Engine API 2.9

By Marti Maria

! !

http://www.littlecms.com

Contents

!
!
! . . . . !

Requeriments

Little CMS 9 p hp 33 kh hk p, 0,9 ck H k hk p ck pk ck2,2 ck p 33 ck pck, H ckckh h h p & h k 9772 9775 9757 9759 9750 9752 ck9754 p p ck kk,

Dependencies

H k hk h h ck h hkh h ck kk h kh p ph h kk ck, k p p ck h kh p p ph kk h h p h ,

h h	h H	,p h , p kh h
ı h	H ck ck p	,hı , p

Installation

Linux/unices

j p p kk ck k p ckckp p ck

!

```
! ! ! ! !
! ! ! ! /0 !
! ! ! ! !
! ! ! ! ! !
!
```

!

h k p hkp ck p p kk, H h kk j

```
! ! ! ! !
! ! ! ! ! ! !
!
```

!

h ck Hh k ck hk h p k k h k ck ckkh p ph h p k k kh , h kk h ckp p h \ p h h

p p ckckh h k p j hk

install h kk j

check hck ck ck p p

clean k ı h p hk

distclean k hk p h ckh ph h j

dist p ckh ph h hk

Windows® MS Visual Studio

p p p ı p k p hp h p ı kćk p, k h
,

Windows® Borland C++ 5.5

2,2 h p h kk p ćk, H hk *Little CMS* h kh h h , p h
pk ćk 2,2 kćk p h p ı h p ph , P j k ćkk,
hk ćk,

Apple® Mac

p h ćk p ı h : p ı . kćk p, H ćkćkh h p ćk p
ćk ph ćkh h h h ,

Other

p k ph ćk p kćk p p ćk p ćk ph ćk h h h , k
ph ćk pj h k h ćk ćkh p hp , H h ćk . pj 33 kh
hk p kćk k ćk k h ćk , H j ćk ćkćk ćk h j p k khj
h ćk h ćkP h p ćkh pj j h , k k j h ph h
p h ćk ,

Note j p h p p hk p 33 h , H ćkćk

```
.    >    : : !
```

h kćk . ćk h ćkh h h ,

Configuration toggles

lcms2.h h ck ckh ph h kk ck p h p h p pp hk p, p h h h j ck hk p hp h ck ’ k ppdhck : , ck kk h k h ***lcms2.h*** ck p p h kk k ,

	*Define this if you are **using** this package as a DLL (windows only)*
H	*Define this if you are **compiling** this package as a DLL (windows only)*
H H	*Uncomment this symbol if you are using non-supported big endian machines and the test bed hints to do so.*
H 31	*Uncomment this symbol if your compiler/machine does NOT support the "long long" type. This is automatically detected on most cases*
P	*Uncomment this if your compiler doesn't work with fast floor function. The test bed will hint to do so if necessary.*
P H H	*Uncomment this line if you want lcms2 to use the black point tag in profile, if commented, lcms2 will compute the black point by its own. Important note: It is safer to leave it commented out, as black point detection feature will work even for missing or wrong black points.*
H P H	*Define this one if you want to define the basic types elsewhere, and want **lcms2.h** to reuse those types.*
PH	*Define this one if you want strict CGATS.13 parsing. By default, Little CMS is tolerant to some issues, like missing “KEYWORD” definitions. If you want errors raised on such situations, define this symbol.*
P	*Uncomment to get rid of pthreads/windows dependency. Without pthreads only cmsDoTransform is reentrant.*
P H H H H	*For pre Windows XP compatibility. See lcms2_internal.h*

Table 1

DLL COMPILATION and use (Windows® only)

!

Little CMS ck ck h k hkh ***lcms2.h*** h h hk ck h k p hk p ckh p ,

h hk pk hk *Little CMS* ***produce*** ck ck h k H , h k ckh ck h h k p p h p p p p p p ck h hh , p h p ı hkck h p ı kck p,

!

Asserting

H p kk *Little CMS* h p k p h p h pp p , h p h ck *Little CMS* H ckh k ck k h ck hkck , ckh k h h kh ckhh k 9 h p k, k h h k p ck ck k ck j h p , H P k hkck ck h p ck,

```
_cmsAssert(a)
```

Parameters:

a: logical expression

Returns:

None

Included files (dependencies)

!

*Used by **lcms2.h***

!

```
                 ! =           /    !
                 ! =             /    !
                 ! =         /    !
                 ! =             /    !
```

!

*Used by **lcms2_plugin.h***

```
                 ! =             /    !
                 ! =         /    !
                 ! =             /    !
                 ! =             /    !
                 ! =             /    !
```

!

Used Internally

```
                 ! =           /    !
```

Generic types

h p h kk ck ck ckck h ck ***lcms2.h*** ppchck ck h h **CMS_BASIC_TYPES_ALREADY_DEFINED**, H h ck h p h k ckh ***lcms2.h***!

!

Basic Types	Bits	Signed!	Comment!
cmsUInt8Number	8	*No*	*Byte*
cmsInt8Number	8	*Yes*	
cmsUInt16Number	16	*No*	*Word*
cmsInt16Number	16	*Yes*	
cmsUInt32Number	32	*No*	*Double word*
cmsInt32Number	32	*Yes*	*Native int on most 32-bit architectures*
cmsUInt64Number	64	*No*	
cmsInt64Number	64	*Yes*	
cmsFloat32Number	32	*Yes*	*IEEE float*
cmsFloat64Number	64	*Yes*	*IEEE cmsFloat64Number*
cmsBool	?	*No*	*TRUE, FALSE Boolean type, which will be using the native integer*

Table 2

!

Derivative Types	Bits	Signed	Comment
cmsSignature	32	*No*	*Base type for ICC signatures*
cmsU8Fixed8Number	8.8 = 16	*No*	!
cmsS15Fixed16Number	15.16 = 32	*Yes*	!
cmsU16Fixed16Number	16.16 = 32	*No*	!

