

Technote 1160

What's New With ColorSync 2.6

CONTENTS

[System Requirements](#)

[Checking for ColorSync 2.6](#)

[ColorSync 2.6 Version Information](#)

[New ColorSync 2.6 APIs](#)

[ColorSync 2.6 Compatibility Issues](#)

[Further References](#)

[Downloadables](#)

ColorSync 2.6 is the latest version of Apple Computer's color management architecture for Mac OS. This Technote describes in detail the changes in this new version of ColorSync. This note is primarily intended for developers who are using the ColorSync APIs.

System Requirements

ColorSync 2.6 for Macintosh requires a PowerPC machine running Mac OS 8.1 or later.

[Back to top](#)

Checking for ColorSync 2.6

Macintosh

To determine if the ColorSync Manager shared libraries have been loaded on a PowerPC machine, use the `Gestalt` function with the `gestaltColorMatchingAttr` selector. Test the bit field (bit 1) indicated by the `gestaltColorMatchingLibLoaded` constant in the response parameter. If the bit is set, the ColorSync Manager shared libraries are loaded. The following code snippet shows how this is done. This code snippet initializes the `ColorSyncAvailable` Boolean variable to `false`.

```
Boolean CheckIfColorSyncAvailableOnPPC (void)
{
    Boolean ColorSyncAvailable = false;
    long gestaltResponse;

    if (Gestalt(gestaltColorMatchingAttr, &gestaltResponse) == noErr)
    {
        ColorSyncAvailable =
            gestaltResponse & (1 << gestaltColorMatchingLibLoaded);
    }

    return ColorSyncAvailable;
}
```

Alternately, you can use the new ColorSync 2.6 function `CMGetColorSyncVersion` (see the section "[New ColorSync 2.6 APIs](#)" below for the details) to check for the presence of ColorSync.

The `CMGetColorSyncVersion` function also returns the ColorSync version information. Here's a code snippet that checks for ColorSync version 2.:

```
#define kColorSync26 0x00000260

CMError err;
UInt32 version;

err = CMGetColorSyncVersion(&version);
if (err == noErr)
{
    if (version >= kColorSync26)
    {
        /* ColorSync 2.6 or better is installed */
    }
}
else
{
    /* ColorSync not present */
}
```

[Back to top](#)

Getting ColorSync 2.6 Version Information

As described in the section "[Checking for ColorSync 2.6](#)", use the new ColorSync 2.6 function `CMGetColorSyncVersion` to get ColorSync version information.

Important Note:

On Macintosh systems with ColorSync 2.6 installed, the `CMGetColorSyncVersion` function returns the value 0x00026000 in the version field. However, to remain consistent with `Gestalt` return values this function should return the value 0x00000260. Future versions of ColorSync 2.6 on the Macintosh will correct this problem and return values which are consistent with `Gestalt`.

Alternately, you can use the `Gestalt` function with the `gestaltColorMatchingVersion` selector to get ColorSync version information.

You can modify and use the following sample code to test for version 2.6 of the ColorSync Manager. This function initializes the Boolean variable `ColorSyncAvailable` to `false` and sets it to `true` if version 2.6 or later of the ColorSync Manager is installed.

```
#define kColorSync26 0x00000260

Boolean CheckForColorSyncVersion26(void)
{
    Boolean ColorSyncAvailable = false;
    long version;

    if (Gestalt(gestaltColorMatchingVersion, &version) == noErr)
    {
        if (version >= kColorSync26)
        {
            ColorSyncAvailable = true;
        }
    }

    return ColorSyncAvailable;
}
```

[Back to top](#)

New ColorSync 2.6 APIs

The following new APIs were added to ColorSync 2.6:

```
CMError CMGetProfileDescriptions (CMPProfileRef prof,
    char *aName,
    UInt32 *aCount,
    Str255 mName,
    ScriptCode *mCode,
    UniChar *uName,
    UniCharCount *uCount);
```

Field Descriptions

prof	A reference to the profile from which to obtain the desc tag info.
aName	A pointer to a field which is to receive the profile name as a 7-bit Roman ASCII string.
aCount	A pointer to a field which is to receive a count of the number of characters returned in the aName field above.
mName	A pointer to a field which is to receive the localized profile name string in Mac script-code format.
mCode	A pointer to a field which is to receive the script code corresponding to the name string returned in the mName parameter above.
uName	A pointer to a field which is to receive the Unicode localized profile name string.
uCount	A pointer to a field which is to receive a count of the number of Unicode (2-byte) characters returned in the uName field above.

```
CMError CMSetProfileDescriptions (CMProfileRef prof,
    const char *aName,
    UInt32 aCount,
    ConstStr255Param mName,
    ScriptCode mCode,
    const UniChar *uName,
    UniCharCount uCount);
```

