color profile, including the **NULL** terminator.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Remarks

Use this function to return the required size of the buffer that is pointed to by the *pProfileName* parameter in the [WcsGetDefaultColorProfile](#) function.

This function is executable in Least-Privileged User Account (LUA) context.

## Requirements

[+] [Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
  - [Windows Color System schemas and algorithms](#)
  - [Functions](#)
  - [COLORPROFILESUBTYPE](#)
  - [COLORPROFILETYPE](#)
  - [WCS\\_PROFILE\\_MANAGEMENT\\_SCOPE](#)
  - [WcsGetDefaultColorProfile](#)
- 

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# WcsGetDefaultRenderingIntent function (icm.h)

Article 02/22/2024

Retrieves the default rendering intent in the specified profile management scope.

## Syntax

C++

```
BOOL WcsGetDefaultRenderingIntent(
    WCS_PROFILE_MANAGEMENT_SCOPE scope,
    PDWORD pdwRenderingIntent
);
```

## Parameters

scope

The profile management scope for this operation, which can be system-wide or the current user only.

pdwRenderingIntent

A pointer to the variable that will hold the rendering intent.

For more information, see [Rendering intents](#).

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Remarks

This function does not revert to the system-wide scope if you do not set the per-user default rendering intent. Instead, it fails, which allows the calling process to distinguish

between the per-user and the system-wide setting. If the per-user rendering intent cannot be retrieved, call this function again with system-wide.

# Requirements

[Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Windows Color System schemas and algorithms](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) | [Get help at Microsoft Q&A](#)

# WcsGetUsePerUserProfiles function (icm.h)

Article02/22/2024

Determines whether the user chose to use a per-user profile association list for the specified device.

## Syntax

C++

```
BOOL WcsGetUsePerUserProfiles(
    LPCWSTR pDeviceName,
    DWORD    dwDeviceClass,
    PBOOL    pUsePerUserProfiles
);
```

## Parameters

pDeviceName

A pointer to a string containing the user-friendly name of the device.

dwDeviceClass

A flag value specifying the class of the device. This parameter must take one of the following values.

 Expand table

Value	Description
CLASS_MONITOR	Specifies a display device.
CLASS_PRINTER	Specifies a printer.
CLASS_SCANNER	Specifies an image-capture device.

pUsePerUserProfiles

A pointer to a location to receive a Boolean value that is **TRUE** if the user chose to use a per-user profile association list for the specified device; otherwise **FALSE**.

# Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Remarks

This function fails if *pDeviceName* points to a device that is not of the class specified by *dwDeviceClass*.

This function is executable in Least-Privileged User Account (LUA) context.

## Requirements

[ ] [Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Windows Color System schemas and algorithms](#)
- [Functions](#)
- [WcsSetUsePerUserProfiles](#)

---

## Feedback

Was this page helpful?

 Yes

 No



# WcsOpenColorProfileA function (icm.h)

Article07/27/2022

Creates a handle to a specified color profile.

## Syntax

C++

```
HPROFILE WcsOpenColorProfileA(  
    PPROFILE pCDMPPProfile,  
    PPROFILE pCAMPProfile,  
    PPROFILE pGMMPProfile,  
    DWORD    dwDesireAccess,  
    DWORD    dwShareMode,  
    DWORD    dwCreationMode,  
    DWORD    dwFlags  
) ;
```

## Parameters

`pCDMPPProfile`

Pointer to a WCS DMP or an ICC color profile structure specifying the profile. You can free the *pCDMPPProfile* pointer after you create the handle. If the profile is ICC and its **dwType** member is set to **DONT\_USE\_EMBEDDED\_WCS\_PROFILES**, **WcsOpenColorProfile** ignores any embedded WCS profile within the ICC profile.

`pCAMPProfile`

A pointer to a profile structure that specifies a WCS color appearance model profile (CAMP). You can free the *pCAMPProfile* pointer after you create the handle. If **NULL**, the default CAMP is used, and the current user setting, **WCS\_PROFILE\_MANAGEMENT\_SCOPE\_CURRENT\_USER**, is used while querying the default CAMP.

`pGMMPProfile`

A pointer to a profile structure that specifies a WCS gamut map model profile (GMMP). You can free the *pGMMPProfile* pointer after you create the handle. If **NULL**, the default GMMP for the default rendering intent is used, and the current user setting,

`WCS_PROFILE_MANAGEMENT_SCOPE_CURRENT_USER`, is used while querying the default GMMP. For a description of rendering intents, see [Rendering Intents](#).

#### `dwDesireAccess`

A flag value that specifies how to access the specified color profile. This parameter must take one of the following values:

<b>Value</b>	<b>Description</b>
<code>PROFILE_READ</code>	Specifies that the color profile opens for read-only access.
<code>PROFILE_READWRITE</code>	Specifies that the color profile opens for both read and write access. The value of this flag is ignored if the profile is a WCS profile.

#### `dwShareMode`

A flag value that specifies actions to take while opening a color profile contained in a file. This parameter must take one of the following values, which are defined in *winnt.h*:

<b>Value</b>	<b>Description</b>
<code>FILE_SHARE_READ</code>	Specifies that you can perform other open (for read access) operations on the profile.
<code>FILE_SHARE_WRITE</code>	Specifies that you can perform other open (for write access) operations on the profile. This flag value is ignored when a WCS profile is opened.

#### `dwCreationMode`

A flag value that specifies the actions to take while opening a color profile if it is contained in a file. This parameter must take one of the following values, which are defined in *winbase.h*:

<b>Value</b>	<b>Description</b>
<code>CREATE_NEW</code>	Specifies that a new profile is created. This function fails if the profile already exists.
<code>CREATE_ALWAYS</code>	Specifies that a new profile is created. If a profile already exists, it is overwritten.
<code>OPEN_EXISTING</code>	Specifies that the profile is opened. This function fails if the profile does not exist.

Value	Description
OPEN_ALWAYS	Specifies that the profile is to be opened if an International Color Consortium (ICC) file exists. If an ICC profile does not exist, WCS creates a new ICC profile. The function will fail for WCS profiles if this flag is set and a WCS profile does not exist.
TRUNCATE_EXISTING	Specifies that the profile is to be opened and truncated to zero bytes. The function fails if the profile does not exist.

### dwFlags

A flag value that specifies whether to use the embedded WCS profile. This parameter has no effect unless *pCDMPProfile* specifies an ICC profile that contains an embedded WCS profile.

This parameter takes one of the following values:

Value	Description
0	Specifies that the embedded WCS profile will be used and the ICC profile specified by <i>pCDMPPProfile</i> will be ignored.
DONT_USE_EMBEDDED_WCS_PROFILES	Specifies that the ICC profile specified by <i>pCDMPPProfile</i> will be used and the embedded WCS profile will be ignored.

## Return value

If this function succeeds, the return value is the handle of the color profile that is opened.

If this function fails, the return value is **NULL**.

## Remarks

This API will take a set of DMP, CAMP, and GMMP and return a WCS profile handle. **NULL** values for GMMP are valid. A **NULL** value for CAMP will be replaced with the default CAMP value.

This API will also accept ICC profiles. Using an ICC profile does not guarantee processing by the WCS CITE engine. The WCS engine will only be used if it is passed at least one WCS profile. Pure ICC workflows will be consistent with legacy behavior.

You can use the handle that this function returns in other color profile management functions.

The *dwCreationMode* flags CREATE\_NEW, CREATE\_ALWAYS, and TRUNCATE\_EXISTING will always return blank ICC HPROFILEs. If other *dwCreationMode* flags are present, the function will determine whether the profile is ICC or WCS XML.

Within the ICC code path, an ICC HPROFILE is returned using the requested sharing, access, and creation flags as specified in the tables above.

Within the WCS path, the *dwCreationMode* flag OPEN\_ALWAYS will fail if the profile does not exist, because WCS profiles cannot be created or edited within the WCS architecture (they must be edited outside of it, using MSXML6). For the same reason, *dwShareMode* flag FILE\_SHARE\_WRITE, and the *dwDesiredAccess* flag PROFILE\_READWRITE are ignored within the WCS path.

Once the handle to the color profile is created, any information used to create that handle can be deleted.

Use the [CloseColorProfile](#) function to close an object handle that is returned by [WcsOpenColorProfile](#).

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Windows Color System schemas and algorithms](#)
- [Functions](#)

# Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# WcsOpenColorProfileW function (icm.h)

Article07/27/2022

Creates a handle to a specified color profile.

## Syntax

C++

```
HPROFILE WcsOpenColorProfileW(
    PPROFILE pCDMPPProfile,
    PPROFILE pCAMPProfile,
    PPROFILE pGMMPProfile,
    DWORD     dwDesireAccess,
    DWORD     dwShareMode,
    DWORD     dwCreationMode,
    DWORD     dwFlags
);
```

## Parameters

pCDMPPProfile

Pointer to a WCS DMP or an ICC color profile structure specifying the profile. You can free the *pCDMPPProfile* pointer after you create the handle. If the profile is ICC and its **dwType** member is set to **DONT\_USE\_EMBEDDED\_WCS\_PROFILES**, **WcsOpenColorProfile** ignores any embedded WCS profile within the ICC profile.

pCAMPProfile

A pointer to a profile structure that specifies a WCS color appearance model profile (CAMP). You can free the *pCAMPProfile* pointer after you create the handle. If **NULL**, the default CAMP is used, and the current user setting, **WCS\_PROFILE\_MANAGEMENT\_SCOPE\_CURRENT\_USER**, is used while querying the default CAMP.

pGMMPProfile

A pointer to a profile structure that specifies a WCS gamut map model profile (GMMP). You can free the *pGMMPProfile* pointer after you create the handle. If **NULL**, the default GMMP for the default rendering intent is used, and the current user setting,

`WCS_PROFILE_MANAGEMENT_SCOPE_CURRENT_USER`, is used while querying the default GMMP. For a description of rendering intents, see [Rendering Intents](#).

#### `dwDesireAccess`

A flag value that specifies how to access the specified color profile. This parameter must take one of the following values:

<b>Value</b>	<b>Description</b>
<code>PROFILE_READ</code>	Specifies that the color profile opens for read-only access.
<code>PROFILE_READWRITE</code>	Specifies that the color profile opens for both read and write access. The value of this flag is ignored if the profile is a WCS profile.

#### `dwShareMode`

A flag value that specifies actions to take while opening a color profile contained in a file. This parameter must take one of the following values, which are defined in *winnt.h*:

<b>Value</b>	<b>Description</b>
<code>FILE_SHARE_READ</code>	Specifies that you can perform other open (for read access) operations on the profile.
<code>FILE_SHARE_WRITE</code>	Specifies that you can perform other open (for write access) operations on the profile. This flag value is ignored when a WCS profile is opened.

#### `dwCreationMode`

A flag value that specifies the actions to take while opening a color profile if it is contained in a file. This parameter must take one of the following values, which are defined in *winbase.h*:

<b>Value</b>	<b>Description</b>
<code>CREATE_NEW</code>	Specifies that a new profile is created. This function fails if the profile already exists.
<code>CREATE_ALWAYS</code>	Specifies that a new profile is created. If a profile already exists, it is overwritten.
<code>OPEN_EXISTING</code>	Specifies that the profile is opened. This function fails if the profile does not exist.

Value	Description
OPEN_ALWAYS	Specifies that the profile is to be opened if an International Color Consortium (ICC) file exists. If an ICC profile does not exist, WCS creates a new ICC profile. The function will fail for WCS profiles if this flag is set and a WCS profile does not exist.
TRUNCATE_EXISTING	Specifies that the profile is to be opened and truncated to zero bytes. The function fails if the profile does not exist.

### dwFlags

A flag value that specifies whether to use the embedded WCS profile. This parameter has no effect unless *pCDMPProfile* specifies an ICC profile that contains an embedded WCS profile.

This parameter takes one of the following values:

Value	Description
0	Specifies that the embedded WCS profile will be used and the ICC profile specified by <i>pCDMPPProfile</i> will be ignored.
DONT_USE_EMBEDDED_WCS_PROFILES	Specifies that the ICC profile specified by <i>pCDMPPProfile</i> will be used and the embedded WCS profile will be ignored.

## Return value

If this function succeeds, the return value is the handle of the color profile that is opened.

If this function fails, the return value is **NULL**.

## Remarks

This API will take a set of DMP, CAMP, and GMMP and return a WCS profile handle. **NULL** values for GMMP are valid. A **NULL** value for CAMP will be replaced with the default CAMP value.

This API will also accept ICC profiles. Using an ICC profile does not guarantee processing by the WCS CITE engine. The WCS engine will only be used if it is passed at least one WCS profile. Pure ICC workflows will be consistent with legacy behavior.

You can use the handle that this function returns in other color profile management functions.

The *dwCreationMode* flags CREATE\_NEW, CREATE\_ALWAYS, and TRUNCATE\_EXISTING will always return blank ICC HPROFILEs. If other *dwCreationMode* flags are present, the function will determine whether the profile is ICC or WCS XML.

Within the ICC code path, an ICC HPROFILE is returned using the requested sharing, access, and creation flags as specified in the tables above.

Within the WCS path, the *dwCreationMode* flag OPEN\_ALWAYS will fail if the profile does not exist, because WCS profiles cannot be created or edited within the WCS architecture (they must be edited outside of it, using MSXML6). For the same reason, *dwShareMode* flag FILE\_SHARE\_WRITE, and the *dwDesiredAccess* flag PROFILE\_READWRITE are ignored within the WCS path.

Once the handle to the color profile is created, any information used to create that handle can be deleted.

Use the [CloseColorProfile](#) function to close an object handle that is returned by [WcsOpenColorProfile](#).

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Windows Color System schemas and algorithms](#)
- [Functions](#)

# Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# WcsSetCalibrationManagementState function (icm.h)

Article02/22/2024

Enables or disables system management of the display calibration state.

## Syntax

C++

```
BOOL WcsSetCalibrationManagementState(
    BOOL bIsEnabled
);
```

## Parameters

bIsEnabled

**TRUE** to enable system management of the display calibration state. **FALSE** to disable system management of the display calibration state.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**.

## Remarks

This function must be called with elevated permissions for it to succeed.

## Requirements

[ ] Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]

Requirement	Value
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) | [Get help at Microsoft Q&A](#)

# WcsSetDefaultColorProfile function (icm.h)

Article 10/05/2021

Sets the default color profile name for the specified profile type in the specified profile management scope.

## ⓘ Note

This API does not support "advanced color" profiles for HDR monitors. Use [ColorProfileSetDisplayDefaultAssociation](#) for managing advanced color profiles.

## Syntax

C++

```
BOOL WcsSetDefaultColorProfile(
    WCS_PROFILE_MANAGEMENT_SCOPE scope,
    PCWSTR pDeviceName,
    COLORPROFILETYPE cptColorProfileType,
    COLORPROFILESUBTYPE cpstColorProfileSubType,
    DWORD dwProfileID,
    LPCWSTR pProfileName
);
```

## Parameters

scope

A [WCS\\_PROFILE\\_MANAGEMENT\\_SCOPE](#) value that specifies the scope of this profile management operation.

pDeviceName

A pointer to the name of the device for which the default color profile is to be set. If **NULL**, a device-independent default profile is used.

cptColorProfileType

A [COLORPROFILETYPE](#) value that specifies the color profile type.

`cpstColorProfileSubType`

A [COLORPROFILESUBTYPE](#) value that specifies the color profile subtype.

`dwProfileID`

The ID of the color space that the color profile represents. This is a custom ID value used to uniquely identify the color space profile within your application.

`pProfileName`

A pointer to a buffer that holds the name of the color profile. See Remarks.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Remarks

If the *pProfileName* parameter is **NULL** and the *profileManagementScope* parameter is **WCS\_PROFILE\_MANAGEMENT\_SCOPE\_CURRENT\_USER**, subsequent calls to **WcsSetDefaultColorProfile** will return the system-wide default profile.

If *profileManagementScope* is **WCS\_PROFILE\_MANAGEMENT\_SCOPE\_CURRENT\_USER**, this function is executable in Least-Privileged User Account (LUA) context. Otherwise, administrative privileges are required. The specified profile must already be installed, but it may be not yet associated with the specified device in the specified profile management scope.

If *profileManagementScope* is **WCS\_PROFILE\_MANAGEMENT\_SCOPE\_CURRENT\_USER**, this function will not correctly function if launched from system context and not a User Account.

When **WcsSetDefaultColorProfile** is called to set a device model profile DMP as the default profile for the RGB or custom working space, only a DMP profile that is of type **RGBVirtualDevice**, **LCD**, or **CRT** is valid; all others are invalid.

When **WcsSetDefaultColorProfile** is called to set an International Color Consortium (ICC) profile as the default profile for the RGB or custom working space, only an ICC profile with class "spac" or "disp", and "RGB" color space is valid; all others are invalid.

See notes on valid profile type/subtype combinations.

# Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Windows Color System schemas and algorithms](#)
- [Functions](#)
- [COLORPROFILESUBTYPE](#)
- [COLORPROFILETYPE](#)
- [WCS\\_PROFILE\\_MANAGEMENT\\_SCOPE](#)
- [WcsGetDefaultColorProfileSize](#)

---

## Feedback

Was this page helpful?



[Get help at Microsoft Q&A](#)

# WcsSetDefaultRenderingIntent function (icm.h)

Article02/22/2024

Sets the default rendering intent in the specified profile management scope.

## Syntax

C++

```
BOOL WcsSetDefaultRenderingIntent(
    WCS_PROFILE_MANAGEMENT_SCOPE scope,
    DWORD dwRenderingIntent
);
```

## Parameters

scope

The profile management scope for this operation, which can be system-wide or the current user only.

dwRenderingIntent

The rendering intent. It can be set to one of the following values:

INTENT\_PERCEPTUAL

INTENT\_RELATIVE\_COLORIMETRIC

INTENT\_SATURATION

INTENT\_ABSOLUTE\_COLORIMETRIC

DWORD\_MAX

If *dwRenderingIntent* is DWORD\_MAX and *scope* is WCS\_PROFILE\_MANAGEMENT\_SCOPE\_CURRENT\_USER, the default rendering intent for the current user reverts to the system-wide default.

For more information, see [Rendering intents](#).

# Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Windows Color System schemas and algorithms](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# WcsSetUsePerUserProfiles function (icm.h)

Article 10/05/2021

Enables a user to specify whether or not to use a per-user profile association list for the specified device.

## Syntax

C++

```
BOOL WcsSetUsePerUserProfiles(
    LPCWSTR pDeviceName,
    DWORD    dwDeviceClass,
    BOOL     usePerUserProfiles
);
```

## Parameters

pDeviceName

A pointer to a string that contains the user-friendly name of the device.

dwDeviceClass

A flag value that specifies the class of the device. This parameter must take one of the following values:

Value	Description
CLASS_MONITOR	Specifies a display device.
CLASS_PRINTER	Specifies a printer.
CLASS_SCANNER	Specifies an image capture device.

usePerUserProfiles

A Boolean value that is **TRUE** if the user wants to use a per-user profile association list for the specified device; otherwise **FALSE**.

# Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call [GetLastError](#).

## Remarks

If *usePerUserProfiles* is **TRUE**, and the user is not already using a per-user profile association list for *pDeviceName*, then the per-user profile association list is initialized by making a copy of the system-wide profile association list for the same device. From then on, changes to the system-wide list are not included in the per-user list.

This function is executable in Least-Privileged User Account (LUA) context.

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- [Basic color management concepts](#)
- [Windows Color System schemas and algorithms](#)
- [Functions](#)
- [WcsGetUsePerUserProfiles](#)

---

## Feedback

Was this page helpful?



Yes



No

Get help at Microsoft Q&A

# WcsTranslateColors function (icm.h)

Article02/22/2024

Translates an array of colors from the source color space to the destination color space as defined by a color transform.

## Syntax

C++

```
BOOL WcsTranslateColors(
    HTRANSFORM    hColorTransform,
    DWORD         nColors,
    DWORD         nInputChannels,
    COLORDATATYPE cdtInput,
    DWORD         cbInput,
    PVOID         pInputData,
    DWORD         nOutputChannels,
    COLORDATATYPE cdtOutput,
    DWORD         cbOutput,
    PVOID         pOutputData
);
```

## Parameters

`hColorTransform`

A handle for the WCS color transform.

`nColors`

The number of elements in the array to which `pInputData` and `pOutputData` point.

`nInputChannels`

The number of channels per element in the array to which `pInputData` points.

`cdtInput`

The input **COLORDATATYPE** color data type.

`cbInput`

The buffer size, in bytes, of `pInputData`.

`pInputData`

A pointer to an array of input colors. The size of the buffer for this array, in bytes, is the **DWORD** value of *cbInput*.

`nOutputChannels`

The number of channels per element in the array to which *pOutputData* points.

`cdtOutput`

The **COLORDATATYPE** output that specified the color data type.

`cbOutput`

The buffer size, in bytes, of *pOutputData*.

`pOutputData`

A pointer to an array of colors that receives the results of the color translation. The size of the buffer for this array, in bytes, is the **DWORD** value of *cbOutput*.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**. For extended error information, call **GetLastError**.

## Remarks

If the input and the output color data types are not compatible with the color transform, this function fails. This function will fail if an ICC transform is used.

## Requirements

[+] [Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]

Requirement	Value
Header	icm.h
Library	Mscms.lib
DLL	Mscms.dll

## See also

- Basic color management concepts
- Windows Color System schemas and algorithms
- Functions

---

## Feedback

Was this page helpful?



[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# XYZCOLOR structure (icm.h)

Article02/22/2024

TBD

## Syntax

C++

```
struct XYZCOLOR {
    WORD X;
    WORD Y;
    WORD Z;
};
```

## Members

X

TBD

Y

TBD

Z

TBD

## Requirements

[+] Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h

# Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# YxyCOLOR structure (icm.h)

Article02/22/2024

TBD

## Syntax

C++

```
struct YxyCOLOR {
    WORD Y;
    WORD x;
    WORD y;
};
```

## Members

Y

TBD

x

TBD

y

TBD

## Requirements

[+] Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	icm.h

# Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# wcsplugin.h header

Article01/24/2023

This header is used by Windows Color System. For more information, see:

- [Windows Color System](#)

wcsplugin.h contains the following programming interfaces:

## Interfaces

### [IDeviceModelPlugIn](#)

Describes the methods that are defined for the IDeviceModelPlugIn Component Object Model (COM) interface.

### [IGamutMapModelPlugIn](#)

Describes the methods that are defined for the IGamutMapModelPlugIn Component Object Model (COM) interface.

## Structures

### [BlackInformation](#)

Contains information for device models that have a black color channel.

### [GamutBoundaryDescription](#)

Defines a gamut boundary.

### [GamutShell](#)

Contains information that defines a gamut shell, which is represented by a list of indexed triangles. The vertex buffer contains the vertices data.

### [GamutShellTriangle](#)

Contains three vertex indices for accessing a vertex buffer.

## [JabColorF](#)

JabColorF (wcsplugin.h) is a structure.

## [JChColorF](#)

JChColorF (wcsplugin.h) is a structure.

## [PrimaryJabColors](#)

This structure contains eight primary colors in Jab coordinates.

## [PrimaryXYZColors](#)

This structure contains eight primary colors in XYZ coordinates.

## [XYZColorF](#)

XYZColorF (wcsplugin.h) is a structure.

---

## Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# BlackInformation structure (wcsplugin.h)

Article 10/05/2021

Contains information for device models that have a black color channel.

## Syntax

C++

```
typedef struct _BlackInformation {
    BOOL   fBlackOnly;
    FLOAT  blackWeight;
} BlackInformation;
```

## Members

fBlackOnly

blackWeight

A value between 0.0 and 1.0 that indicates the relative amount of black to use in the output. A value of 0.0 means that no black is used; a value of 1.0 means that the maximum amount of black is used.

## Remarks

If the source device does not support a black channel, then WCS sets **bBlackOnly** to FALSE.