Table 3

!

Handles	Comment
cmsHANDLE	Generic handle
cmsHPROFILE	Handle to a profile
cmsHTRANSFORM	Handle to a color transform

Table 4

Opaque typedefs	Comment
cmsContext	Pointer to undisclosed _cms_context_struct
cmsToneCurve	Pointer to undisclosed _cms_curve_struct
cmsMLU	Pointer to undisclosed _cms_MLU_struct
cmsIOHANDLER	Pointer to undisclosed _cms_io_handler
cmsNAMEDCOLORLIST	Pointer to undisclosed _cms_NAMEDCOLORLIST_struct

Table 5

Common constants and version retrival

!

p hkh ďk h ďkh ***lcms2.h***

Version/release

P H 9747

!

Maximum number of chars in a path

923

!

Maximum number of channels in ICC profiles

53

!

Magic number to identify an ICC profile

h p	! 1 72748481! ((! !

!

Little CMS signature

k h p	! 1 7 747 84! ((!

!

!

2.8

```
int  cmsGetEncodedCMMversion (void);
```

P p k P H , h h h p k kh h p h h k p h ďk p ďk p ďk ı , j ďk p ďk p h h h , ђ , p P H ďk ďk p h ?

Parameters:

none

Returns:

k P H .

Contexts

p p h h p p kh h k h h ckh p
ckh h , p k kh p p h ck p p ck ı ckh p kh h
ckh p k h , p k h k h k p ck p h ckh
j ck p p ckck ck h p h p kk p p
k h , p kk h h k 9,3 ck h k ckkh
h , h h p h p k p p j p j kk k h
ck h ck ck ck P pp ckh h , p kck kk
kkh *cmsCreateContext* p ck kh h h h
, kckckh p k h ck h ck *Plugin* p p
p h p k p kkh k h P , k kckk p ck h ck
h pp p ckk p P ck p h , p p p
k ck ı , p h ph ck p hck h p
p h ck p ph h h p h p , H
7 h h k kck k k p P h ,

Important Note ph p 9,3 ı hck h p p ck , 9,3 p ck h ck
h ck p p h p j pck h hkh h k
p j , p kh p p ph p h h ckh kck ck
h , p ck p p
h , p h h h ck hck h p p ck , p p p k
p ck p h h ck h p p ck ,

2.6

```
cmsContext  cmsCreateContext(void* Plugin, void* UserData);
```

p h h k h ck k h , kk p h h k h p
p ck h ckck hkk p pck ck k h ckk p,

Parameters:

Plugin: Pointer to plug-in collection. Set to NULL for no plug-ins.

UserData: optional pointer to user-defined data that will be forwarded to plug-ins and logger. Set to NULL for none.

Returns:

A valid cmsContext on success, or NULL on error.

Note*: All memory used by this context is allocated by using the memory plugin, if present, this includes the block for the context itself.*

2.6!

```
cmsContext  cmsDupContext(cmsContext ContextID,  void* NewUserData);
```

kh h kk h ck k h , kk p h h k h p p
ck h ckck hk p pck ck k h ckk p,

Parameters:

UserData: optional pointer to user-defined data that will be forwarded to plug-ins and logger. Set to NULL for using user defined pointer from the source context.

Returns:

A valid cmsContext on success, or NULL on error.

2.6!

```
void  cmsDeleteContext(cmsContext ContextID);
```

p p p h ck h h ckck p k kck p,
H k p ckh P p h ,

!

Parameters:

ContextID: Handle to user-defined context.

Returns:

None

Notes:

The system context, ContextID = NULL cannot be used, the function does nothing in this case.

2.6!

```
void*  cmsGetContextUserData(cmsContext ContextID);
```

P p pck h ck h H p h pck ck p h

!
Parameters:

ContextID: Handle to user-defined context.

Returns:

Pointer to a user-defined data or NULL if no data.

Notes:

The system context, ContextID = NULL cannot be used in this function.

2.0!

```
cmsContext  cmsGetProfileContextID(cmsHPROFILE hProfile);
```

P p H h ck h h p hk ,

!
Parameters:

hProfile: Handle to a profile object

Returns:

Pointer to a user-defined context cargo or NULL if no context

2.0!

```
cmsContext  cmsGetTransformContextID(cmsHTRANSFORM hTransform);
```

!
P p H h ck h h p p ,

!
Parameters:

*hTransform**:** Handle to a color transform object.*

Returns:

Pointer to a user-defined context cargo or NULL if no context.

Plug-Ins

h k h p k H h ck h kh , h h h p kkh p ? kk p h kk k p p h k h ck hk k p ck p *Little CMS* kh p p ck hkk h p hck , k h Hck h p p p h p h , ck k h h p hp kk h k h ck h j k h P h ,

2.0

```
cmsBool cmsPlugin(void* Plugin);
```

k p p k h p h **in the global context**, k h p p kck p p k k h ck h ck k h ck k p,

Parameters:

Plugin: Pointer to plug-in collection.

Returns:

TRUE on success FALSE on error.

Notes

2.0

```
void   cmsUnregisterPlugins(void);
```

h h p p *Little CMS* k k h ck k ph h k h p ck k p ck, p h p h p h k k h h k kk k h h p h p ckh p k h h k k p h hck h h k h p h p,

Parameters:

None

Returns:

None

2.6!

```
cmsBool cmsPluginTHR(cmsContext  ContextID, void* Plugin);
```

!
H kk k h ck h h ,
!

Parameters:

ContextID: Handle to user-defined context.

Plugin: Pointer to plug-in bundle.

Returns:

TRUE on success FALSE on error.