Field Descriptions

prof	A reference to the profile into which to set the desc tag info.
aName	A pointer to a field containing a 7-bit Roman ASCII profile name string which is to be set for the profile. This string must be null-terminated.
aCount	A count of the number of characters in the aName field above.
mName	A pointer to a field containing the localized profile name string in Mac script-code format which is to be set for the profile. This string must be null-terminated.
mCode	The script code corresponding to the name string in the mName parameter above.
uName	A pointer to a field containing the Unicode localized profile name string which is to be set for the profile. This string must be null-terminated.
uCount	A count of the number of Unicode characters in the uName field above (do not confuse this with a byte count, because each Unicode character requires two bytes).

DESCRIPTION

Use these functions to get/set the description tag data for a given profile. The ICC Profile Format Specification (available at <<http://www.color.org>>) includes a Description tag ('desc'), designed to provide more information about a profile than can be contained in a file name. This is especially critical on file systems with 8.3 names. The tag data can consist of up to three separate pieces (strings) of information for a profile. These different strings are designed to allow for display in different languages or on different computer systems. Applications typically use one of the strings to show profiles in a list or a pop-up menu. ColorSync 2.6 includes these new APIs for accessing this information, and also checks for the validity of the 'desc' tag according to the ICC Spec.

```
CMError NCWConcatColorWorld (CMWorldRef *cw,  
                               NCMConcatProfileSet *profileSet,  
                               CMConcatCallbackProc proc,  
                               void *refCon);
```

Field Descriptions

cw	A reference to a color world that the ColorSync Manager returns if the function completes successfully. You pass this reference to other functions that use the color world for color-matching and color-checking sessions.
profileSet	An array of profiles describing the processing to be carried out. The array is in processing order source through destination.
proc	A calling-program-supplied callback function that allows your application to monitor progress or abort the operation.
refCon	A reference constant containing data specified by the calling application program.

New structures for use with this function:

```

struct NCMConcatProfileSet {
    OSType          cmm;                /* e.g., 'KCMS'. Uniquely IDs the CMM, or 0000 */
    unsigned long   flags;              /* specify quality */
    unsigned long   flagsMask;          /* which bits of flags to use to override profile.
*/
    unsigned long   profileCount;        /* how many ProfileSpecs in the following set */
    NCWConcatProfileSpec  profileSpecs[]; /* A new structure, defined below */
};

struct NCWConcatProfileSpec {
    unsigned long   renderingIntent;    /* intent to use along with transformTag. */
    unsigned long   transformTag;       /* one of a set of tag identifiers, defined below
*/
    CMProfileRef    profile;            /* the profile to extract the transform from */
};

enum {
    kNoTransform      = 0,              /* Not used */
    kUseAtoB           = 1,              /* Use 'A2B*' tag from this profile or equivalent */
    kUseBtoA           = 2,              /* Use 'B2A*' tag from this profile or equivalent */
    kUseBtoB           = 3,              /* Use 'pre*' tag from this profile or equivalent */

    /* For typical device profiles the following synonyms may be useful */

    kDeviceToPCS       = kUseAtoB,      /* Device Dependent to Device Independent */
    kPCSToDevice       = kUseBtoA,      /* Device Independent to Device Dependent */
    kPCSToPCS          = kUseBtoB,      /* Independent, through device's gamut */

    /* This is provided for default behavior when specifying rendering intent in the
    /* NCMConcatProfileSpec
    */

    kUseProfileIntent  = (long) 0xFFFFFFFF
                                /* For renderingIntent in NCMConcatProfileSpec*/
};

/*
    Caller-supplied progress function for NCMConcatInit & NCMNewLinkProfile routines
*/

typedef STACK_UPP_TYPE(CMConcatCallbackProcPtr) CMConcatCallbackUPP;
typedef CALLBACK_API( Boolean, CMConcatCallbackProcPtr )
    (long progress, void *refCon);

/*
    Callback Proc Creation example:
*/

CMConcatCallbackUPP myCallbackProc = NewCMConcatCallbackProc(myCallbackProc);

```

DESCRIPTION

The `NCWConcatColorWorld` function defines a color world for color transformations among a series of concatenated profiles. The caller can override the CMM that would normally be selected by `ColorSync` by providing a CMM identifier in the `NCMConcatProfileSet` structure, or pass 0 to accept `ColorSync`'s CMM selection (note that this could either be the user's preferred CMM selection or the CMM called for in the profile). The `flags` and `flagsMask` parameters are provided to allow easy customization of such attributes as quality and gamut-checking, while preserving the other settings. Each profile in the set can be customized by overriding the intent, and the selection of the transform tag. Together with other profiles, a custom-rendering environment can be set up to transform to or from device-dependent spaces with a minimum of gamut compression and/or unnecessary transformations to and from connection spaces. This flexibility comes at the price of specific knowledge of the profile contents and how device gamuts overlap.