If **bBlackOnly** is TRUE, then WCS generates an output device control value where, at most, the black channel will be non-zero. This only happens if the **BlackPreservation** flag was set in WCS. Note that in such cases, the device model may not be providing the closest colorimetric match to the supplied value.

Black preservation is only performed when both the source and destination devices support a black channel. If black is being preserved with these devices, then for each source device control value, where all channels other than the black channel are zero, the **bBlackOnly** flag is TRUE. Note that this means that a value where all channels are zero will also set **bBlackOnly** to TRUE.

**blackWeight** gives us information about the device control values used in the source device.

- For source devices with a black channel, **blackWeight** is set to the black value.
- For source devices without a black channel, the black weight is computed using a combination of *color purity* and *relative lightness*. *Color purity* is defined as  $(\text{maxColorant} - \text{minColorant})/\text{maxColorant}$

*Relative lightness* is defined as  $(\text{the lightness of the color in appearance space} - \text{minimum lightness of destination device}) / (\text{maximum lightness of destination device} - \text{minimum lightness of destination device})$

For RGB devices,  $\text{blackWeight} = (1 - \text{colorPurity}) * (1 - \text{relativeLightness})$

For CMYK devices,  $\text{blackWeight} = \text{colorPurity} * (1 - \text{relativeLightness})$

WCS is responsible for initializing the **BlackInformation** structure.

If **bBlackOnly** is **FALSE**, then the baseline device models for devices with a black channel will use the **blackWeight** to guide the creation of a colorimetrically appropriate output pixel value. For CMYK devices, **blackWeight** provides WCS's initial estimation of a K value and it searches for C, M, and Y values that will lead to the correct colorimetry. If it does not find a match, it adjusts the K value and searches again.

You can set plug-ins to either support or ignore the **BlackInformation**.

## Requirements

Minimum supported client	Windows Vista [desktop apps only]
Minimum supported server	Windows Server 2008 [desktop apps only]
Header	wcsplugin.h

## See also

- [Basic color management concepts](#)
- [Structures](#)

---

## Feedback

Was this page helpful?

 Yes

 No

Get help at Microsoft Q&A

# GamutBoundaryDescription structure (wcsplugin.h)

Article 02/22/2024

Defines a gamut boundary.

## Syntax

C++

```
typedef struct _GamutBoundaryDescription {
    PrimaryJabColors *pPrimaries;
    UINT             cNeutralSamples;
    JabColorF        *pNeutralSamples;
    GamutShell       *pReferenceShell;
    GamutShell       *pPlausibleShell;
    GamutShell       *pPossibleShell;
} GamutBoundaryDescription;
```

## Members

pPrimaries

cNeutralSamples

The number of neutral samples.

pNeutralSamples

pReferenceShell

pPlausibleShell

pPossibleShell

## Requirements

[+] Expand table

Requirement	Value
Minimum supported client	Windows Vista [desktop apps only]
Minimum supported server	Windows Server 2008 [desktop apps only]
Header	wcsplugin.h

## See also

- [Basic color management concepts](#)
- [Structures](#)
- [Windows Color System schemas and algorithms](#)

---

## Feedback

Was this page helpful?



[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# GamutShell structure (wcsplugin.h)

Article 02/22/2024

Contains information that defines a gamut shell, which is represented by a list of indexed triangles. The vertex buffer contains the vertices data.

## Syntax

C++

```
typedef struct _GamutShell {
    FLOAT          JMin;
    FLOAT          JMax;
    UINT           cVertices;
    UINT           cTriangles;
    JabColorF      *pVertices;
    GamutShellTriangle *pTriangles;
} GamutShell;
```

## Members

JMin

The minimum lightness of the shell.

JMax

The maximum lightness of the shell.

cVertices

The number of vertices in the shell.

cTriangles

The number of triangles in the shell.

pVertices

pTriangles

## Requirements

Requirement	Value
Minimum supported client	Windows Vista [desktop apps only]
Minimum supported server	Windows Server 2008 [desktop apps only]
Header	wcsplugin.h

## See also

- [Basic color management concepts](#)
- [Structures](#)
- [Windows Color System schemas and algorithms](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) | [Get help at Microsoft Q&A](#)

# GamutShellTriangle structure (wcsplugin.h)

Article02/22/2024

Contains three vertex indices for accessing a vertex buffer.

## Syntax

C++

```
typedef struct _GamutShellTriangle {
    UINT aVertexIndex[3];
} GamutShellTriangle;
```

## Members

aVertexIndex[3]

An array of three vertex indices that are used for accessing a vertex buffer.

## Requirements

[+] Expand table

Requirement	Value
Minimum supported client	Windows Vista [desktop apps only]
Minimum supported server	Windows Server 2008 [desktop apps only]
Header	wcsplugin.h

## See also

- [Basic color management concepts](#)
- [Structures](#)
- [Windows Color System schemas and algorithms](#)

# Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# IDeviceModelPlugIn interface (wcsplugin.h)

Article 02/16/2023

Describes the methods that are defined for the **IDeviceModelPlugIn** Component Object Model (COM) interface.

- [ColorimetricToDeviceColors](#)
- [DeviceToColorimetricColors](#)
- [DeviceToColorimetricColorsWithBlack](#)
- [GetGamutBoundaryMesh](#)
- [GetGamutBoundaryMeshSize](#)
- [GetNeutralAxis](#)
- [GetNeutralAxisSize](#)
- [GetNumChannels](#)
- [GetPrimarySamples](#)
- [Initialize](#)
- [SetTransformDeviceModelInfo](#)

## Inheritance

The **IDeviceModelPlugIn** interface inherits from the **IUnknown** interface.

## Methods

The **IDeviceModelPlugIn** interface has these methods.

<a href="#">IDeviceModelPlugIn::ColorimetricToDeviceColors</a>  Returns the appropriate XYZ colors in response to the specified number of colors, channels, device colors and the proprietary plug-in algorithms. ( <b>IDeviceModelPlugIn::ColorimetricToDeviceColors</b> )
<a href="#">IDeviceModelPlugIn::ColorimetricToDeviceColorsWithBlack</a>  Returns the appropriate device colors in response to the incoming number of colors, channels, black information, Commission Internationale l'Eclairge XYZ (CIEXYZ) colors and the proprietary plug-in algorithms.

## [IDeviceModelPlugIn::DeviceToColorimetricColors](#)

Returns the appropriate XYZ colors in response to the specified number of colors, channels, device colors and the proprietary plug-in algorithms. (IDeviceModelPlugIn.DeviceToColorimetricColors)

## [IDeviceModelPlugIn::GetGamutBoundaryMesh](#)

Returns the triangular mesh from the plug-in. This function is used to compute the GamutBoundaryDescription.

## [IDeviceModelPlugIn::GetGamutBoundaryMeshSize](#)

Returns the required data structure sizes for the GetGamutBoundaryMesh function.

## [IDeviceModelPlugIn::GetNeutralAxis](#)

The IDeviceModelPlugIn::GetNeutralAxis return the XYZ colorimetry of sample points along the device's neutral axis.

## [IDeviceModelPlugIn::GetNeutralAxisSize](#)

The IDeviceModelPlugIn::GetNeutralAxisSize function returns the number of data points along the neutral axis that are returned by the GetNeutralAxis function.

## [IDeviceModelPlugIn::GetNumChannels](#)

Returns the number of device channels in the parameter pNumChannels.

## [IDeviceModelPlugIn::GetPrimarySamples](#)

Returns the measurement color for the primary sample.

## [IDeviceModelPlugIn::Initialize](#)

Takes a pointer to a Stream that contains the whole device model plug-in as input, and initializes any internal parameters required by the plug-in.

## [IDeviceModelPlugIn::SetTransformDeviceModellInfo](#)

Provides the plug-in with parameters to determine where in the transform sequence the specific plug-in occurs.

# Requirements

Target Platform
Windows

## Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# IDeviceModelPlugIn::ColorimetricToDeviceColors method (wcsplugin.h)

Article 02/22/2024

Returns the appropriate XYZ colors in response to the specified number of colors, channels, device colors and the proprietary plug-in algorithms.

## Syntax

C++

```
HRESULT ColorimetricToDeviceColors(
    [in]  UINT          cColors,
    [in]  UINT          cChannels,
    [in]  const XYZColorF *pXYZColors,
    [out] FLOAT         *pDeviceValues
);
```

## Parameters

[in] cColors

The number of colors in the *pXYZColors* and *pDeviceValues* arrays.

[in] cChannels

The number of color channels in the *pDeviceValues* arrays.

[in] pXYZColors

The pointer to the array of incoming XYZColors to convert to device colors.

[out] pDeviceValues

The pointer to an array that contains the resulting outgoing device color values.

## Return value

If this function succeeds, the return value is S\_OK.

If this function fails, the return value is E\_FAIL.

# Remarks

If *cColors* or *cChannels* is zero, the return value is E\_FAIL. If there are invalid or out of gamut colors (which may occur if there has been prior processing of the gamut map model), then the return value is E\_FAIL.

# Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows Vista [desktop apps only]
Minimum supported server	Windows Server 2008 [desktop apps only]
Target Platform	Windows
Header	wcsplugin.h

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [IDeviceModelPlugIn](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# IDeviceModelPlugIn::ColorimetricToDeviceColorsWithBlack method (wcsplugin.h)

Article 02/22/2024

Returns the appropriate device colors in response to the incoming number of colors, channels, black information, Commission Internationale l'Eclairge XYZ (CIEXYZ) colors and the proprietary plug-in algorithms.

## Syntax

C++

```
HRESULT ColorimetricToDeviceColorsWithBlack(
    [in]  UINT           cColors,
    [in]  UINT           cChannels,
    [out] const XYZColorF *pXYZColors,
    [in]  const BlackInformation *pBlackInformation,
    [in]  FLOAT          *pDeviceValues
);
```

## Parameters

[in] cColors

The number of colors in the *pXYZColors* and *pDeviceValues* arrays.

[in] cChannels

The number of color channels in the *pDeviceValues* arrays.

[out] pXYZColors

A pointer to the array of outgoing [XYZColorF](#) structures.

[in] pBlackInformation

A pointer to the [BlackInformation](#).

[in] pDeviceValues

A pointer to the array of incoming device colors that are to be converted to `XYZColorF` structures.

## Return value

If this function succeeds, the return value is `S_OK`.

If this function fails, the return value is `E_FAIL`. For extended error information, call `GetLastError`.

## Remarks

If `cColors` or `cChannels` is zero, the return value is `E_FAIL`.

## Requirements

[ ] [Expand table](#)

Requirement	Value
Minimum supported client	Windows Vista [desktop apps only]
Minimum supported server	Windows Server 2008 [desktop apps only]
Target Platform	Windows
Header	<code>wcsplugin.h</code>

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [IDeviceModelPlugIn](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# IDeviceModelPlugIn::DeviceToColorimetricColors method (wcsplugin.h)

Article 02/22/2024

Returns the appropriate XYZ colors in response to the specified number of colors, channels, device colors and the proprietary plug-in algorithms.

## Syntax

C++

```
HRESULT DeviceToColorimetricColors(  
    [in]  UINT      cColors,  
    [in]  UINT      cChannels,  
    [in]  const FLOAT *pDeviceValues,  
    [out] XYZColorF *pXYZColors  
) ;
```

## Parameters

[in] cColors

The number of colors in the *pXYZColors* and *pDeviceValues* arrays.

[in] cChannels

The number of color channels in the *pDeviceValues* arrays.

[in] pDeviceValues

A pointer to the array of outgoing XYZColors.

[out] pXYZColors

A pointer to the array of incoming device colors to convert to XYZColors.

## Return value

If this function succeeds, the return value is S\_OK.

If this function fails, the return value is E\_FAIL. For extended error information, call [GetLastError](#).

## Remarks

If *cColors* or *cChannels* is zero, the return value is E\_FAIL.

## Requirements

 [Expand table](#)

Requirement	Value
Minimum supported client	Windows Vista [desktop apps only]
Minimum supported server	Windows Server 2008 [desktop apps only]
Target Platform	Windows
Header	wcsplugin.h

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [IDeviceModelPlugIn](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# IDeviceModelPlugIn::GetGamutBoundaryMesh method (wcsplugin.h)

Article 10/13/2021

Returns the triangular mesh from the plug-in. This function is used to compute the GamutBoundaryDescription.

## Syntax

C++

```
HRESULT GetGamutBoundaryMesh(
    [in]  UINT          cChannels,
    [in]  UINT          cVertices,
    [in]  UINT          cTriangles,
    [out] FLOAT         *pVertices,
    [out] GamutShellTriangle *pTriangles
);
```

## Parameters

[in] cChannels

The number of channels.

[in] cVertices

The number of vertices.

[in] cTriangles

The number of triangles.

[out] pVertices

A pointer to the array of vertices in the plug-in model gamut shell.

[out] pTriangles

A pointer to the array of triangles in the plug-in model gamut shell.

## Return value

If this function succeeds, the return value is S\_OK.

If this function fails, the return value is E\_FAIL.

## Remarks

This function is called by the [CreateMultiProfileTransform](#) function.

## Requirements

Minimum supported client	Windows Vista [desktop apps only]
Minimum supported server	Windows Server 2008 [desktop apps only]
Target Platform	Windows
Header	wcsplugin.h

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [IDeviceModelPlugIn](#)

---

## Feedback

Was this page helpful?



[Get help at Microsoft Q&A](#)

# IDeviceModelPlugIn::GetGamutBoundaryMeshSize method (wcsplugin.h)

Article 02/22/2024

Returns the required data structure sizes for the [GetGamutBoundaryMesh](#) function.

## Syntax

C++

```
HRESULT GetGamutBoundaryMeshSize(
    [out] UINT *pNumVertices,
    [out] UINT *pNumTriangles
);
```

## Parameters

`[out] pNumVertices`

The required number of vertices.

`[out] pNumTriangles`

The required number of triangles.

## Return value

If this function succeeds, the return value is S\_OK.

If the plug-in device model wants WCS to compute its gamut boundary mesh, the return value is S\_FALSE.

If this function fails, the return value is E\_FAIL.

## Remarks

This function is called by the [CreateMultiProfileTransform](#) function.

## Requirements

Requirement	Value
Minimum supported client	Windows Vista [desktop apps only]
Minimum supported server	Windows Server 2008 [desktop apps only]
Target Platform	Windows
Header	wcsplugin.h

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [IDeviceModelPlugIn](#)

---

## Feedback

Was this page helpful?

[!\[\]\(f7106cf6d5b86e1308947762bf489a13\_img.jpg\) Yes](#)[!\[\]\(8b085b3b140a73e04e19ec44f8d1069c\_img.jpg\) No](#)

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# IDeviceModelPlugIn::GetNeutralAxis method (wcsplugin.h)

Article 10/13/2021

The [IDeviceModelPlugIn::GetNeutralAxis](#) return the XYZ colorimetry of sample points along the device's neutral axis.

## Syntax

C++

```
HRESULT GetNeutralAxis(
    [in]  UINT      cColors,
    [out] XYZColorF *pXYZColors
);
```

## Parameters

[in] cColors

The number of points that are returned.

[out] pXYZColors

A pointer to an array of [XYZColorF](#) structures.

## Return value

If this function succeeds, the return value is S\_OK.

If this function fails, the return value is E\_FAIL.

## Remarks

You should define "neutral axis" in a way that is appropriate for your device. Usually, it is points made by the device's gray values. This might be R=G=B, or C=M=Y=0 and any value of K. For some devices, the most pleasing gray may be one that uses a different combination of colorants, such as M=Y=0 and C=K. The plug-in is responsible for

determining the colorimetry of a sampling of the neutral axis values and returning them. The sampling may be as sparse as two points (white and black) or as dense as desired.

There is no requirement that the samples be uniformly spaced in any color space.

## Requirements

Minimum supported client	Windows Vista [desktop apps only]
Minimum supported server	Windows Server 2008 [desktop apps only]
Target Platform	Windows
Header	wcsplugin.h

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [IDeviceModelPlugIn](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# IDeviceModelPlugIn::GetNeutralAxisSize method (wcsplugin.h)

Article02/22/2024

The [IDeviceModelPlugIn::GetNeutralAxisSize](#) function returns the number of data points along the neutral axis that are returned by the [GetNeutralAxis](#) function. It is provided so that a Color Management Module (CMM) can allocate an appropriately sized buffer.

## Syntax

C++

```
HRESULT GetNeutralAxisSize(  
    [out] UINT *pcColors  
) ;
```

## Parameters

[out] pcColors

The number of points that will be returned by a call to [GetNeutralAxis](#). Minimum is 2 (black and white).

## Return value

If this function succeeds, the return value is S\_OK.

If this function fails, the return value is E\_FAIL.

## Requirements

[] [Expand table](#)

Requirement	Value
Minimum supported client	Windows Vista [desktop apps only]
Minimum supported server	Windows Server 2008 [desktop apps only]

Requirement	Value
Target Platform	Windows
Header	wcsplugin.h

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [IDeviceModelPlugIn](#)

---

## Feedback

Was this page helpful?



[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# IDeviceModelPlugIn::GetNumChannels method (wcsplugin.h)

Article 02/22/2024

Returns the number of device channels in the parameter *pNumChannels*.

## Syntax

C++

```
HRESULT GetNumChannels(  
    [out] UINT *pNumChannels  
);
```

## Parameters

[out] *pNumChannels*

A pointer to an unsigned integer representing the number of color channels for your device.

## Return value

If this function succeeds, the return value is S\_OK.

If this function fails, the return value is E\_FAIL.

## Requirements

[+] Expand table

Requirement	Value
Minimum supported client	Windows Vista [desktop apps only]
Minimum supported server	Windows Server 2008 [desktop apps only]
Target Platform	Windows
Header	wcsplugin.h

## See also

- Basic color management concepts
  - Functions
  - IDeviceModelPlugIn
- 

## Feedback

Was this page helpful?



[Provide product feedback](#) | [Get help at Microsoft Q&A](#)

# IDeviceModelPlugIn::GetPrimarySamples method (wcsplugin.h)

Article02/22/2024

Returns the measurement color for the primary sample.

## Syntax

C++

```
HRESULT GetPrimarySamples(
    [out] PrimaryXYZColors *pPrimaryColor
);
```

## Parameters

[out] pPrimaryColor

The primary color type, which is determined by using the hue circle order. If the plugin device model does not natively support primaries for red, yellow, green, cyan, blue, magenta, black and white, it must still return virtual primary data.

## Return value

If this function succeeds, the return value is S\_OK.

If this function fails, the return value is E\_FAIL.

## Requirements

[ ] Expand table

Requirement	Value
Minimum supported client	Windows Vista [desktop apps only]
Minimum supported server	Windows Server 2008 [desktop apps only]
Target Platform	Windows

Requirement	Value
Header	wcsplugin.h

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [IDeviceModelPlugIn](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) | [Get help at Microsoft Q&A](#)

# IDeviceModelPlugIn::Initialize method (wcsplugin.h)

Article02/22/2024

Takes a pointer to a Stream that contains the whole device model plug-in as input, and initializes any internal parameters required by the plug-in.

## Syntax

C++

```
HRESULT Initialize(  
    [in] BSTR bstrXml,  
    [in] UINT cNumModels,  
    [in] UINT iModelPosition  
);
```

## Parameters

[in] `bstrXml`

A string that contains the BSTR XML device model plug-in profile. This parameter stores data as little-endian Unicode XML; however, it may have no leading bytes to tag it as such. Also, the encoding keyword in the XML may not reflect that this is formatted as little-endian Unicode. Furthermore, due to the action of the MSXML engine, the BSTR XML file is processed and might not have exactly the same contents as the original XML file.

[in] `cNumModels`

The number of total models in the transform sequence.

[in] `iModelPosition`

The one-based model position of the other device model in the workflow of `uiNumModels` as provided in the `Initialize` function.

## Return value

If this function succeeds, the return value is `S_OK`.

If this function fails, the return value is E\_FAIL.

## Remarks

If this function is called more than once, subsequent calls release any allocated memory and reinitialize according to the new *bstrXml* parameter.

## Requirements

  Expand table

Requirement	Value
Minimum supported client	Windows Vista [desktop apps only]
Minimum supported server	Windows Server 2008 [desktop apps only]
Target Platform	Windows
Header	wcsplugin.h

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [IDeviceModelPlugIn](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# IDeviceModelPlugIn::SetTransformDeviceModelInfo method (wcsplugin.h)

Article 02/22/2024

Provides the plug-in with parameters to determine where in the transform sequence the specific plug-in occurs.

## Syntax

C++

```
HRESULT SetTransformDeviceModelInfo(
    [in] UINT             iModelPosition,
    [in] IDeviceModelPlugIn *pIDeviceModelOther
);
```

## Parameters

[in] iModelPosition

The one-based model position of the other device model in the workflow of *uiNumModels*, as provided in the [Initialize](#) function.

[in] pIDeviceModelOther

A pointer to a **IDeviceModelPlugIn** interface that contains the other device model in the transform sequence.

## Return value

If this function succeeds, the return value is S\_OK.

If this function fails, the return value is E\_FAIL.

## Remarks

This function is called by the [CreateMultiProfileTransform](#) function, which is responsible for calling **AddRef** and **Release** as appropriate. The function enables plug-in device

models to exchange information in a proprietary manner by accessing proprietary plug-in interface functions.

This function will fail if the other device model is a baseline device model, because such models are not plugins and thus inter-plugin communication is not supported.

## Requirements

[ ] Expand table

Requirement	Value
Minimum supported client	Windows Vista [desktop apps only]
Minimum supported server	Windows Server 2008 [desktop apps only]
Target Platform	Windows
Header	wcsplugin.h

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [IDeviceModelPlugin](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# **IGamutMapModelPlugIn interface (wcsplugin.h)**

Article 02/16/2023

Describes the methods that are defined for the **IGamutMapModelPlugIn** Component Object Model (COM) interface.

- [IGamutMapModelPlugIn::Initialize](#)
- [IGamutMapModelPlugIn::SourceToDestinationAppearanceColors](#)

## **Inheritance**

The **IGamutMapModelPlugIn** interface inherits from the **IUnknown** interface.

## **Methods**

The **IGamutMapModelPlugIn** interface has these methods.

<a href="#">IGamutMapModelPlugIn::Initialize</a>
Initializes a gamut map model profile (GMMP) by using the specified source and destination gamut boundary descriptions and optional source and destination device model plug-ins.
<a href="#">IGamutMapModelPlugIn::SourceToDestinationAppearanceColors</a>
Returns the appropriate gamut-mapped appearance colors in response to the specified number of colors and the CIEJCh colors.

## **Requirements**

Target Platform	Windows
Header	wcsplugin.h

## **Feedback**

Was this page helpful?

 Yes

 No

Get help at Microsoft Q&A

# IGamutMapModelPlugIn::Initialize method (wcsplugin.h)

Article 02/22/2024

Initializes a gamut map model profile (GMMP) by using the specified source and destination gamut boundary descriptions and optional source and destination device model plug-ins.

## Syntax

C++

```
HRESULT Initialize(  
    [in] BSTR                 bstrXml,  
    [in] IDeviceModelPlugIn   *pSrcPlugIn,  
    [in] IDeviceModelPlugIn   *pDestPlugIn,  
    [in] GamutBoundaryDescription *pSrcGBD,  
    [in] GamutBoundaryDescription *pDestGBD  
);
```

## Parameters

[in] bstrXml

A string that contains the BSTR XML GMMP profile. This is little-endian Unicode XML, but without the leading bytes to tag it as such. Also, the encoding keyword in the XML may not reflect this format.

[in] pSrcPlugIn

A pointer to a source [IDeviceModelPlugIn](#). If **NULL**, it indicates the source device model profile is not a plug-in profile.

[in] pDestPlugIn

A pointer to a destination [IDeviceModelPlugIn](#). If **NULL**, it indicates the destination device model profile is not a plug-in profile.

[in] pSrcGBD

A pointer to a source [GamutBoundaryDescription](#).

[in] pDestGBD

A pointer to a destination [GamutBoundaryDescription](#).