2.6!

```
void    cmsUnregisterPluginsTHR(cmsContext ContextID);
```

h h p p h h ck k ph h k h p ck k p ck,
p h p h p h k k h h k kk k h P h
p h p ck p k h h k k p h hck h h k h
p h p,

Parameters:

ContextID: Handle to user-defined context.

Returns:

None

Error logging

!

h hk h p p p p k , p k kk p h ck p p
hk p , p p p , H h p h p ck k p j
h h hkh , p p *Little CMS 2* ck p k h h , h h hkk
english ph h k h h p , h h ck ph
h pı p p k ckh j ,

k h h kck p h p p h h k k k j ck
p p , H h p p p p hkh j h p p ck j p h
ckckck hk, ck k k pck h ,

pp p hk	h ck
PP P H	7
PP P H	5
PP P P	9
PP P H P	0
PP P	1
PP P P	2
PP P	3
PP P PH	4
PP P H	5
PP P P	3
PP P P H	57
PP P H P	55
PP P PP H	59
PP P H	50

Table 6

pp pk ph kk ck h H h p h ck, h h j
h p ck h p h k p h ck hp h ck h h,
k h p ckh kh h h p h p p, k ck k H h 7
k k ,

!
!

```
!
        !     ! )+!    M  F     I       G         * )   D        ! D         E–!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!          430      ! F     D      –!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!        !     ! +     * ◁
!
```

Definition of error logging callback.

2.0

```
void cmsSetLogErrorHandler(cmsLogErrorHandlerFunction Fn);
```

!
kk p hh k p, h h h h kk ck p h k p h p k ck, kkh h h h p p ck p k p ck k *Little CMS* k p, ck k *Little CMS* k pck h ,

!
Parameters:

Fn: Callback to the logger (user defined function), or NULL to reset Little CMS to its default logger.

Returns:

None

2.6

```
void cmsSetLogErrorHandlerTHR(cmsContext ContextID,
                              cmsLogErrorHandlerFunction Fn);
```

!
kk p hh k p p h , h h h h kk ck p h k p h p k ck, kkh h h h p p ck p k p ck k *Little CMS* k p, ck k *Little CMS* k pck h ,

!
Parameters:

ContextID: Handle to user-defined context, or NULL for the global context

Fn: Callback to the logger (user defined function), or NULL to reset Little CMS to its default logger.

Returns:

None

IO handlers

!

H ck p p p h ck ck k h hk p p , kk p ckh ph h H p hk ph p p ck hk p ck h H ck p , H ck p ck p p ck , ck ck p ph hp H ck p k h Hck h p p pck hk ,

2.0

```
cmsIOHANDLER* cmsOpenIOhandlerFromFile(cmsContext ContextID,
                                       const char* FileName,
                                       const char* AccessMode);
```

p H ck p ı p ckh j ck hk ,

Parameters:

ContextID: Pointer to a user-defined context cargo.

FileName: Full path of file resource

AccessMode: “r” to read, “w” to write.

Returns:

A pointer to an iohandler object on success, NULL on error.

!

2.0

```
cmsIOHANDLER* cmsOpenIOhandlerFromStream(cmsContext ContextID,
                                         FILE* Stream);
```

p H ck p ı p kp ck p ,

Parameters:

ContextID: Pointer to a user-defined context cargo.

Returns:

A pointer to an iohandler object on success, NULL on error.

!

2.0

```
cmsIOHANDLER* cmsOpenIOhandlerFromMem(cmsContext ContextID,
                                      void *Buffer,
                                      cmsUInt32Number size,
                                      const char* AccessMode);
```

p H ďk p ı p p k j ,

Parameters:

ContextID: Pointer to a user-defined context cargo.

Buffer: Points to a block of contiguous memory containing the data
size: Buffer's size measured in bytes.
AccessMode: "r" to read, "w" to write.

Returns:

A pointer to an iohandler object on success, NULL on error.

2.0

```
cmsIOHANDLER* cmsOpenIOhandlerFromNULL(cmsContext ContextID);
```

p hďk h ďk p ı h hk p hk h ďk p ďk kk , kk p ďk p h
p p 7 ďk k , kk ph p h ďh pďk h ďk ,

Parameters:

ContextID: Pointer to a user-defined context cargo.

Returns:

A pointer to an iohandler object on success, NULL on error.
!

2.0

```
cmsBool cmsCloseIOhandler(cmsIOHANDLER* io);
```

k h ďk p ı p h h ďkp p ,

Parameters:

io: A pointer to an iohandler object.

Returns:

TRUE on success, FALSE on error. Note that on file write operations, the real flushing to disk may happen on closing the iohandler, so it is important to check the return code.

!

2.8

```
cmsIOHANDLER* cmsGetProfileIOhandler(cmsHPROFILE   hProfile);
```

P p *iohandler* ck h p hk ı ,

Parameters:

hProfile: Handle to a profile object

Returns:

On success, a pointer to the iohandler object used by the profile. NULL on error.

Profile access funtions

!

p h h h p hk , p h k p p h
p hk h *cmsOpenProfileFromFile* ck p p p h p hk h
cmsCreateTransform, h h p p k p pp p h
cmsDoTransform, p ck p p p p hk
cmsDeleteTransform ck*cmsCloseProfile*,

2.0

```
cmsHPROFILE   cmsOpenProfileFromFile(const char *ICCProfile,
                                     const char *sAccess);
```

hk ckH p hk p p h ck h,

Parameters:

ICCProfile: File name w/ full path.

sAccess: "r" for normal operation, "w" for profile creation

Returns:

A handle to an ICC profile object on success, NULL on error.

2.0

```
cmsHPROFILE   cmsOpenProfileFromFileTHR(cmsContext ContextID,
                                        const char *ICCProfile,
                                        const char *sAccess);
```

!

ph p kk h H ck p ,

Parameters:

ContextID: Pointer to a user-defined context cargo.