Note that for standard input and output device profiles, `A2B` and `B2A` tags represent transforms from data space to connection space and from connection space to data space, respectively. Under these circumstances, the caller would not normally be able to use the same transform tags (e.g., `kUseAtoB`) consecutively, since a connection space would not be the same as the subsequent data space. If the spaces aren't the same, the caller will get a `cmCantConcatenateError` error (-178) returned. For profiles of type `cmLinkClass`, `cmAbstractClass`, `cmColorSpaceClass`, and `cmNamedColorClass`, these constants are not always meaningful, and the caller is encouraged to think in terms of the actual tags present in the profiles (e.g., `A2B0` or `B2A0`). Under these conditions, it may well be appropriate to specify two transform tags of the same type consecutively, as long as the actual color spaces align in between tags. If this is not the case, a `cmCantConcatenateError` error (-178) is returned.

The `CallbackProc` is provided as protection against the appearance of a stalled machine during lengthy color world processing. If a CMM takes more than several seconds to process the information and create a color world, it will call the `CallbackProc`, if one is provided, and pass it the `refCon` provided. This is also true for `NCWNewLinkProfile`, described below:

```
CLError NCWNewLinkProfile (CMProfileRef          *prof,
                           const CMProfileLocation *targetLocation,
                           NCMConcatProfileSet    *profileSet,
                           CMConcatCallbackUPP    proc,
                           void                   *refCon);
```

Field Descriptions

prof	The returned profile reference.
targetLocation	The location of the profile, which you specify using the <code>CMProfileLocation</code> data type. Commonly a profile is disk-file based. However, the profile may be a file-based profile, a handle-based profile, or a pointer-based profile.
profileSet	The profile set structure.
proc	A calling-program-supplied callback function that allows your application to monitor progress or abort the operation.
refCon	A reference constant containing data specified by the calling application program.

DESCRIPTION

The same new flexibility in creating color worlds is extended to link profiles, which are now not assumed to go from input device color space to output device color space. The returned profile is open, and should be closed when you are finished with it.

CLError	NCMSetSystemProfile (const CMProfileLocation *profLoc);
---------	---

Field Descriptions

profLoc	The location of the profile, which you specify using the <code>CMProfileLocation</code> data type. Commonly a profile is disk-file based. However, the profile may be a file-based profile, a handle-based profile, or a pointer-based profile.
---------	---

DESCRIPTION

Prior to ColorSync 2.6, the API for setting the System Profile supported only the FSSpec file specification type as a way of specifying a profile. This new API allows for greater flexibility when specifying a system profile.

```
CMError NCMUnflattenProfile (CMProfileLocation *targetLocation,
                             CMFlattenUPP      proc,
                             void               *refCon,
                             Boolean             *preferredCMMnotfound);
```

Field Descriptions

targetLocation	The location of the profile you wish to unflatten, which you specify using the <code>CMProfileLocation</code> data type. Commonly a profile is disk-file based. However, the profile may be a file-based profile, a handle-based profile, or a pointer-based profile.
proc	A user-defined procedure which is called during the unflatten operation.
refCon	A reference constant containing data specified by the calling application program.
preferredCMMnotfound	A return value indicating whether or not the CMM specified in the profile was found.

DESCRIPTION

Use this function to unflatten a profile.

```
CMError CMIterateCMMInfo (CMIterateUPP      proc,
                          UInt32             *count,
                          void               *refCon);
```

Field Descriptions

proc	A calling-program-supplied callback function that allows your application to monitor progress or abort the operation.
count	A count of the number of CMMs referenced is returned here.
refCon	A reference constant containing data specified by the calling application program.