## Return value

If this function succeeds, the return value is S\_OK.

If this function fails, the return value is E\_FAIL.

## Requirements

[\[+\] Expand table](#)

Requirement	Value
Minimum supported client	Windows Vista [desktop apps only]
Minimum supported server	Windows Server 2008 [desktop apps only]
Target Platform	Windows
Header	wcsplugin.h

## See also

- [Basic color management concepts](#)
- [IGamutMapModelPlugIn](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# IGamutMapModelPlugIn::SourceToDestinationAppearanceColors method (wcsplugin.h)

Article 02/22/2024

Returns the appropriate gamut-mapped appearance colors in response to the specified number of colors and the [CIEJCh](#) colors.

## Syntax

C++

```
HRESULT SourceToDestinationAppearanceColors(
    [in]  UINT          cColors,
    [in]  const JChColorF *pInputColors,
    [out] JChColorF      *pOutputColors
);
```

## Parameters

[in] cColors

The number of colors in the *pXYZColors* and *pDeviceValues* arrays.

[in] pInputColors

A pointer to the array of incoming colors to be gamut mapped.

[out] pOutputColors

A pointer to the array of outgoing colors.

## Return value

If this function succeeds, the return value is S\_OK.

If this function fails, the return value is E\_FAIL.

## Requirements

Requirement	Value
Minimum supported client	Windows Vista [desktop apps only]
Minimum supported server	Windows Server 2008 [desktop apps only]
Target Platform	Windows
Header	wcsplugin.h

## See also

- [Basic color management concepts](#)
- [IGamutMapModelPlugIn](#)

---

## Feedback

Was this page helpful?

[!\[\]\(047495e104fa1cc2e267db807f964b64\_img.jpg\) Yes](#)[!\[\]\(8822e5621cf2b5336dce58587073a342\_img.jpg\) No](#)

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# JabColorF structure (wcsplugin.h)

Article 02/22/2024

TBD

## Syntax

C++

```
typedef struct _JabColorF {
    FLOAT J;
    FLOAT a;
    FLOAT b;
} JabColorF;
```

## Members

J

TBD

a

TBD

b

TBD

## Requirements

[+] Expand table

Requirement	Value
Header	wcsplugin.h

## Feedback

Was this page helpful?

 Yes

 No

Provide product feedback  | Get help at Microsoft Q&A

# JChColorF structure (wcsplugin.h)

Article 02/22/2024

TBD

## Syntax

C++

```
typedef struct _JChColorF {
    FLOAT J;
    FLOAT C;
    FLOAT h;
} JChColorF;
```

## Members

J

TBD

C

TBD

h

TBD

## Requirements

[+] Expand table

Requirement	Value
Header	wcsplugin.h

## Feedback

Was this page helpful?

 Yes

 No

Provide product feedback  | Get help at Microsoft Q&A

# PrimaryJabColors structure (wcsplugin.h)

Article 02/22/2024

This structure contains eight primary colors in Jab coordinates.

## Syntax

C++

```
typedef struct _PrimaryJabColors {
    JabColorF red;
    JabColorF yellow;
    JabColorF green;
    JabColorF cyan;
    JabColorF blue;
    JabColorF magenta;
    JabColorF black;
    JabColorF white;
} PrimaryJabColors;
```

## Members

red

Red primary.

yellow

Yellow primary.

green

Green primary.

cyan

Cyan primary.

blue

Blue primary.

magenta

Magenta primary.

black

Black primary.

white

White primary.

## Requirements

[ ] Expand table

Requirement	Value
Minimum supported client	Windows Vista [desktop apps only]
Minimum supported server	Windows Server 2008 [desktop apps only]
Header	wcsplugin.h

## See also

- [Basic color management concepts](#)
- [Structures](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# PrimaryXYZColors structure (wcsplugin.h)

Article 02/22/2024

This structure contains eight primary colors in XYZ coordinates.

## Syntax

C++

```
typedef struct _PrimaryXYZColors {
    XYZColorF red;
    XYZColorF yellow;
    XYZColorF green;
    XYZColorF cyan;
    XYZColorF blue;
    XYZColorF magenta;
    XYZColorF black;
    XYZColorF white;
} PrimaryXYZColors;
```

## Members

red

Red primary.

yellow

Yellow primary.

green

Green primary.

cyan

Cyan primary.

blue

Blue primary.

magenta

Magenta primary.

black

Black primary.

white

White primary.

## Requirements

[ ] Expand table

Requirement	Value
Minimum supported client	Windows Vista [desktop apps only]
Minimum supported server	Windows Server 2008 [desktop apps only]
Header	wcsplugin.h

## See also

- [Basic color management concepts](#)
- [Structures](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# XYZColorF structure (wcsplugin.h)

Article 02/22/2024

TBD

## Syntax

C++

```
typedef struct _XYZColorF {
    FLOAT X;
    FLOAT Y;
    FLOAT Z;
} XYZColorF;
```

## Members

X

TBD

Y

TBD

Z

TBD

## Requirements

[\[+\] Expand table](#)

Requirement	Value
Header	wcsplugin.h

## Feedback

Was this page helpful?

 Yes

 No

Provide product feedback  | Get help at Microsoft Q&A

# wingdi.h header

Article01/24/2023

This header is used by multiple technologies. For more information, see:

- [Data Exchange](#)
- [Direct3D 9 Graphics](#)
- [DirectShow](#)
- [Display Devices Reference](#)
- [Internationalization for Windows Applications](#)
- [OpenGL](#)
- [Tablet PC](#)
- [Windows Color System](#)
- [Windows GDI](#)

wingdi.h contains the following programming interfaces:

## Functions

 [Expand table](#)

<a href="#">AbortDoc</a>
The AbortDoc function stops the current print job and erases everything drawn since the last call to the StartDoc function.
<a href="#">AbortPath</a>
The AbortPath function closes and discards any paths in the specified device context.
<a href="#">AddFontMemResourceEx</a>
The AddFontMemResourceEx function adds the font resource from a memory image to the system.
<a href="#">AddFontResourceA</a>
The AddFontResource function adds the font resource from the specified file to the system font table. The font can subsequently be used for text output by any application. (ANSI)
<a href="#">AddFontResourceExA</a>
The AddFontResourceEx function adds the font resource from the specified file to the system.

Fonts added with the AddFontResourceEx function can be marked as private and not enumerable. (ANSI)

#### [AddFontResourceExW](#)

The AddFontResourceEx function adds the font resource from the specified file to the system. Fonts added with the AddFontResourceEx function can be marked as private and not enumerable. (Unicode)

#### [AddFontResourceW](#)

The AddFontResource function adds the font resource from the specified file to the system font table. The font can subsequently be used for text output by any application. (Unicode)

#### [AlphaBlend](#)

The AlphaBlend function displays bitmaps that have transparent or semitransparent pixels.

#### [AngleArc](#)

The AngleArc function draws a line segment and an arc.

#### [AnimatePalette](#)

The AnimatePalette function replaces entries in the specified logical palette.

#### [Arc](#)

The Arc function draws an elliptical arc.

#### [ArcTo](#)

The ArcTo function draws an elliptical arc.

#### [BeginPath](#)

The BeginPath function opens a path bracket in the specified device context.

#### [BitBlt](#)

The BitBlt function performs a bit-block transfer of the color data corresponding to a rectangle of pixels from the specified source device context into a destination device context.

#### [CancelDC](#)

The CancelDC function cancels any pending operation on the specified device context (DC).

#### [CheckColorsInGamut](#)

The CheckColorsInGamut function determines whether a specified set of RGB triples lies in the

output gamut of a specified device. The RGB triples are interpreted in the input logical color space.

#### [ChoosePixelFormat](#)

The ChoosePixelFormat function attempts to match an appropriate pixel format supported by a device context to a given pixel format specification.

#### [Chord](#)

The Chord function draws a chord (a region bounded by the intersection of an ellipse and a line segment, called a secant). The chord is outlined by using the current pen and filled by using the current brush.

#### [CloseEnhMetaFile](#)

The CloseEnhMetaFile function closes an enhanced-metafile device context and returns a handle that identifies an enhanced-format metafile.

#### [CloseFigure](#)

The CloseFigure function closes an open figure in a path.

#### [CloseMetaFile](#)

The CloseMetaFile function closes a metafile device context and returns a handle that identifies a Windows-format metafile.

#### [CMYK](#)

The CMYK macro creates a CMYK color value by combining the specified cyan, magenta, yellow, and black values.

#### [ColorCorrectPalette](#)

The ColorCorrectPalette function corrects the entries of a palette using the WCS 1.0 parameters in the specified device context.

#### [ColorMatchToTarget](#)

The ColorMatchToTarget function enables you to preview colors as they would appear on the target device.

#### [CombineRgn](#)

The CombineRgn function combines two regions and stores the result in a third region. The two regions are combined according to the specified mode.

#### [CombineTransform](#)

The CombineTransform function concatenates two world-space to page-space transformations.

#### [CopyEnhMetaFileA](#)

The CopyEnhMetaFile function copies the contents of an enhanced-format metafile to a specified file. (ANSI)

#### [CopyEnhMetaFileW](#)

The CopyEnhMetaFile function copies the contents of an enhanced-format metafile to a specified file. (Unicode)

#### [CopyMetaFileA](#)

The CopyMetaFile function copies the content of a Windows-format metafile to the specified file. (ANSI)

#### [CopyMetaFileW](#)

The CopyMetaFile function copies the content of a Windows-format metafile to the specified file. (Unicode)

#### [CreateBitmap](#)

The CreateBitmap function creates a bitmap with the specified width, height, and color format (color planes and bits-per-pixel).

#### [CreateBitmapIndirect](#)

The CreateBitmapIndirect function creates a bitmap with the specified width, height, and color format (color planes and bits-per-pixel).

#### [CreateBrushIndirect](#)

The CreateBrushIndirect function creates a logical brush that has the specified style, color, and pattern.

#### [CreateColorSpaceA](#)

The CreateColorSpace function creates a logical color space. (ANSI)

#### [CreateColorSpaceW](#)

The CreateColorSpace function creates a logical color space. (Unicode)

#### [CreateCompatibleBitmap](#)

The CreateCompatibleBitmap function creates a bitmap compatible with the device that is associated with the specified device context.

## [CreateCompatibleDC](#)

The CreateCompatibleDC function creates a memory device context (DC) compatible with the specified device.

## [CreateDCA](#)

The CreateDC function creates a device context (DC) for a device using the specified name. (ANSI)

## [CreateDCW](#)

The CreateDC function creates a device context (DC) for a device using the specified name. (Unicode)

## [CreateDIBitmap](#)

The CreateDIBitmap function creates a compatible bitmap (DDB) from a DIB and, optionally, sets the bitmap bits.

## [CreateDIBPatternBrush](#)

The CreateDIBPatternBrush function creates a logical brush that has the pattern specified by the specified device-independent bitmap (DIB).

## [CreateDIBPatternBrushPt](#)

The CreateDIBPatternBrushPt function creates a logical brush that has the pattern specified by the device-independent bitmap (DIB).

## [CreateDIBSection](#)

The CreateDIBSection function creates a DIB that applications can write to directly.

## [CreateDiscardableBitmap](#)

The CreateDiscardableBitmap function creates a discardable bitmap that is compatible with the specified device.

## [CreateEllipticRgn](#)

The CreateEllipticRgn function creates an elliptical region.

## [CreateEllipticRgnIndirect](#)

The CreateEllipticRgnIndirect function creates an elliptical region.

## [CreateEnhMetaFileA](#)

The CreateEnhMetaFile function creates a device context for an enhanced-format metafile. This device context can be used to store a device-independent picture. (ANSI)

#### [CreateEnhMetaFileW](#)

The CreateEnhMetaFile function creates a device context for an enhanced-format metafile. This device context can be used to store a device-independent picture. (Unicode)

#### [CreateFontA](#)

The CreateFont function creates a logical font with the specified characteristics. The logical font can subsequently be selected as the font for any device. (ANSI)

#### [CreateFontIndirectA](#)

The CreateFontIndirect function creates a logical font that has the specified characteristics. The font can subsequently be selected as the current font for any device context. (ANSI)

#### [CreateFontIndirectExA](#)

The CreateFontIndirectEx function specifies a logical font that has the characteristics in the specified structure. The font can subsequently be selected as the current font for any device context. (ANSI)

#### [CreateFontIndirectExW](#)

The CreateFontIndirectEx function specifies a logical font that has the characteristics in the specified structure. The font can subsequently be selected as the current font for any device context. (Unicode)

#### [CreateFontIndirectW](#)

The CreateFontIndirect function creates a logical font that has the specified characteristics. The font can subsequently be selected as the current font for any device context. (Unicode)

#### [CreateFontW](#)

The CreateFont function creates a logical font with the specified characteristics. The logical font can subsequently be selected as the font for any device. (Unicode)

#### [CreateHalftonePalette](#)

The CreateHalftonePalette function creates a halftone palette for the specified device context (DC).

#### [CreateHatchBrush](#)

The CreateHatchBrush function creates a logical brush that has the specified hatch pattern and color.

## [CreateIC](#)

The CreateIC function creates an information context for the specified device. (ANSI)

## [CreateICW](#)

The CreateIC function creates an information context for the specified device. (Unicode)

## [CreateMetaFileA](#)

The CreateMetaFile function creates a device context for a Windows-format metafile. (ANSI)

## [CreateMetaFileW](#)

The CreateMetaFile function creates a device context for a Windows-format metafile. (Unicode)

## [CreatePalette](#)

The CreatePalette function creates a logical palette.

## [CreatePatternBrush](#)

The CreatePatternBrush function creates a logical brush with the specified bitmap pattern. The bitmap can be a DIB section bitmap, which is created by the CreateDIBSection function, or it can be a device-dependent bitmap.

## [CreatePen](#)

The CreatePen function creates a logical pen that has the specified style, width, and color. The pen can subsequently be selected into a device context and used to draw lines and curves.

## [CreatePenIndirect](#)

The CreatePenIndirect function creates a logical cosmetic pen that has the style, width, and color specified in a structure.

## [CreatePolygonRgn](#)

The CreatePolygonRgn function creates a polygonal region.

## [CreatePolyPolygonRgn](#)

The CreatePolyPolygonRgn function creates a region consisting of a series of polygons. The polygons can overlap.

## [CreateRectRgn](#)

The CreateRectRgn function creates a rectangular region.

## [CreateRectRgnIndirect](#)

The CreateRectRgnIndirect function creates a rectangular region.

## [CreateRoundRectRgn](#)

The CreateRoundRectRgn function creates a rectangular region with rounded corners.

## [CreateScalableFontResourceA](#)

The CreateScalableFontResource function creates a font resource file for a scalable font. (ANSI)

## [CreateScalableFontResourceW](#)

The CreateScalableFontResource function creates a font resource file for a scalable font. (Unicode)

## [CreateSolidBrush](#)

The CreateSolidBrush function creates a logical brush that has the specified solid color.

## [DeleteColorSpace](#)

The DeleteColorSpace function removes and destroys a specified color space.

## [DeleteDC](#)

The DeleteDC function deletes the specified device context (DC).

## [DeleteEnhMetaFile](#)

The DeleteEnhMetaFile function deletes an enhanced-format metafile or an enhanced-format metafile handle.

## [DeleteMetaFile](#)

The DeleteMetaFile function deletes a Windows-format metafile or Windows-format metafile handle.

## [DeleteObject](#)

The DeleteObject function deletes a logical pen, brush, font, bitmap, region, or palette, freeing all system resources associated with the object. After the object is deleted, the specified handle is no longer valid.

## [DescribePixelFormat](#)

The DescribePixelFormat function obtains information about the pixel format identified by iPixelFormat of the device associated with hdc. The function sets the members of the PIXELFORMATDESCRIPTOR structure pointed to by ppfd with that pixel format data.

## [DeviceCapabilitiesA](#)

The DeviceCapabilities function retrieves the capabilities of a printer driver. (ANSI)

## [DeviceCapabilitiesW](#)

The DeviceCapabilities function retrieves the capabilities of a printer driver. (Unicode)

## [DPtoLP](#)

The DPtoLP function converts device coordinates into logical coordinates. The conversion depends on the mapping mode of the device context, the settings of the origins and extents for the window and viewport, and the world transformation.

## [DrawEscape](#)

The DrawEscape function provides drawing capabilities of the specified video display that are not directly available through the graphics device interface (GDI).

## [Ellipse](#)

The Ellipse function draws an ellipse. The center of the ellipse is the center of the specified bounding rectangle. The ellipse is outlined by using the current pen and is filled by using the current brush.

## [EndDoc](#)

The EndDoc function ends a print job.

## [EndPage](#)

The EndPage function notifies the device that the application has finished writing to a page. This function is typically used to direct the device driver to advance to a new page.

## [EndPath](#)

The EndPath function closes a path bracket and selects the path defined by the bracket into the specified device context.

## [EnumEnhMetaFile](#)

The EnumEnhMetaFile function enumerates the records within an enhanced-format metafile by retrieving each record and passing it to the specified callback function.

## [EnumFontFamiliesA](#)

The EnumFontFamilies function enumerates the fonts in a specified font family that are available on a specified device. (ANSI)

## [EnumFontFamiliesExA](#)

The `EnumFontFamiliesEx` function enumerates all uniquely-named fonts in the system that match the font characteristics specified by the `LOGFONT` structure. `EnumFontFamiliesEx` enumerates fonts based on typeface name, character set, or both. (ANSI)

## [EnumFontFamiliesExW](#)

The `EnumFontFamiliesEx` function enumerates all uniquely-named fonts in the system that match the font characteristics specified by the `LOGFONT` structure. `EnumFontFamiliesEx` enumerates fonts based on typeface name, character set, or both. (Unicode)

## [EnumFontFamiliesW](#)

The `EnumFontFamilies` function enumerates the fonts in a specified font family that are available on a specified device. (Unicode)

## [EnumFontsA](#)

The `EnumFonts` function enumerates the fonts available on a specified device. (ANSI)

## [EnumFontsW](#)

The `EnumFonts` function enumerates the fonts available on a specified device. (Unicode)

## [EnumICMProfilesA](#)

The `EnumICMProfiles` function enumerates the different output color profiles that the system supports for a given device context. (ANSI)

## [EnumICMProfilesW](#)

The `EnumICMProfiles` function enumerates the different output color profiles that the system supports for a given device context. (Unicode)

## [EnumMetaFile](#)

The `EnumMetaFile` function enumerates the records within a Windows-format metafile by retrieving each record and passing it to the specified callback function.

## [EnumObjects](#)

The `EnumObjects` function enumerates the pens or brushes available for the specified device context (DC).

## [EqualRgn](#)

The `EqualRgn` function checks the two specified regions to determine whether they are identical. The function considers two regions identical if they are equal in size and shape.

## [Escape](#)

Enables an application to access the system-defined device capabilities that are not available through GDI.

## [ExcludeClipRect](#)

The ExcludeClipRect function creates a new clipping region that consists of the existing clipping region minus the specified rectangle.

## [ExtCreatePen](#)

The ExtCreatePen function creates a logical cosmetic or geometric pen that has the specified style, width, and brush attributes.

## [ExtCreateRegion](#)

The ExtCreateRegion function creates a region from the specified region and transformation data.

## [ExtEscape](#)

The ExtEscape function enables an application to access device capabilities that are not available through GDI.

## [ExtFloodFill](#)

The ExtFloodFill function fills an area of the display surface with the current brush.

## [ExtSelectClipRgn](#)

The ExtSelectClipRgn function combines the specified region with the current clipping region using the specified mode.

## [ExtTextOutA](#)

The ExtTextOut function draws text using the currently selected font, background color, and text color. You can optionally provide dimensions to be used for clipping, opaquing, or both. (ANSI)

## [ExtTextOutW](#)

The ExtTextOut function draws text using the currently selected font, background color, and text color. You can optionally provide dimensions to be used for clipping, opaquing, or both. (Unicode)

## [FillPath](#)

The FillPath function closes any open figures in the current path and fills the path's interior by using the current brush and polygon-filling mode.

## [FillRgn](#)

The FillRgn function fills a region by using the specified brush.

## [FlattenPath](#)

The FlattenPath function transforms any curves in the path that is selected into the current device context (DC), turning each curve into a sequence of lines.

## [FloodFill](#)

The FloodFill function fills an area of the display surface with the current brush. The area is assumed to be bounded as specified by the color parameter.

## [FrameRgn](#)

The FrameRgn function draws a border around the specified region by using the specified brush.

## [GdiAlphaBlend](#)

The GdiAlphaBlend function displays bitmaps that have transparent or semitransparent pixels.

## [GdiComment](#)

The GdiComment function copies a comment from a buffer into a specified enhanced-format metafile.

## [GdiFlush](#)

The GdiFlush function flushes the calling thread's current batch.

## [GdiGetBatchLimit](#)

The GdiGetBatchLimit function returns the maximum number of function calls that can be accumulated in the calling thread's current batch. The system flushes the current batch whenever this limit is exceeded.

## [GdiGradientFill](#)

The GdiGradientFill function fills rectangle and triangle structures.

## [GdiSetBatchLimit](#)

The GdiSetBatchLimit function sets the maximum number of function calls that can be accumulated in the calling thread's current batch. The system flushes the current batch whenever this limit is exceeded.

## [GdiTransparentBlt](#)

The GdiTransparentBlt function performs a bit-block transfer of the color data corresponding to a rectangle of pixels from the specified source device context into a destination device context.

#### [GetArcDirection](#)

The GetArcDirection function retrieves the current arc direction for the specified device context. Arc and rectangle functions use the arc direction.

#### [GetAspectRatioFilterEx](#)

The GetAspectRatioFilterEx function retrieves the setting for the current aspect-ratio filter.

#### [GetBitmapBits](#)

The GetBitmapBits function copies the bitmap bits of a specified device-dependent bitmap into a buffer.

#### [GetBitmapDimensionEx](#)

The GetBitmapDimensionEx function retrieves the dimensions of a compatible bitmap. The retrieved dimensions must have been set by the SetBitmapDimensionEx function.

#### [GetBkColor](#)

The GetBkColor function returns the current background color for the specified device context.

#### [GetBkMode](#)

The GetBkMode function returns the current background mix mode for a specified device context. The background mix mode of a device context affects text, hatched brushes, and pen styles that are not solid lines.

#### [GetBoundsRect](#)

The GetBoundsRect function obtains the current accumulated bounding rectangle for a specified device context.

#### [GetBrushOrgEx](#)

The GetBrushOrgEx function retrieves the current brush origin for the specified device context. This function replaces the GetBrushOrg function.

#### [GetBValue](#)

The GetBValue macro retrieves an intensity value for the blue component of a red, green, blue (RGB) value.

#### [GetCharABCWidthsA](#)

The GetCharABCWidths function retrieves the widths, in logical units, of consecutive characters in

a specified range from the current TrueType font. This function succeeds only with TrueType fonts. (ANSI)

#### [GetCharABCWidthsFloatA](#)

The GetCharABCWidthsFloat function retrieves the widths, in logical units, of consecutive characters in a specified range from the current font. (ANSI)

#### [GetCharABCWidthsFloatW](#)

The GetCharABCWidthsFloat function retrieves the widths, in logical units, of consecutive characters in a specified range from the current font. (Unicode)

#### [GetCharABCWidthsI](#)

The GetCharABCWidthsI function retrieves the widths, in logical units, of consecutive glyph indices in a specified range from the current TrueType font. This function succeeds only with TrueType fonts.