ICCProfile: File name w/ full path.

sAccess: "r" for normal operation, "w" for profile creation

Returns:

A handle to an ICC profile object on success, NULL on error.

2.0

```
cmsHPROFILE  cmsOpenProfileFromStream(FILE* ICCProfile, const char* sAccess);
```

p ƈkH p hk p p h ƈk h ,

Parameters:

ICCProfile: stream holding the ICC profile.

sAccess: "r" for normal operation, "w" for profile creation

Returns:

A handle to an ICC profile object on success, NULL on error.

!

2.0

```
cmsHPROFILE  cmsOpenProfileFromStreamTHR(cmsContext ContextID,
                                         FILE* ICCProfile,
                                         const char* sAccess);
```

!

ph p kk h H ƈk p ,

Parameters:

ContextID: Pointer to a user-defined context cargo.

Returns:

A handle to an ICC profile object on success, NULL on error.

!

2.0

```
cmsHPROFILE  cmsOpenProfileFromMem(const void * MemPtr,
                                   cmsUInt32Number dwSize);
```

!

H p hk h h hp k h ƈk h p k j , k p h
ƈkƈk ƈk p hk , p h p k ƈk h , h p kƈk
kk p hk h , p h ,

Parameters:

MemPtr: Points to a block of contiguous memory containing the profile
dwSize: Profile's size measured in bytes.

Returns:

A handle to an ICC profile object on success, NULL on error.

2.0

```
cmsHPROFILE  cmsOpenProfileFromMemTHR(cmsContext ContextID,
                                       const void * MemPtr, cmsUInt32Number dwSize);
```

!

ph p kk h H ck p ,

!

Parameters:

ContextID: Pointer to a user-defined context cargo.

MemPtr: Points to a block of contiguous memory containing the profile

dwSize: Profile's size measured in bytes.

Returns:

A handle to an ICC profile object on success, NULL on error.

2.0

```
cmsHPROFILE   cmsOpenProfileFromIOhandlerTHR(cmsContext ContextID,
                                              cmsIOHANDLER* io);
```

p hk p p h ck h , p hk h ck ph ck H P , H
ck p h p p pck hk ,

Parameters:

ContextID: Pointer to a user-defined context cargo.

Io: Pointer to a serialization object.

Returns:

A handle to an ICC profile object on success, NULL on error.

2.6

```
cmsHPROFILE  cmsOpenProfileFromIOhandler2THR(cmsContext ContextID,
                                             cmsIOHANDLER* io
                                             cmsBool write);
```

p hk p p h ck h, p hk h ck ph ck H P, H
ck p h p p pck hk, h h kk h ph kk

Parameters:

ContextID: Pointer to a user-defined context cargo.

Io: Pointer to a serialization object.

write: TRUE to grant write access, FALSE to open the IOHANDLER as read only

Returns:

A handle to an ICC profile object on success, NULL on error.

2.0

```
!
cmsBool  cmsCloseProfile(cmsHPROFILE hProfile);
```

k p hk ck ck p h ckp p , p p pp p p h ckh j
p hk h h k ck ckh j ,

Parameters:

hProfile: Handle to a profile object.

Returns:

TRUE on success, FALSE on error

2.0

```
cmsBool  cmsSaveProfileToFile(cmsHPROFILE hProfile, const char* FileName);
```

p hk h hk ,

Parameters:

hProfile: Handle to a profile object

ICCProfile: File name w/ full path.

Returns:

TRUE on success, FALSE on error.!

2.0

```
cmsBool  cmsSaveProfileToStream(cmsHPROFILE hProfile, FILE* Stream);
```

!

p hk h p ,

Parameters:

hProfile: Handle to a profile object

Returns:

TRUE on success, FALSE on error.

2.0

```
cmsBool  cmsSaveProfileToMem(cmsHPROFILE hProfile,
                             void *MemPtr,  cmsUInt32Number* BytesNeeded);
```

!

ph p p p k j , H h p k k
ck ck k , H h p ck k ckh p k j h p :/7.

!

Parameters:

hProfile: Handle to a profile object.

MemPtr: Points to a block of contiguous memory with enough space to contain the profile
BytesNeeded: points to a cmsUInt32Number, where the function will store profile's size measured in bytes.

Returns:

TRUE on success, FALSE on error.

2.0

```
cmsUInt32Number  cmsSaveProfileToIOhandler(cmsHPROFILE hProfile,
                                           cmsIOHANDLER* io);
```

!

k k H P , H p p p ck p p hk p p
pp p, h ckh h ck h ph k h p k k ck,

Parameters:

hProfile: Handle to a profile object

Io: Pointer to a serialization object.

Returns:

The number of bytes used to store the profile, or zero on error.

Predefined virtual profiles

!

2.0

```
cmsHPROFILE    cmsCreateProfilePlaceholder(cmsContext ContextID);
```

!

p p hk ı p ck k ck p p p,

WARNING h ck p hk h ckckh h p h h ckp k k ,

!

Parameters:

ContextID: Pointer to a user-defined context cargo.

Returns:

A handle to an ICC profile object on success, NULL on error.

2.0

```
cmsHPROFILE    cmsCreateRGBProfile(const cmsCIExyY* WhitePoint,
                                    const cmsCIExyYTRIPLE* Primaries,
                                    cmsToneCurve* const TransferFunction[3]);
```

!

h h p ckh k P p hk ck h h ph ph ck p p h ,
H k kk h ?! h p ck pckP h k p hk ! ck H ckck p
ckck ck HH p h h ,

5	h p hk ph h
9	h ckh h h
0	h P ck k p
1	h p k p
2	h k k p
3	h P ck P
4	h p P
5	h k P
3	p h ck h
57	h p h h

Parameters:

WhitePoint: The white point of the RGB device or space.

Primaries: The primaries in xyY of the device or space.

TransferFunction[]: 3 tone curves describing the device or space gamma.