NEW STRUCTURES FOR USE WITH THIS FUNCTION

```
OSErr CMIterateUPP(
```

```

        CMMInfo*   iterateData,      /* Ptr to a structure containing
                                     /* information about a particular CMM */
        void*      refcon           /* Caller-defined data, passed through from
                                     /* CMIterateCMMInfo API */
    );

struct CMMInfo {
    unsigned long  dataSize;         /* Size of this structure - for compatibility */
    OSType         CMMType;          /* Signature of CMM */
    OSType         CMMMfr;           /* Vendor, e.g. 'appl */
    unsigned long  CMMVersion;       /* Version number */
    Handle         CMMcons;          /* If available, can be a group with varying size
    & depth */
    unsigned char  ASCIIName[32];    /* Pascal string - name */
    unsigned char  ASCIIDesc[256];   /* Pascal string - description or copyright */
    UniCharCount   UniCodeNameCount; /* Count of UniChars in following array */
    UniChar        UniCodeName[32];  /* The name in UniCode chars */
    UniCharCount   UniCodeDescCount; /* Count of UniChars in following array */
    UniChar        UniCodeDesc[256]; /* The description in UniCode chars */
};

```

DESCRIPTION

The `CMIterateCMMInfo` function returns information for all CMMs installed on the system. The caller can pass nil for the `CMIterateUPP` param to simply get a count of CMMs. If a `CMIterateUPP` proc is provided, it will be called once for each CMM installed - with the `CMMInfo` structure filled accordingly. The caller can pass a data reference to `CMIterateCMMInfo` which will then be passed to the `CMIterateUPP`. This might be used to allow some of the information in the `CMMInfo` data structure to be put into a menu, for example, by passing a menu reference as the `refcon`. Either the proc or the count parameter must be provided. The caller will get a `paramErr` if both are nil.

CMError	CMGetColorSyncVersion (UInt32 *version);
---------	--

Field Descriptions

version	The version of ColorSync installed on the machine is returned here
---------	--

DESCRIPTION

Returns ColorSync version information. `CMGetColorSyncVersion` relieves the Mac OS developer from having to call `Gestalt` to find out the version of ColorSync installed on the system.

Important Note:

The return value for ColorSync 2.6 is 0x00026000 on the Macintosh. This is not consistent with what is currently returned by `Gestalt` (0x00000260). Future versions of ColorSync on the Macintosh will be fixed so they are consistent with `Gestalt` return values.

[Back to top](#)

Deprecated APIs

Support for the following APIs will be eliminated or limited in the future:

API	ColorSyncVersion	Reason for Deprecation
(CM)BeginMatching	1.x	Pict-based, uses 1.x Profiles
(CM)UseProfile(Comment)	1.x	Pict-based, uses 1.x Profiles
NCMUseProfileComment	2.x	Pict-based, uses 2.x Profiles
(CM)DrawMatchedPicture	1.x	Pict-based, uses 1.x Profiles
NCMDrawMatchedPicture	2.x	Pict-based, uses 2.x Profiles
(CM)GetProfileName	1.x	Uses 1.x Profiles
(CM)GetProfileAdditionalDataOffset	1.x	Uses 1.x Profiles
(CM)GetProfile	1.x	Profile Responder function
(CM)SetProfile	1.x	Profile Responder function
(CM)SetProfileDescription	1.x	Profile Responder function
(CM)GetIndexedProfile	1.x	Profile Responder function
(CM>DeleteDeviceProfile	1.x	Profile Responder function
ConcatenateProfiles	1.x	Uses 1.x Profiles
CWNewColorWorld	1.x	Uses 1.x Profiles
CMAccelerationLoadTables	2.x	CMM Function - HW acceleration no longer supported
CMAccelerationCalculateData	2.x	CMM Function - HW acceleration no longer supported

[Back to top](#)

New CMM APIs

The following APIs specify new CMM entry points which ColorSync will call to handle the creation of new color worlds or link profiles as returned by the NCWConcatColorWorld and NCWNewLinkProfile functions respectively. These APIs are of interest to CMM developers only.

```
CMError NCMMConcatInit (CMStorageHdl hStorage,
                        NCMMConcatProfileSet *profileSet,
                        CMConcatCallbackProc proc,
                        void *refCon);
```

Field Descriptions

hStorage	The CMM's private instance storage.
profileSet	Set of profiles to concatenate into a color world.
proc	Callback procedure.
refCon	User data passed on to callback procedure.

DESCRIPTION

This is the API which ColorSync calls when an application calls the `NCWConcatCol orWorl d` function. The new selector for this function is `kNCMMConcatIni t`. The CMM is responsible for validating the profiles specified in the profile set and for building a transform using these profiles. If the CMM cannot build such a transform, it should return a `cmCantConcatenateErr` error, or another appropriate error code.

```
CLError NCMMNewLinkProfile (CMStorageHdl      hStorage,
                             CMProfileRef      prof,
                             NCMConcatProfileSet *profileSet,
                             CMConcatCallBackUPP proc,
                             void              *refCon);
```

Field Descriptions

hStorage	The CMM's private instance storage.
prof	The CMM will add the appropriate tags to this profile.
profileSet	Set of profiles to concatenate into a link profile.
proc	Callback procedure.
refCon	User data passed on to callback procedure.