#### [GetCharABCWidthsW](#)

The GetCharABCWidths function retrieves the widths, in logical units, of consecutive characters in a specified range from the current TrueType font. This function succeeds only with TrueType fonts. (Unicode)

#### [GetCharacterPlacementA](#)

The GetCharacterPlacement function retrieves information about a character string, such as character widths, caret positioning, ordering within the string, and glyph rendering. (ANSI)

#### [GetCharacterPlacementW](#)

The GetCharacterPlacement function retrieves information about a character string, such as character widths, caret positioning, ordering within the string, and glyph rendering. (Unicode)

#### [GetCharWidth32A](#)

The GetCharWidth32 function retrieves the widths, in logical coordinates, of consecutive characters in a specified range from the current font. (ANSI)

#### [GetCharWidth32W](#)

The GetCharWidth32 function retrieves the widths, in logical coordinates, of consecutive characters in a specified range from the current font. (Unicode)

#### [GetCharWidthA](#)

The GetCharWidth function retrieves the widths, in logical coordinates, of consecutive characters in a specified range from the current font. (ANSI)

## [GetCharWidthFloatA](#)

The GetCharWidthFloat function retrieves the fractional widths of consecutive characters in a specified range from the current font. (ANSI)

## [GetCharWidthFloatW](#)

The GetCharWidthFloat function retrieves the fractional widths of consecutive characters in a specified range from the current font. (Unicode)

## [GetCharWidthI](#)

The GetCharWidthI function retrieves the widths, in logical coordinates, of consecutive glyph indices in a specified range from the current font.

## [GetCharWidthW](#)

The GetCharWidth function retrieves the widths, in logical coordinates, of consecutive characters in a specified range from the current font. (Unicode)

## [GetClipBox](#)

The GetClipBox function retrieves the dimensions of the tightest bounding rectangle that can be drawn around the current visible area on the device.

## [GetClipRgn](#)

The GetClipRgn function retrieves a handle identifying the current application-defined clipping region for the specified device context.

## [GetColorAdjustment](#)

The GetColorAdjustment function retrieves the color adjustment values for the specified device context (DC).

## [GetColorSpace](#)

The GetColorSpace function retrieves the handle to the input color space from a specified device context.

## [GetCurrentObject](#)

The GetCurrentObject function retrieves a handle to an object of the specified type that has been selected into the specified device context (DC).

## [GetCurrentPositionEx](#)

The GetCurrentPositionEx function retrieves the current position in logical coordinates.

## [GetCValue](#)

The GetCValue macro retrieves the cyan color value from a CMYK color value.

## [GetDCBrushColor](#)

The GetDCBrushColor function retrieves the current brush color for the specified device context (DC).

## [GetDCOrgEx](#)

The GetDCOrgEx function retrieves the final translation origin for a specified device context (DC).

## [GetDCPenColor](#)

The GetDCPenColor function retrieves the current pen color for the specified device context (DC).

## [GetDeviceCaps](#)

The GetDeviceCaps function retrieves device-specific information for the specified device.

## [GetDeviceGammaRamp](#)

The GetDeviceGammaRamp function gets the gamma ramp on direct color display boards having drivers that support downloadable gamma ramps in hardware.

## [GetDIBColorTable](#)

The GetDIBColorTable function retrieves RGB (red, green, blue) color values from a range of entries in the color table of the DIB section bitmap that is currently selected into a specified device context.

## [GetDIBits](#)

The GetDIBits function retrieves the bits of the specified compatible bitmap and copies them into a buffer as a DIB using the specified format.

## [GetEnhMetaFileA](#)

The GetEnhMetaFile function creates a handle that identifies the enhanced-format metafile stored in the specified file. (ANSI)

## [GetEnhMetaFileBits](#)

The GetEnhMetaFileBits function retrieves the contents of the specified enhanced-format metafile and copies them into a buffer.

## [GetEnhMetaFileDescriptionA](#)

The GetEnhMetaFileDescription function retrieves an optional text description from an enhanced-format metafile and copies the string to the specified buffer. (ANSI)

#### [GetEnhMetaFileDescriptionW](#)

The GetEnhMetaFileDescription function retrieves an optional text description from an enhanced-format metafile and copies the string to the specified buffer. (Unicode)

#### [GetEnhMetaFileHeader](#)

The GetEnhMetaFileHeader function retrieves the record containing the header for the specified enhanced-format metafile.

#### [GetEnhMetaFilePaletteEntries](#)

The GetEnhMetaFilePaletteEntries function retrieves optional palette entries from the specified enhanced metafile.

#### [GetEnhMetaFilePixelFormat](#)

The GetEnhMetaFilePixelFormat function retrieves pixel format information for an enhanced metafile.

#### [GetEnhMetaFileW](#)

The GetEnhMetaFile function creates a handle that identifies the enhanced-format metafile stored in the specified file. (Unicode)

#### [GetFontData](#)

The GetFontData function retrieves font metric data for a TrueType font.

#### [GetFontLanguageInfo](#)

The GetFontLanguageInfo function returns information about the currently selected font for the specified display context. Applications typically use this information and the GetCharacterPlacement function to prepare a character string for display.

#### [GetFontUnicodeRanges](#)

The GetFontUnicodeRanges function returns information about which Unicode characters are supported by a font. The information is returned as a GLYPHSET structure.

#### [GetGlyphIndicesA](#)

The GetGlyphIndices function translates a string into an array of glyph indices. The function can be used to determine whether a glyph exists in a font. (ANSI)

#### [GetGlyphIndicesW](#)

The GetGlyphIndices function translates a string into an array of glyph indices. The function can be used to determine whether a glyph exists in a font. (Unicode)

#### [GetGlyphOutlineA](#)

The GetGlyphOutline function retrieves the outline or bitmap for a character in the TrueType font that is selected into the specified device context. (ANSI)

#### [GetGlyphOutlineW](#)

The GetGlyphOutline function retrieves the outline or bitmap for a character in the TrueType font that is selected into the specified device context. (Unicode)

#### [GetGraphicsMode](#)

The GetGraphicsMode function retrieves the current graphics mode for the specified device context.

#### [GetGValue](#)

The GetGValue macro retrieves an intensity value for the green component of a red, green, blue (RGB) value.

#### [GetICMProfileA](#)

The GetICMProfile function retrieves the file name of the current output color profile for a specified device context. (ANSI)

#### [GetICMProfileW](#)

The GetICMProfile function retrieves the file name of the current output color profile for a specified device context. (Unicode)

#### [GetKerningPairsA](#)

The GetKerningPairs function retrieves the character-kerning pairs for the currently selected font for the specified device context. (ANSI)

#### [GetKerningPairsW](#)

The GetKerningPairs function retrieves the character-kerning pairs for the currently selected font for the specified device context. (Unicode)

#### [GetKValue](#)

The GetKValue macro retrieves the black color value from a CMYK color value.

#### [GetLayout](#)

The GetLayout function returns the layout of a device context (DC).

#### [GetLogColorSpaceA](#)

The GetLogColorSpace function retrieves the color space definition identified by a specified handle. (ANSI)

#### [GetLogColorSpaceW](#)

The GetLogColorSpace function retrieves the color space definition identified by a specified handle. (Unicode)

#### [GetMapMode](#)

The GetMapMode function retrieves the current mapping mode.

#### [GetMetaFileA](#)

The GetMetaFile function creates a handle that identifies the metafile stored in the specified file. (ANSI)

#### [GetMetaFileBitsEx](#)

The GetMetaFileBitsEx function retrieves the contents of a Windows-format metafile and copies them into the specified buffer.

#### [GetMetaFileW](#)

The GetMetaFile function creates a handle that identifies the metafile stored in the specified file. (Unicode)

#### [GetMetaRgn](#)

The GetMetaRgn function retrieves the current metaregion for the specified device context.

#### [GetMiterLimit](#)

The GetMiterLimit function retrieves the miter limit for the specified device context.

#### [GetMValue](#)

The GetMValue macro retrieves the magenta color value from a CMYK color value.

#### [GetNearestColor](#)

The GetNearestColor function retrieves a color value identifying a color from the system palette that will be displayed when the specified color value is used.

#### [GetNearestPaletteIndex](#)

The GetNearestPaletteIndex function retrieves the index for the entry in the specified logical palette most closely matching a specified color value.

#### [GetObject](#)

The GetObject function (wingdi.h) retrieves information for the specified graphics object.

#### [GetObjectA](#)

The GetObject function retrieves information for the specified graphics object. (GetObjectA)

#### [GetObjectType](#)

The GetObjectType retrieves the type of the specified object.

#### [GetObjectW](#)

The GetObjectW (Unicode) function (wingdi.h) retrieves information for the specified graphics object.

#### [GetOutlineTextMetricsA](#)

The GetOutlineTextMetrics function retrieves text metrics for TrueType fonts. (ANSI)

#### [GetOutlineTextMetricsW](#)

The GetOutlineTextMetrics function retrieves text metrics for TrueType fonts. (Unicode)

#### [GetPaletteEntries](#)

The GetPaletteEntries function retrieves a specified range of palette entries from the given logical palette.

#### [GetPath](#)

The GetPath function retrieves the coordinates defining the endpoints of lines and the control points of curves found in the path that is selected into the specified device context.

#### [GetPixel](#)

The GetPixel function retrieves the red, green, blue (RGB) color value of the pixel at the specified coordinates.

#### [GetPixelFormat](#)

The GetPixelFormat function obtains the index of the currently selected pixel format of the specified device context.

#### [GetPolyFillMode](#)

The GetPolyFillMode function retrieves the current polygon fill mode.

#### [GetRandomRgn](#)

The GetRandomRgn function copies the system clipping region of a specified device context to a specific region.

#### [GetRasterizerCaps](#)

The GetRasterizerCaps function returns flags indicating whether TrueType fonts are installed in the system.

#### [GetRegionData](#)

The GetRegionData function fills the specified buffer with data describing a region. This data includes the dimensions of the rectangles that make up the region.

#### [GetRgnBox](#)

The GetRgnBox function retrieves the bounding rectangle of the specified region.

#### [GetROP2](#)

The GetROP2 function retrieves the foreground mix mode of the specified device context. The mix mode specifies how the pen or interior color and the color already on the screen are combined to yield a new color.

#### [GetRValue](#)

The GetRValue macro retrieves an intensity value for the red component of a red, green, blue (RGB) value.

#### [GetStockObject](#)

The GetStockObject function retrieves a handle to one of the stock pens, brushes, fonts, or palettes.

#### [GetStretchBltMode](#)

The GetStretchBltMode function retrieves the current stretching mode. The stretching mode defines how color data is added to or removed from bitmaps that are stretched or compressed when the StretchBlt function is called.

#### [GetSystemPaletteEntries](#)

The GetSystemPaletteEntries function retrieves a range of palette entries from the system palette that is associated with the specified device context (DC).

#### [GetSystemPaletteUse](#)

The `GetSystemPaletteUse` function retrieves the current state of the system (physical) palette for the specified device context (DC).

#### [GetTextAlign](#)

The `GetTextAlign` function retrieves the text-alignment setting for the specified device context.

#### [GetTextCharacterExtra](#)

The `GetTextCharacterExtra` function retrieves the current intercharacter spacing for the specified device context.

#### [GetTextCharset](#)

Retrieves a character set identifier for the font that is currently selected into a specified device context.

#### [GetTextCharsetInfo](#)

Retrieves information about the character set of the font that is currently selected into a specified device context.

#### [GetTextColor](#)

The `GetTextColor` function retrieves the current text color for the specified device context.

#### [GetTextExtentExPointA](#)

The `GetTextExtentExPoint` function retrieves the number of characters in a specified string that will fit within a specified space and fills an array with the text extent for each of those characters. (ANSI)

#### [GetTextExtentExPointI](#)

The `GetTextExtentExPointI` function retrieves the number of characters in a specified string that will fit within a specified space and fills an array with the text extent for each of those characters.

#### [GetTextExtentExPointW](#)

The `GetTextExtentExPoint` function retrieves the number of characters in a specified string that will fit within a specified space and fills an array with the text extent for each of those characters. (Unicode)

#### [GetTextExtentPoint32A](#)

The `GetTextExtentPoint32` function computes the width and height of the specified string of text. (ANSI)

#### [GetTextExtentPoint32W](#)

The `GetTextExtentPoint32` function computes the width and height of the specified string of text. (Unicode)

#### [GetTextExtentPointA](#)

The `GetTextExtentPoint` function computes the width and height of the specified string of text. (ANSI)

#### [GetTextExtentPointI](#)

The `GetTextExtentPointI` function computes the width and height of the specified array of glyph indices.

#### [GetTextExtentPointW](#)

The `GetTextExtentPoint` function computes the width and height of the specified string of text. (Unicode)

#### [GetTextFaceA](#)

The `GetTextFace` function retrieves the typeface name of the font that is selected into the specified device context. (ANSI)

#### [GetTextFaceW](#)

The `GetTextFace` function retrieves the typeface name of the font that is selected into the specified device context. (Unicode)

#### [GetTextMetrics](#)

The `GetTextMetrics` function (`wingdi.h`) fills the specified buffer with the metrics for the currently selected font.

#### [GetTextMetricsA](#)

The `GetTextMetrics` function fills the specified buffer with the metrics for the currently selected font. (`GetTextMetricsA`)

#### [GetTextMetricsW](#)

The `GetTextMetricsW` (Unicode) function (`wingdi.h`) fills the specified buffer with the metrics for the currently selected font.

#### [GetViewportExtEx](#)

The `GetViewportExtEx` function retrieves the x-extent and y-extent of the current viewport for the specified device context.

#### [GetViewportOrgEx](#)

The GetViewportOrgEx function retrieves the x-coordinates and y-coordinates of the viewport origin for the specified device context.

#### [GetWindowExtEx](#)

This function retrieves the x-extent and y-extent of the window for the specified device context.

#### [GetWindowOrgEx](#)

The GetWindowOrgEx function retrieves the x-coordinates and y-coordinates of the window origin for the specified device context.

#### [GetWinMetaFileBits](#)

The GetWinMetaFileBits function converts the enhanced-format records from a metafile into Windows-format records and stores the converted records in the specified buffer.

#### [GetWorldTransform](#)

The GetWorldTransform function retrieves the current world-space to page-space transformation.

#### [GetYValue](#)

The GetYValue macro retrieves the yellow color value from a CMYK color value.

#### [GradientFill](#)

The GradientFill function fills rectangle and triangle structures.

#### [IntersectClipRect](#)

The IntersectClipRect function creates a new clipping region from the intersection of the current clipping region and the specified rectangle.

#### [InvertRgn](#)

The InvertRgn function inverts the colors in the specified region.

#### [LineDDA](#)

The LineDDA function determines which pixels should be highlighted for a line defined by the specified starting and ending points.

#### [LineTo](#)

The LineTo function draws a line from the current position up to, but not including, the specified point.

#### [LPtoDP](#)

The LPtoDP function converts logical coordinates into device coordinates. The conversion depends on the mapping mode of the device context, the settings of the origins and extents for the window and viewport, and the world transformation.

#### [MAKEPOINTS](#)

The MAKEPOINTS macro converts a value that contains the x- and y-coordinates of a point into a POINTS structure.

#### [MAKEROP4](#)

The MAKEROP4 macro creates a quaternary raster operation code for use with the MaskBlt function.

#### [MaskBlt](#)

The MaskBlt function combines the color data for the source and destination bitmaps using the specified mask and raster operation.

#### [ModifyWorldTransform](#)

The ModifyWorldTransform function changes the world transformation for a device context using the specified mode.

#### [MoveToEx](#)

The MoveToEx function updates the current position to the specified point and optionally returns the previous position.

#### [OffsetClipRgn](#)

The OffsetClipRgn function moves the clipping region of a device context by the specified offsets.

#### [OffsetRgn](#)

The OffsetRgn function moves a region by the specified offsets.

#### [OffsetViewportOrgEx](#)

The OffsetViewportOrgEx function modifies the viewport origin for a device context using the specified horizontal and vertical offsets.

#### [OffsetWindowOrgEx](#)

The OffsetWindowOrgEx function modifies the window origin for a device context using the specified horizontal and vertical offsets.

#### [PaintRgn](#)

The PaintRgn function paints the specified region by using the brush currently selected into the device context.

#### [PALETTEINDEX](#)

The PALETTEINDEX macro accepts an index to a logical-color palette entry and returns a palette-entry specifier consisting of a COLORREF value that specifies the color associated with the given index.

#### [PALETERGB](#)

The PALETERGB macro accepts three values that represent the relative intensities of red, green, and blue and returns a palette-relative red, green, blue (RGB) specifier consisting of 2 in the high-order byte and an RGB value in the three low-order bytes. An application using a color palette can pass this specifier, instead of an explicit RGB value, to functions that expect a color.

#### [PatBlt](#)

The PatBlt function paints the specified rectangle using the brush that is currently selected into the specified device context. The brush color and the surface color or colors are combined by using the specified raster operation.

#### [PathToRegion](#)

The PathToRegion function creates a region from the path that is selected into the specified device context. The resulting region uses device coordinates.

#### [Pie](#)

The Pie function draws a pie-shaped wedge bounded by the intersection of an ellipse and two radials. The pie is outlined by using the current pen and filled by using the current brush.

#### [PlayEnhMetaFile](#)

The PlayEnhMetaFile function displays the picture stored in the specified enhanced-format metafile.

#### [PlayEnhMetaFileRecord](#)

The PlayEnhMetaFileRecord function plays an enhanced-metafile record by executing the graphics device interface (GDI) functions identified by the record.

#### [PlayMetaFile](#)

The PlayMetaFile function displays the picture stored in the given Windows-format metafile on the specified device.

#### [PlayMetaFileRecord](#)

The PlayMetaFileRecord function plays a Windows-format metafile record by executing the graphics device interface (GDI) function contained within that record.

### [PlgBlt](#)

The PlgBlt function performs a bit-block transfer of the bits of color data from the specified rectangle in the source device context to the specified parallelogram in the destination device context.

### [PolyBezier](#)

The PolyBezier function draws one or more Bézier curves.

### [PolyBezierTo](#)

The PolyBezierTo function draws one or more Bézier curves.

### [PolyDraw](#)

The PolyDraw function draws a set of line segments and Bézier curves.

### [Polygon](#)

The Polygon function draws a polygon consisting of two or more vertices connected by straight lines. The polygon is outlined by using the current pen and filled by using the current brush and polygon fill mode.

### [Polyline](#)

The Polyline function draws a series of line segments by connecting the points in the specified array.

### [PolylineTo](#)

The PolylineTo function draws one or more straight lines.

### [PolyPolygon](#)

The PolyPolygon function draws a series of closed polygons. Each polygon is outlined by using the current pen and filled by using the current brush and polygon fill mode. The polygons drawn by this function can overlap.

### [PolyPolyline](#)

The PolyPolyline function draws multiple series of connected line segments.

### [PolyTextOutA](#)

The PolyTextOut function draws several strings using the font and text colors currently selected in the specified device context. (ANSI)

#### [PolyTextOutW](#)

The PolyTextOut function draws several strings using the font and text colors currently selected in the specified device context. (Unicode)

#### [PtInRegion](#)

The PtInRegion function determines whether the specified point is inside the specified region.

#### [PtVisible](#)

The PtVisible function determines whether the specified point is within the clipping region of a device context.

#### [RealizePalette](#)

The RealizePalette function maps palette entries from the current logical palette to the system palette.

#### [Rectangle](#)

The Rectangle function draws a rectangle. The rectangle is outlined by using the current pen and filled by using the current brush.

#### [RectInRegion](#)

The RectInRegion function determines whether any part of the specified rectangle is within the boundaries of a region.

#### [RectVisible](#)

The RectVisible function determines whether any part of the specified rectangle lies within the clipping region of a device context.

#### [RemoveFontMemResourceEx](#)

The RemoveFontMemResourceEx function removes the fonts added from a memory image file.

#### [RemoveFontResourceA](#)

The RemoveFontResource function removes the fonts in the specified file from the system font table. (ANSI)

#### [RemoveFontResourceExA](#)

The RemoveFontResourceEx function removes the fonts in the specified file from the system font table. (ANSI)

#### [RemoveFontResourceExW](#)

The RemoveFontResourceEx function removes the fonts in the specified file from the system font table. (Unicode)

#### [RemoveFontResourceW](#)

The RemoveFontResource function removes the fonts in the specified file from the system font table. (Unicode)

#### [ResetDCA](#)

The ResetDC function updates the specified printer or plotter device context (DC) using the specified information. (ANSI)

#### [ResetDCW](#)

The ResetDC function updates the specified printer or plotter device context (DC) using the specified information. (Unicode)

#### [ResizePalette](#)

The ResizePalette function increases or decreases the size of a logical palette based on the specified value.

#### [RestoreDC](#)

The RestoreDC function restores a device context (DC) to the specified state. The DC is restored by popping state information off a stack created by earlier calls to the SaveDC function.

#### [RGB](#)

The RGB macro selects a red, green, blue (RGB) color based on the arguments supplied and the color capabilities of the output device.

#### [RoundRect](#)

The RoundRect function draws a rectangle with rounded corners. The rectangle is outlined by using the current pen and filled by using the current brush.

#### [SaveDC](#)

The SaveDC function saves the current state of the specified device context (DC) by copying data describing selected objects and graphic modes (such as the bitmap, brush, palette, font, pen, region, drawing mode, and mapping mode) to a context stack.

## [ScaleViewportExtEx](#)

The ScaleViewportExtEx function modifies the viewport for a device context using the ratios formed by the specified multiplicands and divisors.

## [ScaleWindowExtEx](#)

The ScaleWindowExtEx function modifies the window for a device context using the ratios formed by the specified multiplicands and divisors.

## [SelectClipPath](#)

The SelectClipPath function selects the current path as a clipping region for a device context, combining the new region with any existing clipping region using the specified mode.

## [SelectClipRgn](#)

The SelectClipRgn function selects a region as the current clipping region for the specified device context.

## [SelectObject](#)

The SelectObject function selects an object into the specified device context (DC). The new object replaces the previous object of the same type.

## [SelectPalette](#)

The SelectPalette function selects the specified logical palette into a device context.

## [SetAbortProc](#)

The SetAbortProc function sets the application-defined abort function that allows a print job to be canceled during spooling.

## [SetArcDirection](#)

The SetArcDirection sets the drawing direction to be used for arc and rectangle functions.

## [SetBitmapBits](#)

The SetBitmapBits function sets the bits of color data for a bitmap to the specified values.

## [SetBitmapDimensionEx](#)

The SetBitmapDimensionEx function assigns preferred dimensions to a bitmap. These dimensions can be used by applications; however, they are not used by the system.

## [SetBkColor](#)

The SetBkColor function sets the current background color to the specified color value, or to the nearest physical color if the device cannot represent the specified color value.

#### [SetBkMode](#)

The SetBkMode function sets the background mix mode of the specified device context. The background mix mode is used with text, hatched brushes, and pen styles that are not solid lines.

#### [SetBoundsRect](#)

The SetBoundsRect function controls the accumulation of bounding rectangle information for the specified device context.

#### [SetBrushOrgEx](#)

The SetBrushOrgEx function sets the brush origin that GDI assigns to the next brush an application selects into the specified device context.

#### [SetColorAdjustment](#)

The SetColorAdjustment function sets the color adjustment values for a device context (DC) using the specified values.

#### [SetColorSpace](#)

The SetColorSpace function defines the input color space for a given device context.

#### [SetDCBrushColor](#)

SetDCBrushColor function sets the current device context (DC) brush color to the specified color value. If the device cannot represent the specified color value, the color is set to the nearest physical color.

#### [SetDCPenColor](#)

SetDCPenColor function sets the current device context (DC) pen color to the specified color value. If the device cannot represent the specified color value, the color is set to the nearest physical color.

#### [SetDeviceGammaRamp](#)

The SetDeviceGammaRamp function sets the gamma ramp on direct color display boards having drivers that support downloadable gamma ramps in hardware.

#### [SetDIBColorTable](#)

The SetDIBColorTable function sets RGB (red, green, blue) color values in a range of entries in the color table of the DIB that is currently selected into a specified device context.

## [SetDIBits](#)

The SetDIBits function sets the pixels in a compatible bitmap (DDB) using the color data found in the specified DIB.

## [SetDIBitsToDevice](#)

The SetDIBitsToDevice function sets the pixels in the specified rectangle on the device that is associated with the destination device context using color data from a DIB, JPEG, or PNG image.