Returns:

A handle to an ICC profile object on success, NULL on error.

2.0

```
cmsHPROFILE   cmsCreateRGBProfileTHR(cmsContext ContextID,
                        const cmsCIExyY* WhitePoint,
                        const cmsCIExyYTRIPLE* Primaries,
                        cmsToneCurve* const TransferFunction[3]);
```

!

ph p kk h H ďk p ,

!

Parameters:

ContextID: Pointer to a user-defined context cargo.

WhitePoint: The white point of the RGB device or space.

Primaries: The primaries in xyY of the device or space.

TransferFunction[]: 3 tone curves describing the device or space gamma.

Returns:

A handle to an ICC profile object on success, NULL on error.

2.0

```
cmsHPROFILE   cmsCreateGrayProfile(const cmsCIExyY* WhitePoint,
                                    const cmsToneCurve* TransferFunction);
```

!

h h p p p hk ďk h h ďk p p h , H k
kk h ? h p ďk pďk p ďh k p hk ,

!

!

5	h p hk ph h
9	h ďh h h
0	h p P

!

!

!

!

!

!

!

!
Parameters:

WhitePoint: The white point of the gray device or space.

TransferFunction: tone curve describing the device or space gamma.

Returns:

A handle to an ICC profile object on success, NULL on error.

!

2.0

```
cmsHPROFILE   cmsCreateGrayProfileTHR(cmsContext ContextID,
                          const cmsCIExyY* WhitePoint,
                          const cmsToneCurve* TransferFunction);
```

ph p kk h H ck p ,

Parameters:

ContextID: Pointer to a user-defined context cargo.

WhitePoint: The white point of the gray device or space.

TransferFunction: tone curve describing the device or space gamma.

Returns:

A handle to an ICC profile object on success, NULL on error.

2.0

```
cmsHPROFILE  cmsCreateLinearizationDeviceLink(cmsColorSpaceSignature Space,
                              cmsToneCurve* const TransferFunctions[]);
```

!
h h ck h kh j p h h p k p h p p h
,

!
Parameters:

Space: any cmsColorSpaceSignature from Table 10

TransferFunction[]: tone curves describing the device or space linearization.

Returns:

A handle to an ICC profile object on success, NULL on error.

2.0

```
cmsHPROFILE   cmsCreateLinearizationDeviceLinkTHR(cmsContext ContextID,
                                    cmsColorSpaceSignature ColorSpace,
                                    cmsToneCurve* const TransferFunctions[]);
```

!

ph p kk h H ck p ,

!

Parameters:

ContextID: Pointer to a user-defined context cargo.

ColorSpace: any cmsColorSpaceSignature from Table 10

TransferFunction[]: tone curves describing the device or space linearization.

Returns:

A handle to an ICC profile object on success, NULL on error.

2.0

```
cmsHPROFILE cmsCreateInkLimitingDeviceLink(cmsColorSpaceSignature Space,
                                           cmsFloat64Number Limit);
```

!

h h ck h kh j p h h ph j kh hh ,

Ink-limiting algorithm:

```
! ! !
! ! T   ! >! D! ,! N! ,!   ! ,! L! !
! !   ! T   !   !     M     ! !
! ! ! ! ! ! ! ! S     >! 2! .!  )T   ! .!     M     *! 0!  )D! ,! N! ,!   *!
! ! ! ! ! ! ! !   ! S     ! =1! !
! ! ! ! ! ! ! ! ! ! ! ! ! ! S     >1!
! ! ! ! ! ! ! !       ! ! ! ! ! !
! ! F   ! !
! ! ! ! ! S     >2!
! !       !
!
! ! D! >! S     ! +! D!
! ! N! >! S     ! +! N!
! !   ! >! S     ! +!   !
! ! L ;! E   !   !       !
!
```

!
!
!
!
!

Parameters:

Space: any cmsColorSpaceSignature from Table 10. Currently only cmsSigCmykData is supported.

Limit: Amount of ink limiting in % (0..400%)

Returns:

A handle to an ICC profile object on success, NULL on error.

2.0

```
cmsHPROFILE cmsCreateInkLimitingDeviceLinkTHR(cmsContext ContextID,
                                               cmsColorSpaceSignature Space,
                                               cmsFloat64Number Limit);
```

!

ph p kk h H ck p ,

!

Parameters:

ContextID: Pointer to a user-defined context cargo.

Space: any cmsColorSpaceSignature from Table 10. Currently only cmsSigCmykData is supported.

Limit: Amount of ink limiting in % (0..400%)

Returns:

A handle to an ICC profile object on success, NULL on error.

2.0

```
cmsHPROFILE   cmsCreateLab2Profile(const cmsCIExyY* WhitePoint);
```

!

p → hck h pj h h 9 H p hk , ck p ck h
ck h kk ck *Little CMS* h h p hk ,

Parameters:

WhitePoint: Lab reference white. NULL for D50.

Returns:

A handle to an ICC profile object on success, NULL on error.

2.0

```
cmsHPROFILE   cmsCreateLab2ProfileTHR(cmsContext ContextID,
                                      const cmsCIExyY* WhitePoint);
```

ph p kk h H ck p ,

Parameters:

ContextID: Pointer to a user-defined context cargo.

WhitePoint: Lab reference white. NULL for D50.

Returns:

A handle to an ICC profile object on success, NULL on error.

2.0

```
cmsHPROFILE   cmsCreateLab4Profile(const cmsCIExyY* WhitePoint);
```

!

p → hck h pjh h 1 H p hk ,

Parameters:

WhitePoint: Lab reference white. NULL for D50.

Returns:

A handle to an ICC profile object on success, NULL on error.

2.0

```
cmsHPROFILE   cmsCreateLab4ProfileTHR(cmsContext ContextID,
                                      const cmsCIExyY* WhitePoint);
```

!

ph p kk h H ck p ,

!