DESCRIPTION

This is the API which ColorSync calls when an application calls the `NCWNewLi nkProfi l e` function. The new selector for this function is `kNCMMNewLi nkProfi l e`. The CMM is responsible for validating the profiles specified in the profile set and for building a link profile using these profiles. If the CMM cannot build a link profile, it should return a `cmCantConcatenateErr` error, or another appropriate error code. ColorSync will create the profile (via `CMNewProfi l e`) before calling the CMM, so the CMM need only add the appropriate tags to the profile before returning. ColorSync will also ensure that the tags are written to the profile by calling `CMUpdateProfi l e`, before returning it to the caller.

[Back to top](#)

New ColorSync Scripting Library APIs

ColorSync 2.6 provides a new set of scripting library APIs which are designed to give applications access to the same file format utilities provided by the new ColorSync 2.6 AppleEvent terminology. In fact, the code in ColorSync which responds to AppleEvents actually calls this same library. Listed below are the functions introduced in this new scripting library:

```
CLError CMVal i dImage (const FSSpec *spec);
```

Field Descriptions

spec	A file specification for the the image file you wish to validate.
------	---

DESCRIPTION

This function validates the specified image file. ColorSync will check with any installed scripting plug-ins

to see if they recognize the image's file format. If a scripting plug-in is found which recognizes the image's file format, CMVal i dateImage will return noErr. If the image's file format is not recognized, CMVal i dateImage will return the cmI nval i dI mageFi l e error.

```
CMError CMGetImageSpace (const FSSpec *spec,
                        OSType *space);
```

Field Descriptions

spec	A file specification for the image file.
space	The signature of the data color space of the color values of colors for the image file is returned here.

DESCRIPTION

This function returns the signature of the data color space in which the color values of colors in an image are expressed.

```
CMError CMEmbedImage (const FSSpec *specFrom,
                     const FSSpec *specInto,
                     Boolean      repl,
                     CMProfileRef  embProf);
```

Field Descriptions

specFrom	A file specification for the image file.
specInto	If this parameter is a file, it specifies the resulting image. If this parameter is a folder, it specifies the location of the resulting image which will have the same name as the original file. If this parameter is not provided, the original file is modified.
repl	If a file with the same name already exists, it will be replaced if this parameter is set to true.
embProf	The profile to embed in the image.

DESCRIPTION

This function will embed an image with an ICC profile.

```
CMError CMUnembedImage (const FSSpec *specFrom,
                       const FSSpec *specInto,
                       Boolean      repl);
```

Field Descriptions

specFrom	A file specification for the image file.
specInto	If this parameter is a file, it specifies the resulting image. If this parameter is a folder, it specifies the location of the resulting image which will have the same name as the original file. If this parameter is not provided, the original file is modified.

repl	If a file with the same name already exists, it will be replaced if this parameter is set to true.
------	--

DESCRIPTION

This function will remove any ICC profiles embeded in an image.

```
CMError CMMatchImage (const FSSpec *specFrom,
                      const FSSpec *specInto,
                      Boolean      repl,
                      UInt32       qual,
                      CMProfileRef srcProf,
                      UInt32       srcIntent,
                      CMProfileRef dstProf);
```

Field Descriptions

specFrom	A file specification for the image file.
specInto	If this parameter is a file, it specifies the resulting image. If this parameter is a folder, it specifies the location of the resulting image which will have the same name as the original file. If this parameter is not provided, the original file is modified.
repl	If a file with the same name already exists, it will be replaced if this parameter is set to true.
qual	The optional quality for the match - normal, draft or best (<code>cmNormal Mode</code> , <code>cmDraftMode</code> or <code>cmBestMode</code>).
srcProf	The optional source profile for the match.
srcIntent	The rendering intent for the match - perceptual intent, relative colorimetric intent, saturation intent, or absolute colorimetric intent (<code>cmPerceptual</code> , <code>cmRelativecolorimetric</code> , <code>cmSaturation</code> or <code>cmAbsoluteColorimetric</code>).
dstProf	The destination profile for the match.