## [SetEnhMetaFileBits](#)

The SetEnhMetaFileBits function creates a memory-based enhanced-format metafile from the specified data.

## [SetGraphicsMode](#)

The SetGraphicsMode function sets the graphics mode for the specified device context.

## [SetICMMode](#)

The SetICMMode function causes Image Color Management to be enabled, disabled, or queried on a given device context (DC).

## [SetICMProfileA](#)

The SetICMProfile function sets a specified color profile as the output profile for a specified device context (DC). (ANSI)

## [SetICMProfileW](#)

The SetICMProfile function sets a specified color profile as the output profile for a specified device context (DC). (Unicode)

## [SetLayout](#)

The SetLayout function changes the layout of a device context (DC).

## [SetMapMode](#)

The SetMapMode function sets the mapping mode of the specified device context. The mapping mode defines the unit of measure used to transform page-space units into device-space units, and also defines the orientation of the device's x and y axes.

## [SetMapperFlags](#)

The SetMapperFlags function alters the algorithm the font mapper uses when it maps logical fonts to physical fonts.

## [SetMetaFileBitsEx](#)

The SetMetaFileBitsEx function creates a memory-based Windows-format metafile from the supplied data.

## [SetMetaRgn](#)

The SetMetaRgn function intersects the current clipping region for the specified device context with the current metaregion and saves the combined region as the new metaregion for the specified device context.

## [SetMiterLimit](#)

The SetMiterLimit function sets the limit for the length of miter joins for the specified device context.

## [SetPaletteEntries](#)

The SetPaletteEntries function sets RGB (red, green, blue) color values and flags in a range of entries in a logical palette.

## [SetPixel](#)

The SetPixel function sets the pixel at the specified coordinates to the specified color.

## [SetPixelFormat](#)

The SetPixelFormat function sets the pixel format of the specified device context to the format specified by the iPixelFormat index.

## [SetPixelV](#)

The SetPixelV function sets the pixel at the specified coordinates to the closest approximation of the specified color. The point must be in the clipping region and the visible part of the device surface.

## [SetPolyFillMode](#)

The SetPolyFillMode function sets the polygon fill mode for functions that fill polygons.

## [SetRectRgn](#)

The SetRectRgn function converts a region into a rectangular region with the specified coordinates.

## [SetROP2](#)

The SetROP2 function sets the current foreground mix mode.

## [SetStretchBltMode](#)

The SetStretchBltMode function sets the bitmap stretching mode in the specified device context.

## [SetSystemPaletteUse](#)

The SetSystemPaletteUse function allows an application to specify whether the system palette contains 2 or 20 static colors.

## [SetTextAlign](#)

The SetTextAlign function sets the text-alignment flags for the specified device context.

## [SetTextCharacterExtra](#)

The SetTextCharacterExtra function sets the intercharacter spacing. Intercharacter spacing is added to each character, including break characters, when the system writes a line of text.

## [SetTextColor](#)

The SetTextColor function sets the text color for the specified device context to the specified color.

## [SetTextJustification](#)

The SetTextJustification function specifies the amount of space the system should add to the break characters in a string of text. The space is added when an application calls the TextOut or ExtTextOut functions.

## [SetViewportExtEx](#)

Sets the horizontal and vertical extents of the viewport for a device context by using the specified values.

## [SetViewportOrgEx](#)

The SetViewportOrgEx function specifies which device point maps to the window origin (0,0).

## [SetWindowExtEx](#)

The SetWindowExtEx function sets the horizontal and vertical extents of the window for a device context by using the specified values.

## [SetWindowOrgEx](#)

The SetWindowOrgEx function specifies which window point maps to the viewport origin (0,0).

## [SetWinMetaFileBits](#)

The SetWinMetaFileBits function converts a metafile from the older Windows format to the new enhanced format and stores the new metafile in memory.

#### [SetWorldTransform](#)

The SetWorldTransform function sets a two-dimensional linear transformation between world space and page space for the specified device context. This transformation can be used to scale, rotate, shear, or translate graphics output.

#### [StartDocA](#)

The StartDoc function starts a print job. (ANSI)

#### [StartDocW](#)

The StartDoc function starts a print job. (Unicode)

#### [StartPage](#)

The StartPage function prepares the printer driver to accept data.

#### [StretchBlt](#)

The StretchBlt function copies a bitmap from a source rectangle into a destination rectangle, stretching or compressing the bitmap to fit the dimensions of the destination rectangle, if necessary.

#### [StretchDIBits](#)

The StretchDIBits function copies the color data for a rectangle of pixels in a DIB, JPEG, or PNG image to the specified destination rectangle.

#### [StrokeAndFillPath](#)

The StrokeAndFillPath function closes any open figures in a path, strokes the outline of the path by using the current pen, and fills its interior by using the current brush.

#### [StrokePath](#)

The StrokePath function renders the specified path by using the current pen.

#### [SwapBuffers](#)

The SwapBuffers function exchanges the front and back buffers if the current pixel format for the window referenced by the specified device context includes a back buffer.

#### [TextOutA](#)

The TextOut function writes a character string at the specified location, using the currently selected font, background color, and text color. (ANSI)

#### [TextOutW](#)

The TextOut function writes a character string at the specified location, using the currently selected font, background color, and text color. (Unicode)

#### [TranslateCharsetInfo](#)

Translates character set information and sets all members of a destination structure to appropriate values.

#### [TransparentBlt](#)

The TransparentBlt function performs a bit-block transfer of the color data corresponding to a rectangle of pixels from the specified source device context into a destination device context.

#### [UnrealizeObject](#)

The UnrealizeObject function resets the origin of a brush or resets a logical palette.

#### [UpdateColors](#)

The UpdateColors function updates the client area of the specified device context by remapping the current colors in the client area to the currently realized logical palette.

#### [UpdateICMRegKeyA](#)

The UpdateICMRegKey function manages color profiles and Color Management Modules in the system. (ANSI)

#### [UpdateICMRegKeyW](#)

The UpdateICMRegKey function manages color profiles and Color Management Modules in the system. (Unicode)

#### [wglCopyContext](#)

The wglCopyContext function copies selected groups of rendering states from one OpenGL rendering context to another.

#### [wglCreateContext](#)

The wglCreateContext function creates a new OpenGL rendering context, which is suitable for drawing on the device referenced by hdc. The rendering context has the same pixel format as the device context.

#### [wglCreateLayerContext](#)

The `wglCreateLayerContext` function creates a new OpenGL rendering context for drawing to a specified layer plane on a device context.

#### [wglDeleteContext](#)

The `wglDeleteContext` function deletes a specified OpenGL rendering context.

#### [wglDescribeLayerPlane](#)

The `wglDescribeLayerPlane` function obtains information about the layer planes of a given pixel format.

#### [wglGetCurrentContext](#)

The `wglGetCurrentContext` function obtains a handle to the current OpenGL rendering context of the calling thread.

#### [wglGetCurrentDC](#)

The `wglGetCurrentDC` function obtains a handle to the device context that is associated with the current OpenGL rendering context of the calling thread.

#### [wglGetLayerPaletteEntries](#)

Retrieves the palette entries from a given color-index layer plane for a specified device context.

#### [wglGetProcAddress](#)

The `wglGetProcAddress` function returns the address of an OpenGL extension function for use with the current OpenGL rendering context.

#### [wglMakeCurrent](#)

The `wglMakeCurrent` function makes a specified OpenGL rendering context the calling thread's current rendering context.

#### [wglRealizeLayerPalette](#)

The `wglRealizeLayerPalette` function maps palette entries from a given color-index layer plane into the physical palette or initializes the palette of an RGBA layer plane.

#### [wglSetLayerPaletteEntries](#)

Sets the palette entries in a given color-index layer plane for a specified device context.

#### [wglShareLists](#)

The `wglShareLists` function enables multiple OpenGL rendering contexts to share a single display-list space.

## wglSwapLayerBuffers

The wglSwapLayerBuffers function swaps the front and back buffers in the overlay, underlay, and main planes of the window referenced by a specified device context.

## wglUseFontBitmapsA

The wglUseFontBitmaps function creates a set of bitmap display lists for use in the current OpenGL rendering context. (ANSI)

## wglUseFontBitmapsW

The wglUseFontBitmaps function creates a set of bitmap display lists for use in the current OpenGL rendering context. (Unicode)

## wglUseFontOutlinesA

The wglUseFontOutlines function creates a set of display lists, one for each glyph of the currently selected outline font of a device context, for use with the current rendering context. (ANSI)

## wglUseFontOutlinesW

The wglUseFontOutlines function creates a set of display lists, one for each glyph of the currently selected outline font of a device context, for use with the current rendering context. (Unicode)

## WidenPath

The WidenPath function redefines the current path as the area that would be painted if the path were stroked using the pen currently selected into the given device context.

# Callback functions

[+] Expand table

## ABORTPROC

The AbortProc function is an application-defined callback function used with the SetAbortProc function.

## ENHMFENUMPROC

The EnhMetaFileProc function is an application-defined callback function used with the EnumEnhMetaFile function.

## [GOBJENUMPROC](#)

The `EnumObjectsProc` function is an application-defined callback function used with the `EnumObjects` function.

## [ICMENUMPROCA](#)

The `EnumICMProfilesProcCallback` callback is an application-defined callback function that processes color profile data from `EnumICMProfiles`. (ANSI)

## [ICMENUMPROCW](#)

The `EnumICMProfilesProcCallback` callback is an application-defined callback function that processes color profile data from `EnumICMProfiles`. (Unicode)

## [LINEDDAPROC](#)

The `LineDDAProc` function is an application-defined callback function used with the `LineDDA` function.

## [MFENUMPROC](#)

The `EnumMetaFileProc` function is an application-defined callback function that processes Windows-format metafile records.

# Structures

[Expand table](#)

## [ABC](#)

The ABC structure contains the width of a character in a TrueType font.

## [ABCFLOAT](#)

The ABCFLOAT structure contains the A, B, and C widths of a font character.

## [AXESLISTA](#)

The AXESLIST structure contains information on all the axes of a multiple master font. (ANSI)

## [AXESLISTW](#)

The AXESLIST structure contains information on all the axes of a multiple master font. (Unicode)

## [AXISINFOA](#)

The AXISINFO structure contains information about an axis of a multiple master font. (ANSI)

## [AXISINFOW](#)

The AXISINFO structure contains information about an axis of a multiple master font. (Unicode)

## [BITMAP](#)

The BITMAP structure defines the type, width, height, color format, and bit values of a bitmap.

## [BITMAPCOREHEADER](#)

The BITMAPCOREHEADER structure contains information about the dimensions and color format of a DIB.

## [BITMAPCOREINFO](#)

The BITMAPCOREINFO structure defines the dimensions and color information for a DIB.

## [BITMAPFILEHEADER](#)

The BITMAPFILEHEADER structure contains information about the type, size, and layout of a file that contains a DIB.

## [BITMAPINFO](#)

The BITMAPINFO structure defines the dimensions and color information for a DIB.

## [BITMAPINFOHEADER](#)

The BITMAPINFOHEADER structure contains information about the dimensions and color format of a device-independent bitmap (DIB).

## [BITMAPV4HEADER](#)

The BITMAPV4HEADER structure is the bitmap information header file. It is an extended version of the BITMAPINFOHEADER structure. Applications can use the BITMAPV5HEADER structure for added functionality.

## [BITMAPV5HEADER](#)

The BITMAPV5HEADER structure is the bitmap information header file. It is an extended version of the BITMAPINFOHEADER structure.

## [BLENDFUNCTION](#)

The **BLENDFUNCTION** structure controls blending by specifying the blending functions for source and destination bitmaps.

#### [CHARSETINFO](#)

Contains information about a character set.

#### [CIEXYZ](#)

The **CIEXYZ** structure contains the x,y, and z coordinates of a specific color in a specified color space.

#### [CIEXYZTRIPLE](#)

The **CIEXYZTRIPLE** structure contains the x,y, and z coordinates of the three colors that correspond to the red, green, and blue endpoints for a specified logical color space.

#### [COLORADJUSTMENT](#)

The **COLORADJUSTMENT** structure defines the color adjustment values used by the **StretchBlt** and **StretchDIBits** functions when the stretch mode is **HALFTONE**. You can set the color adjustment values by calling the **SetColorAdjustment** function.

#### [DESIGNVECTOR](#)

The **DESIGNVECTOR** structure is used by an application to specify values for the axes of a multiple master font.

#### [DEVMODEA](#)

The **DEVMODE** data structure contains information about the initialization and environment of a printer or a display device.

#### [DEVMODEW](#)

The **DEVMODEW** structure is used for specifying characteristics of display and print devices in the Unicode (wide) character set.

#### [DIBSECTION](#)

The **DIBSECTION** structure contains information about a DIB created by calling the **CreateDIBSection** function.

#### [DISPLAY\\_DEVICEA](#)

The **DISPLAY\_DEVICE** structure receives information about the display device specified by the **iDevNum** parameter of the **EnumDisplayDevices** function. (ANSI)

#### [DISPLAY\\_DEVICEW](#)

The DISPLAY\_DEVICE structure receives information about the display device specified by the iDevNum parameter of the EnumDisplayDevices function. (Unicode)

#### [DISPLAYCONFIG\\_2DREGION](#)

The DISPLAYCONFIG\_2DREGION structure represents a point or an offset in a two-dimensional space.

#### [DISPLAYCONFIG\\_ADAPTER\\_NAME](#)

The DISPLAYCONFIG\_ADAPTER\_NAME structure contains information about the display adapter.

#### [DISPLAYCONFIG\\_DESKTOP\\_IMAGE\\_INFO](#)

The DISPLAYCONFIG\_DESKTOP\_IMAGE\_INFO structure contains information about the image displayed on the desktop.

#### [DISPLAYCONFIG\\_DEVICE\\_INFO\\_HEADER](#)

The DISPLAYCONFIG\_DEVICE\_INFO\_HEADER structure contains display information about the device.

#### [DISPLAYCONFIG\\_MODE\\_INFO](#)

The DISPLAYCONFIG\_MODE\_INFO structure contains either source mode or target mode information.

#### [DISPLAYCONFIG\\_PATH\\_INFO](#)

The DISPLAYCONFIG\_PATH\_INFO structure is used to describe a single path from a target to a source.

#### [DISPLAYCONFIG\\_PATH\\_SOURCE\\_INFO](#)

The DISPLAYCONFIG\_PATH\_SOURCE\_INFO structure contains source information for a single path.

#### [DISPLAYCONFIG\\_PATH\\_TARGET\\_INFO](#)

The DISPLAYCONFIG\_PATH\_TARGET\_INFO structure contains target information for a single path.

#### [DISPLAYCONFIG\\_RATIONAL](#)

The DISPLAYCONFIG\_RATIONAL structure describes a fractional value that represents vertical and horizontal frequencies of a video mode (that is, vertical sync and horizontal sync).

#### [DISPLAYCONFIG\\_SDR\\_WHITE\\_LEVEL](#)

The DISPLAYCONFIG\_SDR\_WHITE\_LEVEL structure (wingdi.h) contains information about a display's current SDR white level.

## [DISPLAYCONFIG\\_SET\\_TARGET\\_PERSISTENCE](#)

The DISPLAYCONFIG\_SET\_TARGET\_PERSISTENCE structure contains information about setting the display.

## [DISPLAYCONFIG\\_SOURCE\\_DEVICE\\_NAME](#)

The DISPLAYCONFIG\_SOURCE\_DEVICE\_NAME structure contains the GDI device name for the source or view.

## [DISPLAYCONFIG\\_SOURCE\\_MODE](#)

The DISPLAYCONFIG\_SOURCE\_MODE structure represents a point or an offset in a two-dimensional space.

## [DISPLAYCONFIG\\_SUPPORT\\_VIRTUAL\\_RESOLUTION](#)

The DISPLAYCONFIG\_SUPPORT\_VIRTUAL\_RESOLUTION structure contains information on the state of virtual resolution support for the monitor.

## [DISPLAYCONFIG\\_TARGET\\_BASE\\_TYPE](#)

Specifies base output technology info for a given target ID.

## [DISPLAYCONFIG\\_TARGET\\_DEVICE\\_NAME](#)

The DISPLAYCONFIG\_TARGET\_DEVICE\_NAME structure contains information about the target.

## [DISPLAYCONFIG\\_TARGET\\_DEVICE\\_NAME\\_FLAGS](#)

The DISPLAYCONFIG\_TARGET\_DEVICE\_NAME\_FLAGS structure contains information about a target device.

## [DISPLAYCONFIG\\_TARGET\\_MODE](#)

The DISPLAYCONFIG\_TARGET\_MODE structure describes a display path target mode.

## [DISPLAYCONFIG\\_TARGET\\_PREFERRED\\_MODE](#)

The DISPLAYCONFIG\_TARGET\_PREFERRED\_MODE structure contains information about the preferred mode of a display.

## [DISPLAYCONFIG\\_VIDEO\\_SIGNAL\\_INFO](#)

The DISPLAYCONFIG\_VIDEO\_SIGNAL\_INFO structure contains information about the video signal for a display.

## [DOCINFOA](#)

The DOCINFO structure contains the input and output file names and other information used by the StartDoc function. (ANSI)

#### [DOCINFOW](#)

The DOCINFO structure contains the input and output file names and other information used by the StartDoc function. (Unicode)

#### [DRAWPATRECT](#)

The DRAWPATRECT structure defines a rectangle to be created.

#### [EMR](#)

The EMR structure provides the base structure for all enhanced metafile records. An enhanced metafile record contains the parameters for a specific GDI function used to create part of a picture in an enhanced format metafile.

#### [EMRABORTPATH](#)

Contains data for the AbortPath, BeginPath, EndPath, CloseFigure, FlattenPath, WidenPath, SetMetaRgn, SaveDC, and RealizePalette enhanced metafile records.

#### [EMRALPHABLEND](#)

The EMRALPHABLEND structure contains members for the AlphaBlend enhanced metafile record.

#### [EMRANGLEARC](#)

The EMRANGLEARC structure contains members for the AngleArc enhanced metafile record.

#### [EMRARC](#)

The EMRARC, EMRARCTO, EMRCHORD, and EMRPIE structures contain members for the Arc, ArcTo, Chord, and Pie enhanced metafile records.

#### [EMRBITBLT](#)

The EMRBITBLT structure contains members for the BitBlt enhanced metafile record. Note that graphics device interface (GDI) converts the device-dependent bitmap into a device-independent bitmap (DIB) before storing it in the metafile record.

#### [EMRCOLORCORRECTPALETTE](#)

The EMRCOLORCORRECTPALETTE structure contains members for the ColorCorrectPalette enhanced metafile record.

#### [EMRCOLORMATCHTOTARGET](#)

The EMRCOLORMATCHTOTARGET structure contains members for the ColorMatchToTarget enhanced metafile record.

#### [EMRCREATEBRUSHINDIRECT](#)

The EMRCREATEBRUSHINDIRECT structure contains members for the CreateBrushIndirect enhanced metafile record.

#### [EMRCREATECOLORSPACE](#)

The EMRCREATECOLORSPACE structure contains members for the CreateColorSpace enhanced metafile record.

#### [EMRCREATECOLORSPACEW](#)

The EMRCREATECOLORSPACEW structure contains members for the CreateColorSpace enhanced metafile record. It differs from EMRCREATECOLORSPACE in that it has a Unicode logical color space and also has an optional array containing raw source profile data.

#### [EMRCREATEDIBPATTERNBRUSHPT](#)

The EMRCREATEDIBPATTERNBRUSHPT structure contains members for the CreateDIBPatternBrushPt enhanced metafile record. The BITMAPINFO structure is followed by the bitmap bits that form a packed device-independent bitmap (DIB).

#### [EMRCREATEMONOBRUSH](#)

The EMRCREATEMONOBRUSH structure contains members for the CreatePatternBrush (when passed a monochrome bitmap) or CreateDIBPatternBrush (when passed a monochrome DIB) enhanced metafile records.

#### [EMRCREATEPALETTE](#)

The EMRCREATEPALETTE structure contains members for the CreatePalette enhanced metafile record.

#### [EMRCREATEPEN](#)

The EMRCREATEPEN structure contains members for the CreatePen enhanced metafile record.

#### [EMRELLIPSE](#)

The EMRELLIPSE and EMRRECTANGLE structures contain members for the Ellipse and Rectangle enhanced metafile records.

#### [EMREOF](#)

The EMREOF structure contains data for the enhanced metafile record that indicates the end of the metafile.

## [EMREXCLUDECLIPRECT](#)

The EMREXCLUDECLIPRECT and EMRINTERSECTCLIPRECT structures contain members for the ExcludeClipRect and IntersectClipRect enhanced metafile records.

## [EMREXTCREATEFONTINDIRECTW](#)

The EMREXTCREATEFONTINDIRECTW structure contains members for the CreateFontIndirect enhanced metafile record.

## [EMREXTCREATEPEN](#)

The EMREXTCREATEPEN structure contains members for the ExtCreatePen enhanced metafile record. If the record contains a BITMAPINFO structure, it is followed by the bitmap bits that form a packed device-independent bitmap (DIB).

## [EMREXTFLOODFILL](#)

The EMREXTFLOODFILL structure contains members for the ExtFloodFill enhanced metafile record.

## [EMREXTSELECTCLIPRGN](#)

The EMREXTSELECTCLIPRGN structure contains members for the ExtSelectClipRgn enhanced metafile record.

## [EMREXTTEXTOUTA](#)

The EMREXTTEXTOUTA and EMREXTTEXTOUTW structures contain members for the ExtTextOut, TextOut, or DrawText enhanced metafile records.

## [EMRFILLPATH](#)

The EMRFILLPATH,♦EMRSTROKEANDFILLPATH,♦ and EMRSTROKEPATH structures contain members for the FillPath, StrokeAndFillPath, and StrokePath enhanced metafile records.

## [EMRFILLRGN](#)

The EMRFILLRGN structure contains members for the FillRgn enhanced metafile record.

## [EMRFORMAT](#)

The EMRFORMAT structure contains information that identifies graphics data in an enhanced metafile. A GDICOMMENT\_MULTIFORMATS enhanced metafile public comment contains an array of EMRFORMAT structures.

## [EMRFRAMERGN](#)

The EMRFRAMERGN structure contains members for the FrameRgn enhanced metafile record.

## [EMRGDICONMENT](#)

The EMRGDICONMENT structure contains application-specific data.

## [EMRGLSBOUNDEDRECORD](#)

The EMRGLSBOUNDEDRECORD structure contains members for an enhanced metafile record generated by OpenGL functions. It contains data for OpenGL functions with information in pixel units that must be scaled when playing the metafile.

## [EMRGLSRECORD](#)

The EMRGLSRECORD structure contains members for an enhanced metafile record generated by OpenGL functions. It contains data for OpenGL functions that scale automatically to the OpenGL viewport.

## [EMRGRADIENTFILL](#)

The EMRGRADIENTFILL structure contains members for the GradientFill enhanced metafile record.

## [EMRINVERTRGN](#)

The EMRINVERTRGN and EMRPAINTRGN structures contain members for the InvertRgn and PaintRgn enhanced metafile records.

## [EMRLINETO](#)

The EMRLINETO and EMRMOVETOEX structures contains members for the LineTo and MoveToEx enhanced metafile records.

## [EMRMASKBLT](#)

The EMRMASKBLT structure contains members for the MaskBlt enhanced metafile record. Note that graphics device interface (GDI) converts the device-dependent bitmap into a device-independent bitmap (DIB) before storing it in the metafile record.

## [EMRMODIFYWORLDTRANSFORM](#)

The EMRMODIFYWORLDTRANSFORM structure contains members for the ModifyWorldTransform enhanced metafile record.

## [EMROFFSETCLIPRGN](#)

The EMROFFSETCLIPRGN structure contains members for the OffsetClipRgn enhanced metafile record.