Parameters:

ContextID: Pointer to a user-defined context cargo.

WhitePoint: Lab reference white. NULL for D50.

Returns:

A handle to an ICC profile object on success, NULL on error.

!
!
!
!
!

2.0

```
cmsHPROFILE   cmsCreateXYZProfile(void);
```

p → hck h pj h h 1 H p hk , h h ck h k
k ph ph h h 27,

Parameters:

None

Returns:

A handle to an ICC profile object on success, NULL on error.

2.0

```
cmsHPROFILE   cmsCreateXYZProfileTHR(cmsContext ContextID);
```

!

ph p kk h H ck p ,

!

Parameters:

ContextID: Pointer to a user-defined context cargo.

Returns:

A handle to an ICC profile object on success, NULL on error.

!

2.0

```
cmsHPROFILE   cmsCreate_sRGBProfile(void);
```

!

p H hp k p hk p P , P h ck pckP k p p ck
p h k ck h p h 5333 p h p ph p ck H p ,

sRGB white point is D65.	
xyY	7,0594 7,0935 5,7

Primaries are ITU-R BT.709-5 (xYY)	
R	7,3177 7,0077 5,7
G	7,0777 7,3777 5,7
B	7,5277 7,7377 5,7

sRGB transfer functions are defined by:

```
   ! ! S   SHC -H   SHC -! C   SHC! =! 1 /15156!
!
! ! ! ! S! >! ! S   SHC! 0! 23 /: 3!
! ! ! ! H! >! ! H   SHC! 0! 23 /: 3!
! ! ! ! C! >! ! C   SHC! 0! 23 /: 3!
!
!
      !   ! ! S   SHC -H   SHC -! C   SHC!   >! 1 /15156!
!
! ! ! ! S! >!  ) )S   SHC! ,! 1 /166 *! 0! 2 /166 *3 /5!
```

2.0

```
cmsHPROFILE   cmsCreateNULLProfileTHR(cmsContext ContextID);
```

!

ph p kk h H ck p ,

!

Parameters:

ContextID: Pointer to a user-defined context cargo.

Returns:

A handle to an ICC profile object on success, NULL on error.

2.0

```
cmsHPROFILE   cmsCreateBCHSWabstractProfile(int nLUTPoints,
                                  cmsFloat64Number Bright,
                                  cmsFloat64Number Contrast,
                                  cmsFloat64Number Hue,
                                  cmsFloat64Number Saturation,
                                  int TempSrc,
                                  int TempDest);
```

!

p p ck h kh j p h h p ph p p h ck h
h p k h , h h p hh ck p p

Parameters:

nLUTPoints : Resulting color map resolution

Bright: Bright increment. May be negative

Contrast : Contrast increment. May be negative.

Hue : Hue displacement in degree.

Saturation: Saturation increment. May be negative

TempSrc: Source white point temperature

TempDest: Destination white point temperature.

Returns:

A handle to an ICC profile object on success, NULL on error.

Notes

To prevent white point adjustment, set TempSrc = TempDest = 0

2.0

```
cmsHPROFILE   cmsCreateBCHSWabstractProfileTHR(cmsContext ContextID,
                                              int nLUTPoints,
                                              cmsFloat64Number Bright,
                                              cmsFloat64Number Contrast,
                                              cmsFloat64Number Hue,
                                              cmsFloat64Number Saturation,
                                              int TempSrc,
                                              int TempDest);
```

!

ph p kk h H ck p ,

!

Parameters:

ContextID: Pointer to a user-defined context cargo.

nLUTPoints : Resulting colormap resolution

Bright: Bright increment. May be negative

Contrast : Contrast increment. May be negative.

Hue : Hue displacement in degree.

Saturation: Saturation increment. May be negative

TempSrc, TempDest: Source, Destination white point temperatures

Returns:

A handle to an ICC profile object on success, NULL on error.!

2.0

```
cmsHPROFILE   cmsTransform2DeviceLink(cmsHTRANSFORM hTransform,
                                      cmsFloat64Number Version,
                                      cmsUInt32Number dwFlags);
```

!

p ck h kh j p hk p h k p p p , h p hk ck
p h h p hk ck , ckh hh ck p h p
h k h ck h kh j p , ck p h p h p 5,7; 1,0

Parameters:

hTransform: Handle to a color transform object.

Version: The target devicelink version number.

dwFlags: A combination of bit-field constants described in k 19.

Returns:

A handle to an ICC profile object on success, NULL on error.

Obtaining localized info from profiles

H p h ph p 1,7 H p ck h ck p hp ck ck h p ck HH h ck ck ph ck p h p hk ck ph h p ckh k p , p h p p kk ck p hk k kh ck p k k p h ck p ph ck , p hk ck p ck h k kh ck p h ckh p ph , H k p ck p k ck h k kh ck p hk ckck ckh h h ck h ckh p k , h ck h 1 h *Little CMS* k kk h ph : k . ck khk kh ck h ck , p h kk p H ck k h h h ck . p ck hk ck kk h ckh k ph ph *Little CMS* p hck h khh ckh p p p ,

HHh ph k 4 h ck hck p h p p h p h h p hk , k kh h ph j h ck h k ck p ck h p ck k , p p p HHk p , kh ck ck p

Language Code: k ,k , ck pck h 303 9 h 303ı , k

Country Codes: ,h , h p ck p h h 0533 h ck , k

H p h **“en”** p’ kh : ck**“US”** p’ h ck : p h k ckh p hk , H h j k ck p h p hk ck h k hh k ck p , *Little CMS* hkk p p p p ,

H ck . p ck ı j hp ph h p hk

For the language:

```
    O M        !
```

For the country:

```
    O D       !
```

h hkk p p hp ph h p h , p h hkk ph k hck ck p k h , H h p h ck k p k ckk h k p p p k h j ph hk k ,

```
        !     !  !
!!!!!!!!!!!!!       E          !! >! 1-!
!!!!!!!!!!!!!       N           ! >! 2-!
!!!!!!!!!!!!!       N     !!!!!!!! >! 3-!
!!!!!!!!!!!!!       D         !!!! >! 4!
 !           ◁
```
!