DESCRIPTION

Use this function to color match an image file.

```
CMError CMProofImage (const FSSpec *specFrom,
                      const FSSpec *specInto,
                      Boolean      repl,
                      UInt32       qual,
                      CMProfileRef srcProf,
                      UInt32       srcIntent,
                      CMProfileRef dstProf,
                      CMProfileRef prfProf);
```

Field Descriptions

specFrom	A file specification for the image file.
specInto	If this parameter is a file, it specifies the resulting image. If this parameter is a folder, it specifies the location of the resulting image which will have the same name as the original

	file. If this parameter is not provided, the original file is modified.
repl	If a file with the same name already exists, it will be replaced if this parameter is set to true.
qual	The optional quality for the match - normal, draft or best (<code>cmNormal Mode</code> , <code>cmDraftMode</code> or <code>cmBestMode</code>).
srcProf	The optional source profile for the match.
srcIntent	The rendering intent for the match between the source and destination profiles - perceptual intent, relative colorimetric intent, saturation intent, or absolute colorimetric intent (<code>cmPerceptual</code> , <code>cmRelativecolorimetric</code> , <code>cmSaturation</code> or <code>cmAbsoluteColorimetric</code>).
dstProf	The destination profile for the match.
prfProf	The proof profile for the match between the destination and proof profiles.

DESCRIPTION

Use this function to proof an image file.

```
CMError CMLinkImage (const FSSpec *specFrom,
                    const FSSpec *specInto,
                    Boolean      repl,
                    UInt32      qual,
                    CMProfileRef lnkProf,
                    UInt32      lnkIntent);
```

Field Descriptions

specFrom	A file specification for the image file.
specInto	If this parameter is a file, it specifies the resulting image. If this parameter is a folder, it specifies the location of the resulting image which will have the same name as the original file. If this parameter is not provided, the original file is modified.
repl	If a file with the same name already exists, it will be replaced if this parameter is set to true.
qual	The optional quality for the match - normal, draft or best (<code>cmNormal Mode</code> , <code>cmDraftMode</code> or <code>cmBestMode</code>).
lnkProf	The device link profile for the match.
lnkIntent	The rendering intent for the match - perceptual intent, relative colorimetric intent, saturation intent or absolute colorimetric intent (<code>cmPerceptual</code> , <code>cmRelativecolorimetric</code> , <code>cmSaturation</code> or <code>cmAbsoluteColorimetric</code>).

DESCRIPTION

Use this function to match an image file with a device link profile.

```
CMError CMCountImageProfiles (const FSSpec *spec,
                             UInt32      *count);
```

Field Descriptions

spec	A file specification for the image file.
count	A count of the embedded profiles for the image is returned here.

DESCRIPTION

Use this function to obtain a count of the number of embedded profiles for a given image.

```
CMError CMGetIndImageProfile (const FSSpec *spec,
                             UInt32      index,
                             CMProfileRef *prof);
```

Field Descriptions

spec	A file specification for the image file.
index	The numeric index of the profile to return.
prof	The profile is returned here.

DESCRIPTION

Use this function to obtain a specific embedded profile for a given image.

```
CMError CMSetIndImageProfile (const FSSpec *specFrom,
                              const FSSpec *specInto,
                              Boolean      repl,
                              UInt32      index,
                              CMProfileRef prof);
```

Field Descriptions

specFrom	A file specification for the image file.
specInto	If this parameter is a file, it specifies the resulting image. If this parameter is a folder, it specifies the location of the resulting image which will have the same name as the original file. If this parameter is not provided, the original file is modified.
repl	If a file with the same name already exists, it will be replaced if this parameter is set to true.
index	The numeric index of the profile to set.
prof	The profile to set at the index designated by the index parameter.

DESCRIPTION

Use this function to set a specific embedded profile for a given image.

New CMBitmap Types

Several new CMBi tmap spaces were added in ColorSync 2.6 (see below) to provide developers with a

wide range of data formats appropriate for multiple platforms.

Several bitmap spaces now can have a new space attribute flag `cmLittleEndianPacking` set to indicate Little-Endian data. Another new variant `cmReverseChannelPacking`, which can apply to just about all of the new and old spaces is a "reverse channels" attribute.