## [EMRPIXELFORMAT](#)

The EMRPIXELFORMAT structure contains the members for the SetPixelFormat enhanced metafile record. The pixel format information in ENHMETAHEADER refers to this structure.

#### [EMRPLGBT](#)

The EMRPLGBT structure contains members for the PlgBlt enhanced metafile record. Note that graphics device interface (GDI) converts the device-dependent bitmap into a device-independent bitmap (DIB) before storing it in the metafile record.

#### [EMRPOLYDRAW](#)

The EMRPOLYDRAW structure contains members for the PolyDraw enhanced metafile record.

#### [EMRPOLYDRAW16](#)

The EMRPOLYDRAW16 structure contains members for the PolyDraw enhanced metafile record.

#### [EMRPOLYLINE](#)

The EMRPOLYLINE, EMRPOLYBEZIER, EMRPOLYGON, EMRPOLYBEZIERTO, and EMRPOLYLINETO structures contain members for the Polyline, PolyBezier, Polygon, PolyBezierTo, and PolylineTo enhanced metafile records.

#### [EMRPOLYLINE16](#)

The EMRPOLYLINE16, EMRPOLYBEZIER16, EMRPOLYGON16, EMRPOLYBEZIERTO16, and EMRPOLYLINETO16 structures contain members for the Polyline, PolyBezier, Polygon, PolyBezierTo, and PolylineTo enhanced metafile records.

#### [EMRPOLYPOLYLINE](#)

The EMRPOLYPOLYLINE and EMRPOLYPOLYGON structures contain members for the PolyPolyline and PolyPolygon enhanced metafile records.

#### [EMRPOLYPOLYLINE16](#)

The EMRPOLYPOLYLINE16 and EMRPOLYPOLYGON16 structures contain members for the PolyPolyline and PolyPolygon enhanced metafile records.

#### [EMRPOLYTEXTOUTA](#)

The EMRPOLYTEXTOUTA and EMRPOLYTEXTOUTW structures contain members for the PolyTextOut enhanced metafile record.

#### [EMRRESIZEPALETTE](#)

The EMRRESIZEPALETTE structure contains members for the ResizePalette enhanced metafile record.

## [EMRRESTOREDC](#)

The EMRRESTOREDC structure contains members for the RestoreDC enhanced metafile record.

## [EMRROUNDRRECT](#)

The EMRROUNDRRECT structure contains members for the RoundRect enhanced metafile record.

## [EMRSCALEVIEWPORTEXTEX](#)

The EMRSCALEVIEWPORTEXTEX and EMRSCALEWINDOWEXTEX structures contain members for the ScaleViewportExtEx and ScaleWindowExtEx enhanced metafile records.

## [EMRSELECTCLIPPATH](#)

Contains parameters for the SelectClipPath, SetBkMode, SetMapMode, SetPolyFillMode, SetROP2, SetStretchBltMode, SetTextAlign, SetICMMode , and SetLayout enhanced metafile records.

## [EMRSELECTOBJECT](#)

The EMRSELECTOBJECT and EMRDELETEOBJECT structures contain members for the SelectObject and DeleteObject enhanced metafile records.

## [EMRSELECTPALETTE](#)

The EMRSELECTPALETTE structure contains members for the SelectPalette enhanced metafile record. Note that the bForceBackground parameter in SelectPalette is always recorded as TRUE, which causes the palette to be realized as a background palette.

## [EMRSETARCDIRECTION](#)

The EMRSETARCDIRECTION structure contains members for the SetArcDirection enhanced metafile record.

## [EMRSETBKCOLOR](#)

The EMRSETBKCOLOR and EMRSETTEXTCOLOR structures contain members for the SetBkColor and SetTextColor enhanced metafile records.

## [EMRSETCOLORADJUSTMENT](#)

The EMRSETCOLORADJUSTMENT structure contains members for the SetColorAdjustment enhanced metafile record.

## [EMRSETCOLORSPACE](#)

The EMRSETCOLORSPACE, EMRSELECTCOLORSPACE, and EMRDELETECOLORSPACE structures contain members for the SetColorSpace and DeleteColorSpace enhanced metafile records.

## [EMRSETDIBITSTODEVICE](#)

The EMRSETDIBITSTODEVICE structure contains members for the SetDIBitsToDevice enhanced metafile record.

## [EMRSETICMPROFILE](#)

The EMRSETICMPROFILE structure contains members for the SetICMProfile enhanced metafile record.

## [EMRSETMAPPERFLAGS](#)

The EMRSETMAPPERFLAGS structure contains members for the SetMapperFlags enhanced metafile record.

## [EMRSETMITERLIMIT](#)

The EMRSETMITERLIMIT structure contains members for the SetMiterLimit enhanced metafile record.

## [EMRSETPALETTEENTRIES](#)

The EMRSETPALETTEENTRIES structure contains members for the SetPaletteEntries enhanced metafile record.

## [EMRSETPIXELV](#)

The EMRSETPIXELV structure contains members for the SetPixelV enhanced metafile record. When an enhanced metafile is created, calls to SetPixel are also recorded in this record.

## [EMRSETVIEWPORTEXTEX](#)

The EMRSETVIEWPORTEXTEX and EMRSETWINDOWEXTEX structures contains members for the SetViewportExtEx and SetWindowExtEx enhanced metafile records.

## [EMRSETVIEWPORTORGEX](#)

The EMRSETVIEWPORTORGEX, EMRSETWINDOWORGEX, and EMRSETBRUSHORGEX structures contain members for the SetViewportOrgEx, SetWindowOrgEx, and SetBrushOrgEx enhanced metafile records.

## [EMRSETWORLDTRANSFORM](#)

The EMRSETWORLDTRANSFORM structure contains members for the SetWorldTransform enhanced metafile record.

## [EMRSTRETCHBLT](#)

The EMRSTRETCHBLT structure contains members for the StretchBlt enhanced metafile record.

Note that graphics device interface (GDI) converts the device-dependent bitmap into a device-independent bitmap (DIB) before storing it in the metafile record.

#### [EMRSTRETCHDIBITS](#)

The EMRSTRETCHDIBITS structure contains members for the StretchDIBits enhanced metafile record.

#### [EMRTEXT](#)

The EMRTEXT structure contains members for text output.

#### [EMRTRANSPARENTBLT](#)

The EMRTRANSPARENTBLT structure contains members for the TransparentBLT enhanced metafile record.

#### [ENHMETAHEADER](#)

The ENHMETAHEADER structure contains enhanced-metafile data such as the dimensions of the picture stored in the enhanced metafile, the count of records in the enhanced metafile, the resolution of the device on which the picture was created, and so on. This structure is always the first record in an enhanced metafile.

#### [ENHMETARECORD](#)

The ENHMETARECORD structure contains data that describes a graphics device interface (GDI) function used to create part of a picture in an enhanced-format metafile.

#### [ENUMLOGFONTA](#)

The ENUMLOGFONT structure defines the attributes of a font, the complete name of a font, and the style of a font. (ANSI)

#### [ENUMLOGFONTEXA](#)

The ENUMLOGFONTEX structure contains information about an enumerated font. (ANSI)

#### [ENUMLOGFONTEXDV](#)

The ENUMLOGFONTEXDV structure contains the information used to create a font. (ANSI)

#### [ENUMLOGFONTEXDW](#)

The ENUMLOGFONTEXDV structure contains the information used to create a font. (Unicode)

#### [ENUMLOGFONTEXW](#)

The ENUMLOGFONTEX structure contains information about an enumerated font. (Unicode)

## [ENUMLOGFONTW](#)

The ENUMLOGFONT structure defines the attributes of a font, the complete name of a font, and the style of a font. (Unicode)

## [ENUMTEXTMETRICA](#)

The ENUMTEXTMETRIC structure contains information about a physical font. (ANSI)

## [ENUMTEXTMETRICW](#)

The ENUMTEXTMETRIC structure contains information about a physical font. (Unicode)

## [EXTLOGFONTA](#)

The EXTLOGFONT structure defines the attributes of a font. (ANSI)

## [EXTLOGFONTW](#)

The EXTLOGFONT structure defines the attributes of a font. (Unicode)

## [EXTLOGOPEN](#)

The EXTLOGOPEN structure defines the pen style, width, and brush attributes for an extended pen.

## [FIXED](#)

The FIXED structure contains the integral and fractional parts of a fixed-point real number.

## [FONTSIGNATURE](#)

Contains information identifying the code pages and Unicode subranges for which a given font provides glyphs.

## [GCP\\_RESULTSA](#)

The GCP\_RESULTS structure contains information about characters in a string. This structure receives the results of the GetCharacterPlacement function. For some languages, the first element in the arrays may contain more, language-dependent information. (ANSI)

## [GCP\\_RESULTSW](#)

The GCP\_RESULTS structure contains information about characters in a string. This structure receives the results of the GetCharacterPlacement function. For some languages, the first element in the arrays may contain more, language-dependent information. (Unicode)

## [GLYPHMETRICS](#)

The GLYPHMETRICS structure contains information about the placement and orientation of a glyph in a character cell.

#### [GLYPHMETRICSFLOAT](#)

The GLYPHMETRICSFLOAT structure contains information about the placement and orientation of a glyph in a character cell.

#### [GLYPHSET](#)

The GLYPHSET structure contains information about a range of Unicode code points.

#### [GRADIENT\\_RECT](#)

The GRADIENT\_RECT structure specifies the index of two vertices in the pVertex array in the GradientFill function. These two vertices form the upper-left and lower-right boundaries of a rectangle.

#### [GRADIENT\\_TRIANGLE](#)

The GRADIENT\_TRIANGLE structure specifies the index of three vertices in the pVertex array in the GradientFill function. These three vertices form one triangle.

#### [HANDLETABLE](#)

The HANDLETABLE structure is an array of handles, each of which identifies a graphics device interface (GDI) object.

#### [KERNINGPAIR](#)

The KERNINGPAIR structure defines a kerning pair.

#### [LAYERPLANEDESCRIPTOR](#)

The LAYERPLANEDESCRIPTOR structure describes the pixel format of a drawing surface.

#### [LOCALESIGNATURE](#)

Contains extended font signature information, including two code page bitfields (CPBs) that define the default and supported character sets and code pages. This structure is typically used to represent the relationships between font coverage and locales.

#### [LOGBRUSH](#)

The LOGBRUSH structure defines the style, color, and pattern of a physical brush. It is used by the CreateBrushIndirect and ExtCreatePen functions.

#### [LOGBRUSH32](#)

The LOGBRUSH32 structure defines the style, color, and pattern of a physical brush.

## [LOGCOLORSPACE](#)

The LOGCOLORSPACE structure contains information that defines a logical color space. (ANSI)

## [LOGCOLORSPACEW](#)

The LOGCOLORSPACE structure contains information that defines a logical color space. (Unicode)

## [LOGFONTA](#)

The LOGFONT structure defines the attributes of a font. (ANSI)

## [LOGFONTW](#)

The LOGFONT structure defines the attributes of a font. (Unicode)

## [LOGPALETTE](#)

The LOGPALETTE structure defines a logical palette.

## [LOGPEN](#)

The LOGPEN structure defines the style, width, and color of a pen. The CreatePenIndirect function uses the LOGPEN structure.

## [MAT2](#)

The MAT2 structure contains the values for a transformation matrix used by the GetGlyphOutline function.

## [METAFILEPICT](#)

Defines the metafile picture format used for exchanging metafile data through the clipboard.

## [METAHEADER](#)

The METAHEADER structure contains information about a Windows-format metafile.

## [METARECORD](#)

The METARECORD structure contains a Windows-format metafile record.

## [NEWTEXTMETRICA](#)

The NEWTEXTMETRIC structure contains data that describes a physical font. (ANSI)

## [NEWTEXTMETRICEXA](#)

The NEWTEXTMETRICEX structure contains information about a physical font. (ANSI)

#### [NEWTEXTMETRICEXW](#)

The NEWTEXTMETRICEX structure contains information about a physical font. (Unicode)

#### [NEWTEXTMETRICW](#)

The NEWTEXTMETRIC structure contains data that describes a physical font. (Unicode)

#### [OUTLINETEXTMETRICA](#)

The OUTLINETEXTMETRIC structure contains metrics describing a TrueType font. (ANSI)

#### [OUTLINETEXTMETRICW](#)

The OUTLINETEXTMETRIC structure contains metrics describing a TrueType font. (Unicode)

#### [PALETTEENTRY](#)

Specifies the color and usage of an entry in a logical palette.

#### [PANOSE](#)

The PANOSE structure describes the PANOSE font-classification values for a TrueType font. These characteristics are then used to associate the font with other fonts of similar appearance but different names.

#### [PIXELFORMATDESCRIPTOR](#)

The PIXELFORMATDESCRIPTOR structure describes the pixel format of a drawing surface.

#### [POINTFLOAT](#)

The POINTFLOAT structure contains the x and y coordinates of a point.

#### [POINTFX](#)

The POINTFX structure contains the coordinates of points that describe the outline of a character in a TrueType font.

#### [POLYTEXTA](#)

The POLYTEXT structure describes how the PolyTextOut function should draw a string of text. (ANSI)

#### [POLYTEXTW](#)

The POLYTEXT structure describes how the PolyTextOut function should draw a string of text. (Unicode)

#### [PSFEATURE\\_CUSTPAPER](#)

The PSFEATURE\_CUSTPAPER structure contains information about a custom paper size for a PostScript driver. This structure is used with the GET\_PS\_FEATURESETTING printer escape function.

#### [PSFEATURE\\_OUTPUT](#)

The PSFEATURE\_OUTPUT structure contains information about PostScript driver output options. This structure is used with the GET\_PS\_FEATURESETTING printer escape function.

#### [PSINJECTDATA](#)

The PSINJECTDATA structure is a header for the input buffer used with the POSTSCRIPT\_INJECTION printer escape function.

#### [RASTERIZER\\_STATUS](#)

The RASTERIZER\_STATUS structure contains information about whether TrueType is installed. This structure is filled when an application calls the GetRasterizerCaps function.

#### [RGBQUAD](#)

The RGBQUAD structure describes a color consisting of relative intensities of red, green, and blue.

#### [RGBTRIPLE](#)

The RGBTRIPLE structure describes a color consisting of relative intensities of red, green, and blue. The bmciColors member of the BITMAPCOREINFO structure consists of an array of RGBTRIPLE structures.

#### [RGNDATA](#)

The RGNDATA structure contains a header and an array of rectangles that compose a region. The rectangles are sorted top to bottom, left to right. They do not overlap.

#### [RGNDATAHEADER](#)

The RGNDATAHEADER structure describes the data returned by the GetRegionData function.

#### [TEXTMETRICA](#)

The TEXTMETRIC structure contains basic information about a physical font. All sizes are specified in logical units; that is, they depend on the current mapping mode of the display context. (ANSI)

#### [TEXTMETRICW](#)

The TEXTMETRIC structure contains basic information about a physical font. All sizes are specified

in logical units; that is, they depend on the current mapping mode of the display context.  
(Unicode)

## TRIVERTEX

The TRIVERTEX structure contains color information and position information.

## TTPOLYCURVE

The TTPOLYCURVE structure contains information about a curve in the outline of a TrueType character.

## TTPOLYGONHEADER

The TTPOLYGONHEADER structure specifies the starting position and type of a contour in a TrueType character outline.

## WCRANGE

The WCRANGE structure specifies a range of Unicode characters.

## XFORM

The XFORM structure specifies a world-space to page-space transformation.

# Enumerations

[+] Expand table

## DISPLAYCONFIG\_DEVICE\_INFO\_TYPE

The DISPLAYCONFIG\_DEVICE\_INFO\_TYPE enumeration specifies the type of display device info to configure or obtain through the DisplayConfigSetDeviceInfo or DisplayConfigGetDeviceInfo function.

## DISPLAYCONFIG\_MODE\_INFO\_TYPE

The DISPLAYCONFIG\_MODE\_INFO\_TYPE enumeration specifies that the information that is contained within the DISPLAYCONFIG\_MODE\_INFO structure is either source or target mode.

## DISPLAYCONFIG\_PIXELFORMAT

The DISPLAYCONFIG\_PIXELFORMAT enumeration specifies pixel format in various bits per pixel (BPP) values.

## [DISPLAYCONFIG\\_ROTATION](#)

The DISPLAYCONFIG\_ROTATION enumeration specifies the clockwise rotation of the display.

## [DISPLAYCONFIG\\_SCALING](#)

The DISPLAYCONFIG\_SCALING enumeration specifies the scaling transformation applied to content displayed on a video present network (VidPN) present path.

## [DISPLAYCONFIG\\_SCANLINE\\_ORDERING](#)

The DISPLAYCONFIG\_SCANLINE\_ORDERING enumeration specifies the method that the display uses to create an image on a screen.

## [DISPLAYCONFIG\\_TOPOLOGY\\_ID](#)

The DISPLAYCONFIG\_TOPOLOGY\_ID enumeration specifies the type of display topology.

## [DISPLAYCONFIG\\_VIDEO\\_OUTPUT\\_TECHNOLOGY](#)

The DISPLAYCONFIG\_VIDEO\_OUTPUT\_TECHNOLOGY enumeration specifies the target's connector type.

---

# Feedback

Was this page helpful?



[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# CheckColorsInGamut function (wingdi.h)

Article04/02/2021

The **CheckColorsInGamut** function determines whether a specified set of RGB triples lies in the output [gamut](#) of a specified device. The RGB triples are interpreted in the input logical color space.

## Syntax

C++

```
BOOL CheckColorsInGamut(
    HDC      hdc,
    LPRGBTRIPLE lpRGBTriple,
    LPVOID   dlpBuffer,
    DWORD    nCount
);
```

## Parameters

hdc

Handle to the device context whose output gamut to be checked.

lpRGBTriple

Pointer to an array of RGB triples to check.

dlpBuffer

Pointer to the buffer in which the results are to be placed. This buffer must be at least as large as *nCount* bytes.

nCount

The number of elements in the array of triples.

## Return value

If this function succeeds, the return value is a nonzero value.

If this function fails, the return value is zero.

## Remarks

The function places the test results in the buffer pointed to by *lpBuffer*. Each byte in the buffer corresponds to an *RGB triple*, and has an unsigned value between CM\_IN\_GAMUT (= 0) and CM\_OUT\_OF\_GAMUT (= 255). The value 0 denotes that the color is in gamut, while a nonzero value denotes that it is out of gamut. For any integer *n* such that  $0 < n < 255$ , a result value of *n* + 1 indicates that the corresponding color is at least as far out of gamut as would be indicated by a result value of *n*, as specified by the ICC Profile Format Specification. For more information on the ICC Profile Format Specification, see the sources listed in [Further information](#).

Note that for this function to succeed, WCS must be enabled for the device context handle that is passed in through the *hDC* parameter. WCS can be enabled for a device context handle by calling the [SetICMMode](#) function.

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	wingdi.h
Library	Gdi32.lib
DLL	Gdi32.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [SetICMMode](#)

---

## Feedback



Was this page helpful?  

[Get help at Microsoft Q&A](#)

# CIEXYZ structure (wingdi.h)

Article 02/22/2024

The **CIEXYZ** structure contains the *x*, *y*, and *z* coordinates of a specific color in a specified color space.

## Syntax

C++

```
typedef struct tagCIEXYZ {
    FXPT2DOT30 ciexyzX;
    FXPT2DOT30 ciexyzY;
    FXPT2DOT30 ciexyzZ;
} CIEXYZ;
```

## Members

`ciexyzX`

The *x* coordinate in fix point (2.30).

`ciexyzY`

The *y* coordinate in fix point (2.30).

`ciexyzZ`

The *z* coordinate in fix point (2.30).

## Requirements

[Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	wingdi.h

## See also

- Basic color management concepts
  - Structures
- 

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# CIEXYZTRIPLE structure (wingdi.h)

Article02/22/2024

The **CIEXYZTRIPLE** structure contains the *x*, *y*, and *z* coordinates of the three colors that correspond to the red, green, and blue endpoints for a specified logical color space.

## Syntax

C++

```
typedef struct tagCIEXYZTRIPLE {  
    CIEXYZ ciexyzRed;  
    CIEXYZ ciexyzGreen;  
    CIEXYZ ciexyzBlue;  
} CIEXYZTRIPLE;
```

## Members

`ciexyzRed`

The xyz coordinates of red endpoint.

`ciexyzGreen`

The xyz coordinates of green endpoint.

`ciexyzBlue`

The xyz coordinates of blue endpoint.

## Requirements

[ ] [Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Header	wingdi.h

## See also

- Basic color management concepts
  - Structures
- 

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# CMYK macro (wingdi.h)

The **CMYK** macro creates a CMYK color value by combining the specified cyan, magenta, yellow, and black values.

## Syntax

C++

```
COLORREF CMYK(  
    c,  
    m,  
    y,  
    k  
) ;
```

## Parameters

c

The cyan value for the color to be created.

m

The magenta value for the color to be created.

y

The yellow value for the color to be created.

k

The black value for the color to be created.

## Return value

Type: **COLORREF**

A CMYK color value.

## Requirements

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	wingdi.h

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [GetCValue](#)
- [GetKValue](#)
- [GetMValue](#)
- [GetYValue](#)
- [Macros for CMYK values and colors](#)

---

Last updated on 07/09/2025

# ColorCorrectPalette function (wingdi.h)

Article 02/22/2024

The **ColorCorrectPalette** function corrects the entries of a palette using the WCS 1.0 parameters in the specified device context.

## Syntax

C++

```
BOOL ColorCorrectPalette(  
    HDC      hdc,  
    HPALETTE hPal,  
    DWORD    deFirst,  
    DWORD    num  
) ;
```

## Parameters

hdc

Specifies a device context whose WCS parameters to use.

hPal

Specifies the handle to the palette to be color corrected.

deFirst

Specifies the first entry in the palette to be color corrected.

num

Specifies the number of entries to color correct.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**.

# Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	wingdi.h
Library	Gdi32.lib
DLL	Gdi32.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# ColorMatchToTarget function (wingdi.h)

Article04/02/2021

The **ColorMatchToTarget** function enables you to preview colors as they would appear on the target device.

## Syntax

C++

```
BOOL ColorMatchToTarget(
    HDC    hdc,
    HDC    hdcTarget,
    DWORD  action
);
```

## Parameters

hdc

Specifies the device context for previewing, generally the screen.

hdcTarget

Specifies the target device context, generally a printer.

action

A constant that can have one of the following values.

Value	Meaning
CS_ENABLE	Map the colors to the target device's color gamut. This enables color proofing. All subsequent draw commands to the DC will render colors as they would appear on the target device.
CS_DISABLE	Disable color proofing.
CS_DELETE_TRANSFORM	If color management is enabled for the target profile, disable it and delete the concatenated transform.

# Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**.

## Remarks

**ColorMatchToTarget** can be used to proof the colors of a color output device on another color output device. Setting the *uiAction* parameter to CS\_ENABLE causes all subsequent drawing commands to the DC to render colors as they would appear on the target device. If *uiAction* is set to CS\_DISABLE, proofing is turned off. However, the current color transform is not deleted from the DC. It is just inactive.

When **ColorMatchToTarget** is called, the color transform for the target device is performed first, and then the transform to the preview device is applied to the results of the first transform. This is used primarily for checking gamut mapping conditions. Before using this function, you must enable WCS for both device contexts.

This function cannot be cascaded. While color mapping to the target is enabled by setting *uiAction* to CS\_ENABLE, application changes to the color space or gamut mapping method are ignored. Those changes then take effect when color mapping to the target is disabled.

**Note** A memory leak will not occur if an application does not delete a transform using CS\_DELETE\_TRANSFORM. The transform will be deleted when either the device context (DC) is closed, or when the application color space is deleted. However if the transform is not going to be used again, or if the application will not be performing any more color matching on the DC, it should explicitly delete the transform to free the memory it occupies.

The *uiAction* parameter should only be set to CS\_DELETE\_TRANSFORM if color management is enabled before the **ColorMatchToTarget** function is called.