2.0

```
cmsUInt32Number cmsGetProfileInfo(cmsHPROFILE hProfile,
                                  cmsInfoType Info,
                                  const char LanguageCode[3],
                                  const char CountryCode[3],
                                  wchar_t* Buffer,
                                  cmsUInt32Number BufferSize);
```

!

p kh p h ph p p hk ck kh h k kh h , ph p p p ck hck p ,

Parameters:

hProfile: Handle to a profile object

Info: A selector of which info to return

Language Code: first name language code from ISO-639/2.
Country Code: first name region code from ISO-3166.
Buffer: pointer to a memory block to get the result. NULL to calculate size only
BufferSize: Amount of byes allocated in Buffer, or 0 to calculate size only.

Returns:

Number of required bytes to hold the result. 0 on error.

2.0

```
cmsUInt32Number cmsGetProfileInfoASCII(cmsHPROFILE hProfile,
                                       cmsInfoType Info,
                                       const char LanguageCode[3],
                                       const char CountryCode[3],
                                       char* Buffer,
                                       cmsUInt32Number BufferSize);
```

p kh p h ph p p hk ck kh h k kh h , ph p p p ck HH,

Parameters:

hProfile: Handle to a profile object

Info: A selector of which info to return

Language Code: first name language code from ISO-639/2.
Country Code: first name region code from ISO-3166.
Buffer: pointer to a memory block to get the result. NULL to calculate size only
BufferSize: Amount of byes allocated in Buffer, or 0 to calculate size only.

Returns:

Number of required bytes to hold the result. 0 on error.

Profile feature detection

! !

2.0

```
cmsBool  cmsDetectBlackPoint(cmsCIEXYZ* BlackPoint,
                             cmsHPROFILE hProfile,
                             cmsUInt32Number Intent,
                             cmsUInt32Number dwFlags);
```

!

h k j h h p hk , ck k j h h k ph ,

Parameters:

BlackPoint: Pointer to H object to receive the detected black point.
hProfile: Handle to a profile object
Intent: A H 09 pholding the intent code, as described in H section.
dwFlags: reserved (unused). Set it to 0

Returns:

TRUE on success, FALSE on error

2.8

```
cmsBool  cmsDetectDestinationBlackPoint(cmsCIEXYZ* BlackPoint,
                             cmsHPROFILE hProfile,
                             cmsUInt32Number Intent,
                             cmsUInt32Number dwFlags);
```

!

h k j h h ck h h p hk h k j h h H
k ph ,

Parameters:

BlackPoint: Pointer to H object to receive the detected black point.
hProfile: Handle to a profile object
Intent: A H 09 pholding the intent code, as described in H section.
dwFlags: reserved (unused). Set it to 0

Returns:

TRUE on success, FALSE on error

!

2.0

```
cmsFloat64Number  cmsDetectTAC(cmsHPROFILE hProfile);
```

!

p k k p p ph ck p p h kh h h j
p, h h k ck p h p pp ck h p H k H j
p p k p p , h h ck h k p p p
h p hk h , k pj p hk , P p hk 177 h p p ck, h
ck ck kh k p 3 41 41 h ,

Parameters:

hProfile: Handle to a profile object

Returns:

h ck p p h 7 pp p,

Accessing profiler header

!

2.0

```
cmsBool cmsGetHeaderCreationDateTime(cmsHPROFILE hProfile, struct tm *Dest);
```

!

P p ck ck h p hk p ck, h h h kck p ckh p hk ck p,

!

Parameters:

hProfile: Handle to a profile object

Dest: pointer to struct tm object to hold the result.

Returns:

TRUE on success, FALSE on error

2.0

```
cmsUInt32Number cmsGetHeaderFlags(cmsHPROFILE hProfile);
```

!

ck p k h H p hk ı , p hk k h kckck h k h ckh
ph h p ckh ph ck p h ck h h , k h hh
53 h p p p ck p H , k h h hh 7 ck5 kk ck h ckh ckh k 4,

Position	Field Length (bits)	Field Contents

Table 7

!

Parameters:

hProfile: Handle to a profile object

Returns:

Flags field of profile header.!

!
!
!
!
!
!

!

2.0

```
void   cmsSetHeaderFlags(cmsHPROFILE hProfile, cmsUInt32Number Flags);
```

!

ck p k h H p hk ı , khck k p ck h ckh k 4,

Parameters:

hProfile: Handle to a profile object.

Flags: Flags field of profile header.!

Returns:

None

!

2.0

```
cmsUInt32Number   cmsGetHeaderManufacturer(cmsHPROFILE hProfile);
```

!

P p p p h p ck ph ckh ck p, h h kh h hck k

p ck ck p p , k h kck p p hk ,

! !

Parameters:

hProfile: Handle to a profile object

Returns:

p hk p p h p p ckh ck p,

2.0

```
void  cmsSetHeaderManufacturer(cmsHPROFILE hProfile,
                               cmsUInt32Number manufacturer);
```

!

p p h p h ck p, h h kh h hck k p ck ck

p p , k h kck p p hk ,

!

Parameters:

hProfile: Handle to a profile object.

Manufacturer: The profile manufacturer signature to store in the header.

Returns:

None

!
!

2.0

```
cmsUInt32Number  cmsGetHeaderModel(cmsHPROFILE hProfile);
```

!
P p ck k h p ck ph ckh ck p, h h kh h hck k p ck ck
ck k , k h kck p p hk ,

!
Parameters:

hProfile: Handle to a profile object

Returns:

The profile model signature stored in the header.