Bitmap spaces supported in ColorSync 2.5

- `cmGray16Space` = `cmGraySpace`,
- `cmGrayA32Space` = `cmGrayASpace`,
- `cmRGB16Space` = `cmWord5ColorPacking` + `cmRGBSpace`,
- `cmRGB24Space` = `cm24_8ColorPacking` + `cmRGBSpace`,
- `cmRGB32Space` = `cm32_8ColorPacking` + `cmRGBSpace`,
- `cmRGB48Space` = `cm48_16ColorPacking` + `cmRGBSpace`,
- `cmARGB32Space` = `cm32_8ColorPacking` + `cmAlphaFirstPacking` + `cmRGBASpace`,
- `cmRGBA32Space` = `cm32_8ColorPacking` + `cmAlphaLastPacking` + `cmRGBASpace`,
- `cmCMYK32Space` = `cm32_8ColorPacking` + `cmCMYKSpace`,
- `cmCMYK64Space` = `cm64_16ColorPacking` + `cmCMYKSpace`,
- `cmHSV32Space` = `cmLong10ColorPacking` + `cmHSVSpace`,
- `cmHLS32Space` = `cmLong10ColorPacking` + `cmHLSpace`,
- `cmYXY32Space` = `cmLong10ColorPacking` + `cmYXYSpace`,
- `cmXYZ32Space` = `cmLong10ColorPacking` + `cmXYZSpace`,
- `cmLUV32Space` = `cmLong10ColorPacking` + `cmLUVSpace`,
- `cmLAB24Space` = `cm24_8ColorPacking` + `cmLABSpace`,
- `cmLAB32Space` = `cmLong10ColorPacking` + `cmLABSpace`,
- `cmLAB48Space` = `cm48_16ColorPacking` + `cmLABSpace`,
- `cmGamutResult1Space` = `cmOneBitDirectPacking` + `cmGamutResultSpace`,
- `cmNamedIndexed32Space` = `cm32_32ColorPacking` + `cmNamedIndexedSpace`,
- `cmMCFive8Space` = `cm40_8ColorPacking` + `cmMCFiveSpace`,
- `cmMCSix8Space` = `cm48_8ColorPacking` + `cmMCSixSpace`,
- `cmMCSeven8Space` = `cm56_8ColorPacking` + `cmMCSevenSpace`,
- `cmMCEight8Space` = `cm64_8ColorPacking` + `cmMCEightSpace`

New Spaces added for 2.6

- `cmGray8Space` = `cmGraySpace` + `cm8_8ColorPacking`, 8-bit gray
- `cmGrayA16Space` = `cmGrayASpace` + `cm16_8ColorPacking`, 8-bit gray + 8-bit alpha
- `cmGray16LSpace` = `cmGraySpace` + `cmLittleEndianPacking`, 16-bit gray, little-endian
- `cmGrayA32LSpace` = `cmGrayASpace` + `cmLittleEndianPacking`, 16-bit gray + 16-bit alpha, little-endian
- `cmRGB565Space` = `cmRGBSpace` + `cmWord565ColorPacking`, Variant of 1.5.5.5
- `cmRGB565LSpace` = `cmRGBSpace` + `cmWord565ColorPacking` + `cmLittleEndianPacking`, Variant of 1.5.5.5, little-endian
- `cmRGB16LSpace` = `cmRGBSpace` + `cmWord5ColorPacking` + `cmLittleEndianPacking`, 1.5.5.5, little-endian
- `cmRGB48LSpace` = `cmRGBSpace` + `cm48_16ColorPacking` + `cmLittleEndianPacking`, 16-bits per channel RGB, little-endian
- `cmARGB64Space` = `cmRGBASpace` + `cm64_16ColorPacking` + `cmAlphaFirstPacking`, 16-bits per channel RGB w/leading alpha channel
- `cmARGB64LSpace` = `cmRGBASpace` + `cm64_16ColorPacking` + `cmAlphaFirstPacking` + `cmLittleEndianPacking`, bits / chl RGB w/leading alpha, little-endian
- `cmRGBA64Space` = `cmRGBASpace` + `cm64_16ColorPacking` + `cmAlphaLastPacking`, 16-bits / chl RGB with trailing alpha channel
- `cmRGBA64LSpace` = `cmRGBASpace` + `cm64_16ColorPacking` + `cmAlphaLastPacking` + `cmLittleEndianPacking` bits / chl RGB w/trailing alpha, little-endian
- `cmCMYK64LSpace` = `cmCMYKSpace` + `cm64_16ColorPacking` + `cmLittleEndianPacking`, 16-bits per channel CMYK, little-endian
- `cmXYZ24Space` = `cmXYZSpace` + `cm24_8ColorPacking`, 8-bit per channel XYZ
- `cmXYZ48Space` = `cmXYZSpace` + `cm48_16ColorPacking`, 16-bits per channel XYZ

- `cmXYZ48LSpace = cmXYZSpace + cm48_16ColorPacking + cmLittleEndianPacking`, 16-bits
- per channel XYZ, little-endian
- `cmLAB48LSpace = cmLABSpace + cm48_16ColorPacking + cmLittleEndianPacking`, 16-bits
- per channel Lab, little-endian
- `cmNamedIndexed32LSpace = cm32_32ColorPacking + cmNamedIndexedSpace + cmLittleEndianPacking`, 32-bit index, little-endian

ICC Profile Description (' desc') Tag Handling

One important change in the recent release of ColorSync (version 2.6) is how it handles the description (' desc') tag of ICC profiles.