## Requirements

Minimum supported client

Windows 2000 Professional [desktop apps only]

Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	wingdi.h
Library	Gdi32.lib
DLL	Gdi32.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# CreateColorSpaceA function (wingdi.h)

Article 11/20/2024

The `CreateColorSpace` function creates a logical [color space](#).

## Syntax

C++

```
HCOLORSPACE CreateColorSpaceA(  
    LPLOGCOLORSPACEA lplcs  
) ;
```

## Parameters

`lplcs`

Pointer to the [LOGCOLORSPACE](#) data structure.

## Return value

If this function succeeds, the return value is a handle that identifies a color space.

If this function fails, the return value is `NULL`.

## Remarks

When the color space is no longer needed, use `DeleteColorSpace` to delete it.

**Windows 95/98/Me:** `CreateColorSpaceW` is supported by the Microsoft Layer for Unicode. To use this, you must add certain files to your application, as outlined in [Microsoft Layer for Unicode on Windows 95/98/Me Systems](#).

### Note

The wingdi.h header defines `CreateColorSpace` as an alias that automatically selects the ANSI or Unicode version of this function based on the definition of the `UNICODE` preprocessor constant. Mixing usage of the encoding-neutral alias with

code that is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

# Requirements

[+] Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	wingdi.h
Library	Gdi32.lib
DLL	Gdi32.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [DeleteColorSpaceW](#)

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) | [Get help at Microsoft Q&A](#)

# CreateColorSpaceW function (wingdi.h)

Article 11/20/2024

The `CreateColorSpace` function creates a logical [color space](#).

## Syntax

C++

```
HCOLORSPACE CreateColorSpaceW(
    LPLOGCOLORSPACEW lplcs
);
```

## Parameters

`lplcs`

Pointer to the [LOGCOLORSPACE](#) data structure.

## Return value

If this function succeeds, the return value is a handle that identifies a color space.

If this function fails, the return value is `NULL`.

## Remarks

When the color space is no longer needed, use `DeleteColorSpace` to delete it.

**Windows 95/98/Me:** `CreateColorSpaceW` is supported by the Microsoft Layer for Unicode. To use this, you must add certain files to your application, as outlined in [Microsoft Layer for Unicode on Windows 95/98/Me Systems](#).

### Note

The wingdi.h header defines `CreateColorSpace` as an alias that automatically selects the ANSI or Unicode version of this function based on the definition of the `UNICODE` preprocessor constant. Mixing usage of the encoding-neutral alias with

code that is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

# Requirements

[+] Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	wingdi.h
Library	Gdi32.lib
DLL	Gdi32.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [DeleteColorSpaceW](#)

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback](#) | [Get help at Microsoft Q&A](#)

# DeleteColorSpace function (wingdi.h)

Article02/22/2024

The [DeleteColorSpace](#) function removes and destroys a specified [color space](#).

## Syntax

C++

```
BOOL DeleteColorSpace(  
    HCOLORSPACE hcs  
) ;
```

## Parameters

`hcs`

Specifies the handle to a color space to delete.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**.

## Requirements

[+] [Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	wingdi.h
Library	Gdi32.lib
DLL	Gdi32.dll

## See also

- Basic color management concepts
  - Functions
- 

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# EnumICMProfilesA function (wingdi.h)

Article02/09/2023

The **EnumICMProfiles** function enumerates the different output color profiles that the system supports for a given device context.

## Syntax

C++

```
int EnumICMProfilesA(
    HDC         hdc,
    ICMENUMPROCA proc,
    LPARAM      param
);
```

## Parameters

hdc

Specifies the device context.

proc

Specifies the procedure instance address of a callback function defined by the application. (See [EnumICMProfilesProcCallback](#).)

param

Data supplied by the application that is passed to the callback function along with the color profile information.

## Return value

This function returns zero if the application interrupted the enumeration. The return value is -1 if there are no color profiles to enumerate. Otherwise, the return value is the last value returned by the callback function.

## Remarks

The **EnumICMProfiles** function returns a list of profiles that are associated with a device context (DC), and whose settings match those of the DC. It is possible for a device context to contain device profiles that are not associated with particular hardware devices, or device profiles that do not match the settings of the DC. The sRGB profile is an example. The [SetICMProfile](#) function is used to associate these types of profiles with a DC. The [GetICMProfile](#) function can be used to retrieve a profile that is not enumerated by the **EnumICMProfiles** function.

**Windows 95/98/Me:**[EnumICMProfilesW](#) is supported by the Microsoft Layer for Unicode. To use this, you must add certain files to your application, as outlined in [Microsoft Layer for Unicode on Windows 95/98/Me Systems](#).

 **Note**

The wingdi.h header defines **EnumICMProfiles** as an alias which automatically selects the ANSI or Unicode version of this function based on the definition of the **UNICODE** preprocessor constant. Mixing usage of the encoding-neutral alias with code that is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	wingdi.h
Library	Gdi32.lib
DLL	Gdi32.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [DeleteColorSpaceW](#)
- [ICMENUMPROCA callback function](#)

- [GetICMProfileW](#)
  - [SetICMProfileW](#)
- 

## Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# EnumICMProfilesW function (wingdi.h)

Article02/09/2023

The **EnumICMProfiles** function enumerates the different output color profiles that the system supports for a given device context.

## Syntax

C++

```
int EnumICMProfilesW(
    HDC         hdc,
    ICMENUMPROCW proc,
    LPARAM      param
);
```

## Parameters

hdc

Specifies the device context.

proc

Specifies the procedure instance address of a callback function defined by the application. (See [EnumICMProfilesProcCallback](#).)

param

Data supplied by the application that is passed to the callback function along with the color profile information.

## Return value

This function returns zero if the application interrupted the enumeration. The return value is -1 if there are no color profiles to enumerate. Otherwise, the return value is the last value returned by the callback function.

## Remarks

The **EnumICMProfiles** function returns a list of profiles that are associated with a device context (DC), and whose settings match those of the DC. It is possible for a device context to contain device profiles that are not associated with particular hardware devices, or device profiles that do not match the settings of the DC. The sRGB profile is an example. The [SetICMProfile](#) function is used to associate these types of profiles with a DC. The [GetICMProfile](#) function can be used to retrieve a profile that is not enumerated by the **EnumICMProfiles** function.

**Windows 95/98/Me:**[EnumICMProfilesW](#) is supported by the Microsoft Layer for Unicode. To use this, you must add certain files to your application, as outlined in [Microsoft Layer for Unicode on Windows 95/98/Me Systems](#).

 **Note**

The wingdi.h header defines **EnumICMProfiles** as an alias which automatically selects the ANSI or Unicode version of this function based on the definition of the **UNICODE** preprocessor constant. Mixing usage of the encoding-neutral alias with code that is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	wingdi.h
Library	Gdi32.lib
DLL	Gdi32.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [DeleteColorSpaceW](#)
- [ICMENUMPROCA callback function](#)

- [GetICMProfileW](#)
  - [SetICMProfileW](#)
- 

## Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# GetColorSpace function (wingdi.h)

Article04/02/2021

The **GetColorSpace** function retrieves the handle to the input color space from a specified device context.

## Syntax

C++

```
HCOLORSPACE GetColorSpace(  
    HDC hdc  
) ;
```

## Parameters

hdc

Specifies a device context that is to have its input color space handle retrieved.

## Return value

If the function succeeds, the return value is the current input color space handle.

If this function fails, the return value is **NULL**.

## Remarks

**GetColorSpace** obtains the handle to the input color space regardless of whether color management is enabled for the device context.

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows

Header	wingdi.h
Library	Gdi32.lib
DLL	Gdi32.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?



[Get help at Microsoft Q&A](#)

# GetCValue macro (wingdi.h)

The **GetCValue** macro retrieves the cyan color value from a CMYK color value.

## Syntax

C++

```
BYTE GetCValue(  
    cmyk  
)
```

## Parameters

cmyk

CMYK color value from which the cyan color value will be retrieved.

## Return value

Type: **BYTE**

The cyan color value from a CMYK color value.

## Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	wingdi.h

## See also

- [Basic color management concepts](#)
- [Functions](#)

- [CMYK](#)
  - [GetKValue](#)
  - [GetMValue](#)
  - [GetYValue](#)
  - [Macros for CMYK values and colors](#)
- 

Last updated on 07/09/2025

# GetDeviceGammaRamp function (wingdi.h)

Article 09/23/2022

The **GetDeviceGammaRamp** function gets the [gamma ramp](#) on direct color display boards having drivers that support downloadable gamma ramps in hardware.

## ⓘ Important

We strongly recommend that you don't use this API. Use of this API is subject to major limitations. See [SetDeviceGammaRamp](#) for more information.

## Syntax

C++

```
BOOL GetDeviceGammaRamp(
    HDC     hdc,
    LPVOID lpRamp
);
```

## Parameters

(hdc)

Specifies the device context of the direct color display board in question.

(lpRamp)

Points to a buffer where the function can place the current gamma ramp of the color display board. The gamma ramp is specified in three arrays of 256 **WORD** elements each, which contain the mapping between RGB values in the frame buffer and digital-analog-converter (DAC) values. The sequence of the arrays is red, green, blue.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**.

# Example

C++

```
WORD gArray[3][256];
GetDeviceGammaRamp(handle, gArray);
// `handle` is the device context. See GetDC for more details.
// `gArray` will hold the gamma array values in a 2-D array
```

## Remarks

Direct color display modes do not use color lookup tables and are usually 16, 24, or 32 bit. Not all direct color video boards support loadable gamma ramps.

**GetDeviceGammaRamp** succeeds only for devices with drivers that support downloadable gamma ramps in hardware.

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	wingdi.h
Library	Gdi32.lib
DLL	Gdi32.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

## Feedback

Was this page helpful?

 Yes

 No

Get help at Microsoft Q&A

# GetICMProfileA function (wingdi.h)

Article11/20/2024

The **GetICMProfile** function retrieves the file name of the current output color profile for a specified device context.

## Syntax

C++

```
BOOL GetICMProfileA(
    HDC     hdc,
    LPDWORD pBufSize,
    LPSTR   pszFilename
);
```

## Parameters

hdc

Specifies a device context from which to retrieve the color profile.

pBufSize

Pointer to a **DWORD** that contains the size of the buffer pointed to by *lpszFilename*. For the ANSI version of this function, the size is in bytes. For the Unicode version, the size is in WCHARs. If this function is successful, on return this parameter contains the size of the buffer actually used. However, if the buffer is not large enough, this function returns **FALSE**. In this case, the **GetLastError()** function returns **ERROR\_INSUFFICIENT\_BUFFER** and the **DWORD** pointed to by this parameter contains the size needed for the *lpszFilename* buffer.

pszFilename

Points to the buffer that receives the path name of the profile.

## Return value

If this function succeeds, the return value is **TRUE**. It also returns **TRUE** if the *lpszFilename* parameter is **NULL** and the size required for the buffer is copied into *lpchName*.

If this function fails, the return value is **FALSE**.

## Remarks

**GetICMProfile** obtains the file name of the current output profile regardless of whether or not color management is enabled for the device context.

Given a device context, **GetICMProfile** will output, through the parameter *lpszFilename*, the path name of the file containing the color profile currently being used by the device context. It will also output, through the parameter *lpcbName*, the length of the string containing the path name.

It is possible that the profile name returned by **GetICMProfile** will not be in the list of profiles returned by **EnumICMProfiles**. The **EnumICMProfiles** function returns all color space profiles that are associated with a device context (DC) whose settings match that of the DC. If the **SetICMProfile** function is used to set the current profile, a profile may be associated with the DC that does not match its settings. For instance, the **SetICMProfile** function can be used to associate the device-independent sRGB profile with a DC. This profile will be used as the current WCS profile for that DC, and calls to **GetICMProfile** will return its file name. However, the profile will not appear in the list of profiles that is returned from **EnumICMProfiles**.

If this function is called before any calls to the **SetICMProfile** function, it can be used to get the default profile for a device context.

**Windows 95/98/Me:** **GetICMProfileW** is supported by the Microsoft Layer for Unicode. To use this, you must add certain files to your application, as outlined in [Microsoft Layer for Unicode on Windows 95/98/Me Systems](#) ↗.

### ⓘ Note

The wingdi.h header defines **GetICMProfile** as an alias that automatically selects the ANSI or Unicode version of this function based on the definition of the **UNICODE** preprocessor constant. Mixing usage of the encoding-neutral alias with code that is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

## Requirements

[+] Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	wingdi.h
Library	Gdi32.lib
DLL	Gdi32.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [DeleteColorSpaceW](#)
- [ICMENUMPROCA callback function](#)
- [EnumICMProfilesW](#)
- [SetICMProfileW](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | Get help at Microsoft Q&A

# GetICMProfileW function (wingdi.h)

Article 02/09/2023

The **GetICMProfile** function retrieves the file name of the current output color profile for a specified device context.

## Syntax

C++

```
BOOL GetICMProfileW(
    HDC     hdc,
    LPDWORD pBufSize,
    LPWSTR  pszFilename
);
```

## Parameters

hdc

Specifies a device context from which to retrieve the color profile.

pBufSize

Pointer to a **DWORD** that contains the size of the buffer pointed to by *lpszFilename*. For the ANSI version of this function, the size is in bytes. For the Unicode version, the size is in WCHARs. If this function is successful, on return this parameter contains the size of the buffer actually used. However, if the buffer is not large enough, this function returns **FALSE**. In this case, the **GetLastError()** function returns **ERROR\_INSUFFICIENT\_BUFFER** and the **DWORD** pointed to by this parameter contains the size needed for the *lpszFilename* buffer.

pszFilename

Points to the buffer that receives the path name of the profile.

## Return value

If this function succeeds, the return value is **TRUE**. It also returns **TRUE** if the *lpszFilename* parameter is **NULL** and the size required for the buffer is copied into *lpcbName*.

If this function fails, the return value is **FALSE**.

## Remarks

**GetICMProfile** obtains the file name of the current output profile regardless of whether or not color management is enabled for the device context.

Given a device context, **GetICMProfile** will output, through the parameter *lpszFilename*, the path name of the file containing the color profile currently being used by the device context. It will also output, through the parameter *lpcbName*, the length of the string containing the path name.

It is possible that the profile name returned by **GetICMProfile** will not be in the list of profiles returned by [EnumICMProfiles](#). The **EnumICMProfiles** function returns all color space profiles that are associated with a device context (DC) whose settings match that of the DC. If the [SetICMProfile](#) function is used to set the current profile, a profile may be associated with the DC that does not match its settings. For instance, the **SetICMProfile** function can be used to associate the device-independent sRGB profile with a DC. This profile will be used as the current WCS profile for that DC, and calls to **GetICMProfile** will return its file name. However, the profile will not appear in the list of profiles that is returned from [EnumICMProfiles](#).

If this function is called before any calls to the **SetICMProfile** function, it can be used to get the default profile for a device context.

**Windows 95/98/Me:** **GetICMProfileW** is supported by the Microsoft Layer for Unicode. To use this, you must add certain files to your application, as outlined in [Microsoft Layer for Unicode on Windows 95/98/Me Systems](#).

### Note

The wingdi.h header defines **GetICMProfile** as an alias which automatically selects the ANSI or Unicode version of this function based on the definition of the UNICODE preprocessor constant. Mixing usage of the encoding-neutral alias with code that is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

## Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	wingdi.h
Library	Gdi32.lib
DLL	Gdi32.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [EnumICMProfilesW](#)
- [SetICMProfileW](#)

---

## Feedback

Was this page helpful?



[Get help at Microsoft Q&A](#)

# GetKValue macro (wingdi.h)

The `GetKValue` macro retrieves the black color value from a CMYK color value.

## Syntax

C++

```
BYTE GetKValue(  
    cmyk  
)
```

## Parameters

cmyk

CMYK color value from which the black color value will be retrieved.

## Return value

Type: `BYTE`

The black color value from a CMYK color value.

## Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	wingdi.h

## See also

- [Basic color management concepts](#)
- [Functions](#)

- [CMYK](#)
  - [GetCValue](#)
  - [GetMValue](#)
  - [GetYValue](#)
  - [Macros for CMYK values and colors](#)
- 

Last updated on 07/09/2025

# GetLogColorSpaceA function (wingdi.h)

Article11/20/2024

The **GetLogColorSpace** function retrieves the [color space](#) definition identified by a specified handle.

## Syntax

C++

```
BOOL GetLogColorSpaceA(
    HCOLORSPACE     hColorSpace,
    LPLOGCOLORSPACEA lpBuffer,
    DWORD           nSize
);
```

## Parameters

`hColorSpace`

Specifies the handle to a color space.

`lpBuffer`

Points to a buffer to receive the [LOGCOLORSPACE](#) structure.

`nSize`

Specifies the maximum size of the buffer.

## Return value

If this function succeeds, the return value is TRUE.

If this function fails, the return value is FALSE.

## Remarks

**Windows 95/98/Me:** `GetLogColorSpaceW` is supported by the Microsoft Layer for Unicode. To use this, you must add certain files to your application, as outlined in [Microsoft Layer for Unicode on Windows 95/98/Me Systems](#).

## Note

The wingdi.h header defines GetLogColorSpace as an alias that automatically selects the ANSI or Unicode version of this function based on the definition of the UNICODE preprocessor constant. Mixing usage of the encoding-neutral alias with code that is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

# Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	wingdi.h
Library	Gdi32.lib
DLL	Gdi32.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# GetLogColorSpaceW function (wingdi.h)

Article11/20/2024

The **GetLogColorSpace** function retrieves the [color space](#) definition identified by a specified handle.

## Syntax

C++

```
BOOL GetLogColorSpaceW(
    HCOLORSPACE     hColorSpace,
    LPLOGCOLORSPACEW lpBuffer,
    DWORD           nSize
);
```

## Parameters

**hColorSpace**

Specifies the handle to a color space.

**lpBuffer**

Points to a buffer to receive the [LOGCOLORSPACE](#) structure.

**nSize**

Specifies the maximum size of the buffer.

## Return value

If this function succeeds, the return value is TRUE.

If this function fails, the return value is FALSE.

## Remarks

**Windows 95/98/Me:** `GetLogColorSpaceW` is supported by the Microsoft Layer for Unicode. To use this, you must add certain files to your application, as outlined in [Microsoft Layer for Unicode on Windows 95/98/Me Systems](#).

## Note

The wingdi.h header defines GetLogColorSpace as an alias that automatically selects the ANSI or Unicode version of this function based on the definition of the UNICODE preprocessor constant. Mixing usage of the encoding-neutral alias with code that is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

# Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	wingdi.h
Library	Gdi32.lib
DLL	Gdi32.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# GetMValue macro (wingdi.h)

The **GetMValue** macro retrieves the magenta color value from a CMYK color value.

## Syntax

C++

```
BYTE GetMValue(  
    cmyk  
)
```

## Parameters

cmyk

CMYK color value from which the magenta color value will be retrieved.

## Return value

Type: **BYTE**

The magenta color value from a CMYK color value.

## Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	wingdi.h

## See also

- [Basic color management concepts](#)
- [Functions](#)

- [CMYK](#)
  - [GetCValue](#)
  - [GetKValue](#)
  - [GetYValue](#)
  - [Macros for CMYK values and colors](#)
- 

Last updated on 07/09/2025

# GetYValue macro (wingdi.h)

The `GetYValue` macro retrieves the yellow color value from a CMYK color value.

## Syntax

C++

```
BYTE GetYValue(  
    cmyk  
)
```

## Parameters

cmyk

CMYK color value from which the yellow color value will be retrieved.

## Return value

Type: `BYTE`

The yellow color value from a CMYK color value.

## Requirements

 Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	wingdi.h

## See also

- [Basic color management concepts](#)
- [Functions](#)

- [CMYK](#)
  - [GetCValue](#)
  - [GetKValue](#)
  - [Macros for CMYK values and colors](#)
- 

Last updated on 07/09/2025

# ICMENUMPROCA callback function (wingdi.h)

Article 11/20/2024

The `EnumICMProfilesProcCallback` callback is an application-defined callback function that processes color profile data from `EnumICMProfiles`.

## Syntax

C++

```
ICMENUMPROCA Icmenumproca;

int Icmenumproca(
    LPSTR unnamedParam1,
    LPARAM unnamedParam2
)
{...}
```

## Parameters

unnamedParam1

unnamedParam2

## Return value

This function must return a positive value to continue enumeration, or zero to stop enumeration. It may not return a negative value.

## Remarks

### ⓘ Note

The wingdi.h header defines `ICMENUMPROC` as an alias that automatically selects the ANSI or Unicode version of this function based on the definition of the `UNICODE` preprocessor constant. Mixing usage of the encoding-neutral alias with

code that is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

# Requirements

[+] Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	wingdi.h

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [EnumICMProfilesW](#)

---

## Feedback

Was this page helpful?



Yes



No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# ICMENUMPROCW callback function (wingdi.h)

Article 11/20/2024

The `EnumICMProfilesProcCallback` callback is an application-defined callback function that processes color profile data from `EnumICMProfiles`.

## Syntax

C++

```
ICMENUMPROCW Icmenumprocw;

int Icmenumprocw(
    LPWSTR unnamedParam1,
    LPARAM unnamedParam2
)
{...}
```

## Parameters

unnamedParam1

unnamedParam2

## Return value

This function must return a positive value to continue enumeration, or zero to stop enumeration. It may not return a negative value.

## Remarks

### ⓘ Note

The wingdi.h header defines `ICMENUMPROC` as an alias that automatically selects the ANSI or Unicode version of this function based on the definition of the `UNICODE` preprocessor constant. Mixing usage of the encoding-neutral alias with

code that is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

# Requirements

[+] Expand table

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	wingdi.h

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [EnumICMProfilesW](#)

## Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# LOGCOLORSPACEA structure (wingdi.h)

Article09/01/2022

The **LOGCOLORSPACE** structure contains information that defines a logical [color space](#).

## Syntax

C++

```
typedef struct tagLOGCOLORSPACEA {
    DWORD         lcsSignature;
    DWORD         lcsVersion;
    DWORD         lcsSize;
    LCSCSTYPE     lcsCSType;
    LCGAMUTMATCH lcsIntent;
    CIEXYZTRIPLE lcsEndpoints;
    DWORD         lcsGammaRed;
    DWORD         lcsGammaGreen;
    DWORD         lcsGammaBlue;
    CHAR          lcsFilename[MAX_PATH];
} LOGCOLORSPACEA, *LPLOGCOLORSPACEA;
```

## Members

**lcsSignature**

Color space signature. At present, this member should always be set to `LCS_SIGNATURE`.

**lcsVersion**

Version number; must be `0x400`.

**lcsSize**

Size of this structure, in bytes.

**lcsCSType**

Color space type. The member can be one of the following values.

Value	Meaning
<code>LCS_CALIBRATED_RGB</code>	Color values are calibrated RGB values. The values are translated using the endpoints specified by the <code>lcsEndpoints</code> member before being passed to the device.

LCS_sRGB	Color values are sRGB values.
LCS_WINDOWS_COLOR_SPACE	Color values are Windows default color space color values.

If LCS\_CALIBRATED\_RGB is not specified, the **lcsEndpoints** member is ignored.

#### **lcsIntent**

The gamut mapping method. This member can be one of the following values.

<b>Value</b>	<b>Intent</b>	<b>ICC Name</b>	<b>Meaning</b>
LCS_GM_ABS_	Match	Absolute Colorimetric	Maintain the white point. Match the colors to their nearest color in the destination gamut.
COLORIMETRIC			
LCS_GM_	Graphic	Saturation	Maintain saturation. Used for business charts and other situations in which undithered colors are required.
BUSINESS			
LCS_GM_	Proof	Relative Colorimetric	Maintain colorimetric match. Used for graphic designs and named colors.
GRAPHICS			
LCS_GM_	Picture	Perceptual	Maintain contrast. Used for photographs and natural images.
IMAGES			

#### **lcsEndpoints**

Red, green, blue endpoints.

#### **lcsGammaRed**

Scale of the red coordinate.

#### **lcsGammaGreen**

Scale of the green coordinate.

#### **lcsGammaBlue**

Scale of the blue coordinate.

#### **lcsFilename[MAX\_PATH]**

A null-terminated string that names a color profile file. This member is typically set to zero, but may be used to set the color space to be exactly as specified by the color profile. This is useful for devices that input color values for a specific printer, or when using an installable image color matcher. If a color profile is specified, all other members of this structure should be set to reasonable values, even if the values are not completely accurate.

## Remarks

Like palettes, but unlike pens and brushes, a pointer must be passed when creating a LogColorSpace.

If the **IcsCSType** member is set to LCS\_sRGB or LCS\_WINDOWS\_COLOR\_SPACE, the other members of this structure are ignored, and WCS uses the sRGB color space. The **IcsEndpoints**, **IcsGammaRed**, **IcsGammaGreen**, and **IcsGammaBlue** members are used to describe the logical color space. The **IcsEndpoints** member is a **CIEXYZTRIPLE** that contains the x, y, and z values of the color space's RGB endpoint.

The required DWORD bit format for the **IcsGammaRed**, **IcsGammaGreen**, and **IcsGammaBlue** is an 8.8 fixed point integer left-shifted by 8 bits. This means 8 integer bits are followed by 8 fraction bits. Taking the bit shift into account, the required format of the 32-bit DWORD is:

00000000nnnnnnnnfffffff00000000

Whenever the **IcsFilename** member contains a file name and the **IcsCSType** member is set to LCS\_CALIBRATED\_RGB, WCS ignores the other members of this structure. It uses the color space in the file as the color space to which this **LOGCOLORSPACE** structure refers.

The relation between tri-stimulus values X,Y,Z and chromaticity values x,y,z is as follows:

$$x = X/(X+Y+Z)$$

$$y = Y/(X+Y+Z)$$

$$z = Z/(X+Y+Z)$$

If the **IcsCSType** member is set to LCS\_sRGB or LCS\_WINDOWS\_COLOR\_SPACE, the other members of this structure are ignored, and ICM uses the sRGB color space. Applications should still initialize the rest of the structure since CreateProfileFromLogColorSpace ignores **IcsCSType** member and uses **IcsEndpoints**, **IcsGammaRed**, **IcsGammaGreen**, **IcsGammaBlue** members to create a profile, which may not be initialized in case of LCS\_sRGB or LCS\_WINDOWS\_COLOR\_SPACE color spaces.

## Note

The wingdi.h header defines LOGCOLORSPACE as an alias which automatically selects the ANSI or Unicode version of this function based on the definition of the UNICODE preprocessor constant. Mixing usage of the encoding-neutral alias with code that is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

## Requirements

<b>Minimum supported client</b>	Windows 2000 Professional [desktop apps only]
<b>Minimum supported server</b>	Windows 2000 Server [desktop apps only]
<b>Header</b>	wingdi.h

## See also

[BITMAPV4HEADER](#)

[BITMAPV5HEADER](#)

[CMYK](#)

[RGB](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# LOGCOLORSPACEW structure (wingdi.h)

Article09/01/2022

The **LOGCOLORSPACE** structure contains information that defines a logical [color space](#).

## Syntax

C++

```
typedef struct tagLOGCOLORSPACEW {
    DWORD         lcsSignature;
    DWORD         lcsVersion;
    DWORD         lcsSize;
    LCSCSTYPE     lcsCSType;
    LCGAMUTMATCH lcsIntent;
    CIEXYZTRIPLE lcsEndpoints;
    DWORD         lcsGammaRed;
    DWORD         lcsGammaGreen;
    DWORD         lcsGammaBlue;
    WCHAR         lcsFilename[MAX_PATH];
} LOGCOLORSPACEW, *LPLOGCOLORSPACEW;
```

## Members

**lcsSignature**

Color space signature. At present, this member should always be set to `LCS_SIGNATURE`.

**lcsVersion**

Version number; must be `0x400`.

**lcsSize**

Size of this structure, in bytes.

**lcsCSType**

Color space type. The member can be one of the following values.

Value	Meaning
<code>LCS_CALIBRATED_RGB</code>	Color values are calibrated RGB values. The values are translated using the endpoints specified by the <code>lcsEndpoints</code> member before being passed to the device.

LCS_sRGB	Color values are sRGB values.
LCS_WINDOWS_COLOR_SPACE	Color values are Windows default color space color values.

If LCS\_CALIBRATED\_RGB is not specified, the **lcsEndpoints** member is ignored.

#### **lcsIntent**

The gamut mapping method. This member can be one of the following values.

<b>Value</b>	<b>Intent</b>	<b>ICC Name</b>	<b>Meaning</b>
LCS_GM_ABS_	Match	Absolute Colorimetric	Maintain the white point. Match the colors to their nearest color in the destination gamut.
COLORIMETRIC			
LCS_GM_	Graphic	Saturation	Maintain saturation. Used for business charts and other situations in which undithered colors are required.
BUSINESS			
LCS_GM_	Proof	Relative Colorimetric	Maintain colorimetric match. Used for graphic designs and named colors.
GRAPHICS			
LCS_GM_	Picture	Perceptual	Maintain contrast. Used for photographs and natural images.
IMAGES			

#### **lcsEndpoints**

Red, green, blue endpoints.

#### **lcsGammaRed**

Scale of the red coordinate.

#### **lcsGammaGreen**

Scale of the green coordinate.

#### **lcsGammaBlue**

Scale of the blue coordinate.

#### **lcsFilename[MAX\_PATH]**

A null-terminated string that names a color profile file. This member is typically set to zero, but may be used to set the color space to be exactly as specified by the color profile. This is useful for devices that input color values for a specific printer, or when using an installable image color matcher. If a color profile is specified, all other members of this structure should be set to reasonable values, even if the values are not completely accurate.

## Remarks

Like palettes, but unlike pens and brushes, a pointer must be passed when creating a LogColorSpace.

If the **IcsCSType** member is set to LCS\_sRGB or LCS\_WINDOWS\_COLOR\_SPACE, the other members of this structure are ignored, and WCS uses the sRGB color space. The **IcsEndpoints**, **IcsGammaRed**, **IcsGammaGreen**, and **IcsGammaBlue** members are used to describe the logical color space. The **IcsEndpoints** member is a **CIEXYZTRIPLE** that contains the x, y, and z values of the color space's RGB endpoint.

The required DWORD bit format for the **IcsGammaRed**, **IcsGammaGreen**, and **IcsGammaBlue** is an 8.8 fixed point integer left-shifted by 8 bits. This means 8 integer bits are followed by 8 fraction bits. Taking the bit shift into account, the required format of the 32-bit DWORD is:

00000000nnnnnnnnfffffff00000000

Whenever the **IcsFilename** member contains a file name and the **IcsCSType** member is set to LCS\_CALIBRATED\_RGB, WCS ignores the other members of this structure. It uses the color space in the file as the color space to which this **LOGCOLORSPACE** structure refers.

The relation between tri-stimulus values X,Y,Z and chromaticity values x,y,z is as follows:

$$x = X/(X+Y+Z)$$

$$y = Y/(X+Y+Z)$$

$$z = Z/(X+Y+Z)$$

If the **IcsCSType** member is set to LCS\_sRGB or LCS\_WINDOWS\_COLOR\_SPACE, the other members of this structure are ignored, and ICM uses the sRGB color space. Applications should still initialize the rest of the structure since CreateProfileFromLogColorSpace ignores **IcsCSType** member and uses **IcsEndpoints**, **IcsGammaRed**, **IcsGammaGreen**, **IcsGammaBlue** members to create a profile, which may not be initialized in case of LCS\_sRGB or LCS\_WINDOWS\_COLOR\_SPACE color spaces.

## Note

The wingdi.h header defines LOGCOLORSPACE as an alias which automatically selects the ANSI or Unicode version of this function based on the definition of the UNICODE preprocessor constant. Mixing usage of the encoding-neutral alias with code that is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

## Requirements

<b>Minimum supported client</b>	Windows 2000 Professional [desktop apps only]
<b>Minimum supported server</b>	Windows 2000 Server [desktop apps only]
<b>Header</b>	wingdi.h

## See also

[BITMAPV4HEADER](#)

[BITMAPV5HEADER](#)

[CMYK](#)

[RGB](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# SetColorSpace function (wingdi.h)

Article02/22/2024

The `SetColorSpace` function defines the input [color space](#) for a given device context.

## Syntax

C++

```
HCOLORSPACE SetColorSpace(  
    HDC      hdc,  
    HCOLORSPACE hcs  
)
```

## Parameters

hdc

Specifies the handle to a device context.

hcs

Identifies handle to the color space to set.

## Return value

If this function succeeds, the return value is a handle to the *hColorSpace* being replaced.

If this function fails, the return value is **NULL**.

## Requirements

[ ] [Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows

Requirement	Value
Header	wingdi.h
Library	Gdi32.lib
DLL	Gdi32.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?



[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# SetDeviceGammaRamp function (wingdi.h)

Article04/02/2021

The **SetDeviceGammaRamp** function sets the [gamma ramp](#) on direct color display boards having drivers that support downloadable gamma ramps in hardware.

## Important

We strongly recommend that you don't use this API. Use of this API is subject to major limitations:

- **SetDeviceGammaRamp** implements heuristics to check whether a provided ramp will result in an unreadable screen. If a ramp violates those heuristics, then the function fails silently (that is, it returns **TRUE**, but it doesn't set your ramp). For that reason, you can't expect to use this function to set *just any arbitrary* gamma ramp. In particular, the heuristics prevent ramps that would result in nearly all pixels approaching a single value (such as fullscreen black/white) as this may prevent a user from recovering the screen.
- Because of the function's global nature, any other application on the system could, at any time, overwrite any ramp that you've set. In some cases the operating system itself may reserve the use of this function, causing any existing ramp to be overwritten. The gamma ramp is also reset on most display events (connecting/disconnecting a monitor, resolution changes, etc.). So you can't be certain that any ramp you set is in effect.
- This API has undefined behavior in HDR modes.
- This API has undefined interaction with both built-in and third-party color calibration solutions.

For color calibration, we recommend that you create an International Color Consortium (ICC) profile, and let the OS apply the profile. For advanced original equipment manufacturer (OEM) scenarios, there's a device driver model that you can use to customize color calibration more directly. See the [Windows Color System](#) for information on managing color profiles.

For blue light filtering, Windows now provides built-in support called [Night Light](#). We recommend directing users to this feature.

For color adaptation (for example, adjusting color calibration based on ambient light sensors), Windows now provides built-in support, which we recommend for use by OEMs.

For custom filter effects, there are a variety of built-in accessibility [color filters](#) to help with a range of cases.

## Syntax

C++

```
BOOL SetDeviceGammaRamp(  
    HDC     hdc,  
    LPVOID lpRamp  
) ;
```

## Parameters

hdc

Specifies the device context of the direct color display board in question.

lpRamp

Pointer to a buffer containing the gamma ramp to be set. The gamma ramp is specified in three arrays of 256 **WORD** elements each, which contain the mapping between RGB values in the frame buffer and digital-analog-converter (*DAC*) values. The sequence of the arrays is red, green, blue. The RGB values must be stored in the most significant bits of each WORD to increase DAC independence.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**.

## Remarks

Direct color display modes do not use color lookup tables and are usually 16, 24, or 32 bit. Not all direct color video boards support loadable gamma ramps. **SetDeviceGammaRamp** succeeds only for devices with drivers that support downloadable gamma ramps in hardware.

 **Note**

This API can take a non-trivial amount of time to execute. It may take as long as 200ms to return on some hardware.

## Requirements

<b>Minimum supported client</b>	Windows 2000 Professional [desktop apps only]
<b>Minimum supported server</b>	Windows 2000 Server [desktop apps only]
<b>Target Platform</b>	Windows
<b>Header</b>	wingdi.h
<b>Library</b>	Gdi32.lib
<b>DLL</b>	Gdi32.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# SetICMMode function (wingdi.h)

Article04/02/2021

The **SetICMMode** function causes Image Color Management to be enabled, disabled, or queried on a given device context (DC).

## Syntax

C++

```
int SetICMMode(
    HDC hdc,
    int mode
);
```

## Parameters

hdc

Identifies handle to the device context.

mode

Turns on and off image color management. This parameter can take one of the following constant values.

Value	Meaning
ICM_ON	Turns on color management. Turns off old-style color correction of halftones.
ICM_OFF	Turns off color management. Turns on old-style color correction of halftones.
ICM_QUERY	Queries the current state of color management.
ICM_DONE_OUTSIDEDC	Turns off color management inside DC. Under Windows 2000, also turns off old-style color correction of halftones. Not supported under Windows 95.

## Return value

If this function succeeds, the return value is a nonzero value.

If this function fails, the return value is zero.

If ICM\_QUERY is specified and the function succeeds, the nonzero value returned is ICM\_ON or ICM\_OFF to indicate the current mode.

## Remarks

If the system cannot find an ICC color profile to match the state of the device, **SetICMMODE** fails and returns zero.

Once WCS is enabled for a device context (DC), colors passed into the DC using most Win32 API functions are color matched. The primary exceptions are **BitBlt** and **StretchBlt**. The assumption is that when performing a bit block transfer (blit) from one DC to another, the two DCs are already compatible and need no color correction. If this is not the case, color correction may be performed. Specifically, if a device independent bitmap (DIB) is used as the source for a blit, and the blit is performed into a DC that has WCS enabled, color matching will be performed. If this is not what you want, turn WCS off for the destination DC by calling **SetICMMODE** before calling **BitBlt** or **StretchBlt**.

If the **CreateCompatibleDC** function is used to create a bitmap in a DC, it is possible for the bitmap to be color matched twice, once when it is created and once when a blit is performed. The reason is that a bitmap in a DC created by the **CreateCompatibleDC** function acquires the current brush, pens, and palette of the source DC. However, WCS will be disabled by default for the new DC. If WCS is later enabled for the new DC by using the **SetICMMODE** function, a color correction will be done. To prevent double color corrections through the use of the **CreateCompatibleDC** function, use the **SetICMMODE** function to turn WCS off for the source DC before the **CreateCompatibleDC** function is called.

When a compatible DC is created from a printer's DC (see **CreateCompatibleDC** ), the default is for color matching to always be performed if it is enabled for the printer's DC. The default color profile for the printer is used when a blit is performed into the printer's DC using **SetDIBitsToDevice** or **StretchDIBits**. If this is not what you want, turn WCS off for the printer's DC by calling **SetICMMODE** before calling **SetDIBitsToDevice** or **StretchDIBits**.

Also, when printing to a printer's DC with WCS turned on, the **SetICMMODE** function needs to be called after every call to the **StartPage** function to turn back on WCS. The **StartPage** function calls the **RestoreDC** and **SaveDC** functions, which result in WCS being turned off for the printer's DC.

# Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	wingdi.h
Library	Gdi32.lib
DLL	Gdi32.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [BitBlt](#)
- [CreateCompatibleDC](#)
- [SetDIBitsToDevice](#)
- [StartPage](#)
- [StretchBlt](#)
- [StretchDIBits](#)

---

## Feedback

Was this page helpful?



[Get help at Microsoft Q&A](#)

# SetICMProfileA function (wingdi.h)

Article 02/22/2024

The **SetICMProfile** function sets a specified color profile as the output profile for a specified device context (DC).

## Syntax

C++

```
BOOL SetICMProfileA(  
    HDC    hdc,  
    LPSTR lpFileName  
) ;
```

## Parameters

hdc

Specifies a device context in which to set the color profile.

lpFileName

Specifies the path name of the color profile to be set.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**.

## Remarks

**SetICMProfile** associates a color profile with a device context. It becomes the output profile for that device context. The color profile does not have to be associated with any particular device. Device-independent profiles such as sRGB can also be used. If the color profile is not associated with a hardware device, it will be returned by [GetICMProfile](#), but not by [EnumICMProfiles](#).

Note that under Windows 95 or later, the PostScript device driver for printers assumes a CMYK color model. Therefore, all PostScript printers must use a CMYK color profile. Windows 2000 does not have this limitation.

**SetICMProfile** supports only RGB profiles in compatible DCs.

**Windows 95/98/Me:** **SetICMProfileW** is supported by the Microsoft Layer for Unicode. To use this, you must add certain files to your application, as outlined in [Microsoft Layer for Unicode on Windows 95/98/Me Systems](#).

 **Note**

The wingdi.h header defines SetICMProfile as an alias that automatically selects the ANSI or Unicode version of this function based on the definition of the UNICODE preprocessor constant. Mixing usage of the encoding-neutral alias with code that is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

## Requirements

 [Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	wingdi.h
Library	Gdi32.lib
DLL	Gdi32.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [EnumICMProfilesW](#)
- [GetICMProfileW](#)

---

# Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# SetICMProfileW function (wingdi.h)

Article 02/22/2024

The **SetICMProfile** function sets a specified color profile as the output profile for a specified device context (DC).

## Syntax

C++

```
BOOL SetICMProfileW(
    HDC     hdc,
    LPWSTR lpFileName
);
```

## Parameters

hdc

Specifies a device context in which to set the color profile.

lpFileName

Specifies the path name of the color profile to be set.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**.

## Remarks

**SetICMProfile** associates a color profile with a device context. It becomes the output profile for that device context. The color profile does not have to be associated with any particular device. Device-independent profiles such as sRGB can also be used. If the color profile is not associated with a hardware device, it will be returned by [GetICMProfile](#), but not by [EnumICMProfiles](#).

Note that under Windows 95 or later, the PostScript device driver for printers assumes a CMYK color model. Therefore, all PostScript printers must use a CMYK color profile. Windows 2000 does not have this limitation.

**SetICMProfile** supports only RGB profiles in compatible DCs.

**Windows 95/98/Me:** **SetICMProfileW** is supported by the Microsoft Layer for Unicode. To use this, you must add certain files to your application, as outlined in [Microsoft Layer for Unicode on Windows 95/98/Me Systems](#).

 **Note**

The wingdi.h header defines SetICMProfile as an alias that automatically selects the ANSI or Unicode version of this function based on the definition of the UNICODE preprocessor constant. Mixing usage of the encoding-neutral alias with code that is not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

## Requirements

 [Expand table](#)

Requirement	Value
Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	wingdi.h
Library	Gdi32.lib
DLL	Gdi32.dll

## See also

- [Basic color management concepts](#)
- [Functions](#)
- [EnumICMProfilesW](#)
- [GetICMProfileW](#)

---

# Feedback

Was this page helpful?

 Yes

 No

[Provide product feedback ↗](#) | [Get help at Microsoft Q&A](#)

# UpdateICMRegKeyA function (wingdi.h)

Article 02/09/2023

*(Obsolete; retained for backward compatibility)*

The **UpdateICMRegKey** function manages color profiles and Color Management Modules in the system.

## Syntax

C++

```
BOOL UpdateICMRegKeyA(
    DWORD reserved,
    LPSTR lpszCMID,
    LPSTR lpszFileName,
    UINT command
);
```

## Parameters

`reserved`

Reserved, must be set to zero.

`lpszCMID`

Points to a string that specifies the ICC profile identifier for the color management DLL to use with the profile.

`lpszFileName`

Points to a fully qualified ICC color profile file name or to a **DEVMODE** structure.

`command`

Specifies a function to execute. It can have one of the following values.

Value	Meaning
<code>ICM_ADDPROFILE</code>	Installs the ICC profile in the system.
<code>ICM_DELETEPROFILE</code>	Uninstalls the ICC profile from the system, but does not

	delete the file.
<b>ICM_QUERYPROFILE</b>	Determines whether the profile is already installed in the system.
<b>ICM_SETDEFAULTPROFILE</b>	Makes the profile first among equals.
<b>ICM_REGISTERICMATCHER</b>	Registers a CMM in the system. The <i>pszFileName</i> parameter points to a fully qualified path for the CMM DLL. The <i>lpszCMID</i> parameter points to a <b>DWORD</b> identifying the CMM.
<b>ICM_UNREGISTERICMATCHER</b>	Unregisters the CMM from the system. The <i>lpszCMID</i> parameter points to a <b>DWORD</b> identifying the CMM.
<b>ICM_QUERYMATCH</b>	Determines whether a profile exists based on the <b>DEVMODE</b> structure pointed to by the <i>pszFileName</i> parameter.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**.

## Remarks

Not all parameters are used by all functions. The *nCommand* parameter specifies the function to execute.

This function is retained for backward compatibility and may be removed in future versions of ICM.

**Windows 95/98/Me:** **UpdateICMRegKeyW** is supported by the Microsoft Layer for Unicode. To use this, you must add certain files to your application, as outlined in [Microsoft Layer for Unicode on Windows 95/98/Me Systems](#) ↗.

### ⓘ Note

The wingdi.h header defines **UpdateICMRegKey** as an alias which automatically selects the ANSI or Unicode version of this function based on the definition of the **UNICODE** preprocessor constant. Mixing usage of the encoding-neutral alias with code that not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

# Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	wingdi.h
Library	Gdi32.lib
DLL	Gdi32.dll

## See also

- [Basic color management concepts](#)
- [Obsolete WCS functions](#)

## Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)

# UpdateICMRegKeyW function (wingdi.h)

Article 02/09/2023

*(Obsolete; retained for backward compatibility)*

The **UpdateICMRegKey** function manages color profiles and Color Management Modules in the system.

## Syntax

C++

```
BOOL UpdateICMRegKeyW(
    DWORD reserved,
    LPWSTR lpszCMID,
    LPWSTR lpszFileName,
    UINT command
);
```

## Parameters

`reserved`

Reserved, must be set to zero.

`lpszCMID`

Points to a string that specifies the ICC profile identifier for the color management DLL to use with the profile.

`lpszFileName`

Points to a fully qualified ICC color profile file name or to a **DEVMODE** structure.

`command`

Specifies a function to execute. It can have one of the following values.

Value	Meaning
<code>ICM_ADDPROFILE</code>	Installs the ICC profile in the system.
<code>ICM_DELETEPROFILE</code>	Uninstalls the ICC profile from the system, but does not

	delete the file.
<b>ICM_QUERYPROFILE</b>	Determines whether the profile is already installed in the system.
<b>ICM_SETDEFAULTPROFILE</b>	Makes the profile first among equals.
<b>ICM_REGISTERICMATCHER</b>	Registers a CMM in the system. The <i>pszFileName</i> parameter points to a fully qualified path for the CMM DLL. The <i>lpszCMID</i> parameter points to a <b>DWORD</b> identifying the CMM.
<b>ICM_UNREGISTERICMATCHER</b>	Unregisters the CMM from the system. The <i>lpszCMID</i> parameter points to a <b>DWORD</b> identifying the CMM.
<b>ICM_QUERYMATCH</b>	Determines whether a profile exists based on the <b>DEVMODE</b> structure pointed to by the <i>pszFileName</i> parameter.

## Return value

If this function succeeds, the return value is **TRUE**.

If this function fails, the return value is **FALSE**.

## Remarks

Not all parameters are used by all functions. The *nCommand* parameter specifies the function to execute.

This function is retained for backward compatibility and may be removed in future versions of ICM.

**Windows 95/98/Me:** **UpdateICMRegKeyW** is supported by the Microsoft Layer for Unicode. To use this, you must add certain files to your application, as outlined in [Microsoft Layer for Unicode on Windows 95/98/Me Systems](#) ↗.

### ⓘ Note

The wingdi.h header defines **UpdateICMRegKey** as an alias which automatically selects the ANSI or Unicode version of this function based on the definition of the **UNICODE** preprocessor constant. Mixing usage of the encoding-neutral alias with code that not encoding-neutral can lead to mismatches that result in compilation or runtime errors. For more information, see [Conventions for Function Prototypes](#).

# Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	wingdi.h
Library	Gdi32.lib
DLL	Gdi32.dll

## See also

- [Basic color management concepts](#)
- [Obsolete WCS functions](#)

---

## Feedback

Was this page helpful?

 Yes

 No

[Get help at Microsoft Q&A](#)