The ' desc' tag of a profile, as defined by the ICC, contains up to three strings. The first is a required 7-bit Roman ASCII string. The second is an optional localized Unicode string. The third, also optional, is a localized string in Mac script-code format. Applications typically use one of the available strings to show the name of profiles in a list or pop-up menu. There are a few other important devilish details in the ICC definition of the ' desc' tag. One is that all three strings must be null terminated. Another is that all three strings are preceded by a character count that includes the null terminator. It is also worth noting that for the Unicode string, the character count must not be confused with a byte count because each Unicode character requires two bytes.

Previous releases of ColorSync only make partial use of this tag and, as a result, only performed limited error checking on its contents. For example, The ColorSync function

`CMGetScriptProfileDescription` would return the Mac script-code from a profile if it was present and, if not, it would return the 7-bit Roman ASCII string. The Unicode string was simply ignored and in some cases, if the Unicode and/or Mac script-code string were non-compliant, ColorSync would return garbage (or the ASCII string if you were lucky) without returning `cmProfileErr` code.

ColorSync 2.6 provides a new function, `CMGetProfileDescriptions`, to give the user access to all three possible strings in the ' desc' tag. In doing so, much stricter attention had to be paid to the compliance of the ' desc' tag. For example, if either the ASCII string or the Mac script-code strings is not null terminated, or if any of the string's character counts are invalid or beyond the range of the ' desc' tag, the `cmProfileErr` code is returned.

In order to achieve optimal performance when applications add profiles to a list or pop-up menu, ColorSync maintains a cache of all the profiles installed in the "ColorSync Profiles" folder and its sub-directories. Among other things this cache file contains the three possible names of each profile obtained by calling `CMGetProfileDescriptions`. If `CMGetProfileDescriptions` returns an error because the ' desc' tag is non-compliant, then the profile is not added to the cache. This is why non-compliant profiles, even though they are properly installed in the "ColorSync Profiles" folder, no longer show up in the ColorSync control panel or the ColorSync plug-ins pop-up menus with ColorSync 2.6 installed.

The remedy for this problem is to repair the affected profiles. Unfortunately, the "Rename Profile" AppleScript that is installed as part of ColorSync 2.6 can not be used to repair profiles with bad ' desc' tags because it can only operate on profiles in the ColorSync profile cache. Instead, a simple stand-alone tool called "Profile First Aid" to verify and repair any profile will be made available on the the ColorSync web site <<http://www.apple.com/coloursync>>.

[Back to top](#)

ColorSync 2.6 Compatibility Issues

ColorSync 1.X/2.X Support

Support for version 1.0 profiles and hybrid (1.0/2.0) color worlds has been removed in ColorSync 2.6. ColorSync 1.0 profiles, APIs and CMMs are not supported.

CMM

No longer required to support ColorSync 1.0 Profiles. Existing CMMs will be compatible.

New CMM APIs: `NCMConcatInit`, `NCMMNewLinkProfile` (see section "[New CMM APIs](#)" for the details) which a CMM may choose to implement. If a given CMM does not implement these new APIs, the default CMM is invoked instead.

Profile Searching

ColorSync 2.6 will support new profile locations (System Folder) for Profile searches made with the search functions (`CMNewProfileSearch`, etc.). Profiles in subfolders within the profiles folder will be found as well.

Color Worlds

New API `NCWConcatColorWorld` (see section "[New ColorSync 2.6 APIs](#)" for the details) puts power and responsibility of color world design into the hands of developers. Callers can select which profile tags and rendering intents to use. It also allows for new combinations of profiles which were previously not supported (e.g., more than one device-link profile).

QuickDraw Matching

2.X QuickDraw calls will continue to be supported (`N/NCMBeginMatching`, `CMEndMatching`, `N/CMDrawMatchedPicture`, `CWMatchPixmap`, `CWCheckPixmap`).

Scripting & File Formats

JPEG, GIF as well as TIFF file format plug-ins are now supported.

[Back to top](#)

Further References

- [Inside Macintosh:Managing Color With ColorSync.](#)

[Back to top](#)

Change History

- Originally written in April 1999.
- Corrected the `NCMConcatProfileSet` structure definition. 4/26/99

Downloadables



[Acrobat version of this Note \(98K\)](#)

[Back to top](#)

To contact us, please use the [Contact Us](#) page.
Updated: 26-April-99

[Technotes](#) | [Contents](#)

[Previous Technote](#) | [Next Technote](#)