2.0

```
void  cmsSetHeaderModel(cmsHPROFILE hProfile, cmsUInt32Number model);
```

!
ck k h p h p hk ck p, h h kh h hck k p ck ck ck k
, k h kck p p hk ,

!
Parameters:

hProfile: Handle to a profile object
model: The profile model signature to store in the header.

Returns:

None

!

Device attributes

pp k ck h ck k pp ck k 1 5 ph h ,!

P k h	p p
k	

Table 8

!

2.0

```
void  cmsGetHeaderAttributes(cmsHPROFILE hProfile , cmsUInt64Number* Flags );
```

!

ph k ck ph ckh k 5,

Parameters:

hProfile: Handle to a profile object

Flags: a pointer to a cmsUInt64Number to receive the flags.

Returns:

None

2.0

```
void  cmsSetHeaderAttributes(cmsHPROFILE hProfile, cmsUInt64Number Flags);
```

ph k h p hk ck p, k p p ckh k 5,

Parameters:

hProfile: Handle to a profile object

Flags: The flags to be set.

Returns:

None

Profile classes

h k p hk k h p	
h H k	7 40303 49 p
h h k k	7 3 3 41 49 p
h k	7 47494149 p p
h h j k	7 3 333 3 kh j
h p k	7 35394041
h k p k	7 40473530
h ck k p k	7 3 3ck303 k

Table 9

2.0

```
cmsProfileClassSignature  cmsGetDeviceClass(cmsHPROFILE hProfile);
```

ck h k h p p p hk ck p,

Parameters:

hProfile: Handle to a profile object

Returns:

Device class of profile as described in k 3

2.0

```
void    cmsSetDeviceClass(cmsHPROFILE hProfile, cmsProfileClassSignature sig);
```

ck h k h p h p hk ck p,

Parameters:

hProfile: Handle to a profile object

sig: Device class of profile as described in k 3

Returns:

None

!

Profile versioning

2.0

```
void    cmsSetProfileVersion(cmsHPROFILE hProfile, cmsFloat64Number Version);
```

H p h h p hk ck p, p h h h h h k , !

Parameters:

hProfile: Handle to a profile object

Version: Profile version in readable floating point format.

Returns:

None

2.0

```
cmsFloat64Number    cmsGetProfileVersion(cmsHPROFILE hProfile);
```

!

P p p hk H p h , p h h ck ck ck p ck k k h h p ,!

Parameters:

hProfile: Handle to a profile object

Returns:

The profile ICC version, in readable floating point format.!

2.0

```
cmsUInt32Number   cmsGetEncodedICCversion(cmsHPROFILE hProfile);
```

!

P p p hk H p h h p h h p ckh ck p,

!

Parameters:

hProfile: Handle to a profile object

Returns:

The encoded ICC profile version.

!

2.0

```
void  cmsSetEncodedICCversion(cmsHPROFILE hProfile,
                              cmsUInt32Number Version);
```

!

H ph h p hk ck p h ck ckh ,

!

Parameters:

hProfile: Handle to a profile object

Version: Profile version in the same format as it will be stored in profile header.

Returns:

None

Info on profile implementation

2.0

```
cmsBool   cmsIsMatrixShaper(cmsHPROFILE hProfile);
```

P p p ph ph p h p hk , p hk kck ph p ck kk,

Parameters:

hProfile: Handle to a profile object

Returns:

TRUE if the profile holds a matrix-shaper, FALSE otherwise.

!

2.1

```
cmsBool   cmsIsCLUT(cmsHPROFILE hProfile,
                    cmsUInt32Number Intent,
                    cmsUInt32Number UsedDirection);
```

!

P p p h p h p hk p h h ckckp h ,

!

Parameters:

hProfile: Handle to a profile object

H *cmsUInt32Number holding the intent code, as described in Intents section.*

Direction: any of following values:

```
! MDNT  TFE BT  OQ  !!!!!! 1!
! MDNT  TFE BT P  Q  !!!!! 2!
! MDNT  TFE BT QSPPG!!!!!! 3!
```

!

Returns:

TRUE if a CLUT is present for given intent and direction, FALSE otherwise.

!

Color spaces

k p h p	
h	1 696:6B31! (! (!
h	1 5D727331! (M ! (!
h	1 5D868731! (M ! (!
h p	1 6:547383! (D (!
h	1 6:898:31! (! (!
h P	1 63585331! (SHC! (!
h p	1 5863526:! (HSB (!
h	1 59646731! (IT ! (!
h k	1 595D6431! (IMT! (!
h j	1 545E6:5C! (DN L(!
h	1 545E6:31! (DN ! (!
h 5	1 5E545942! (NDI2(!!
h 9	1 5E545943! (NDI3(!
h 0	1 5E545944! (NDI4(!!
h 1	1 5E545945! (NDI5(!
h 2	1 5E545946! (NDI6(!!
h 3	1 5E545947! (NDI7(!!
h 4	1 5E545948! (NDI8(!!
h 5	1 5E545949! (NDI9(!!
h 3	1 5E54594:! (NDI:(!!
h	1 5E54594B! (NDIB(!!
h	1 5E54594C! (NDIC(!!
h	1 5E54594D! (NDID(!!
h	1 5E54594E! (NDIE(!!
h	1 5E54594F! (NDIF(!!
h	1 5E54594G! (NDIG(!!
h ƈk	1 7 7 747 ! ((!
h 5 k p	1 42545D63! (2DMS(!!
h 9 k p	1 43545D63! (3DMS(!
h 0 k p	1 44545D63! (4DMS(!
h 1 k p	1 45545D63! (5DMS(!
h 2 k p	1 46545D63! (6DMS(!
h 3 k p	1 47545D63! (7DMS(!
h 4 k p	1 48545D63! (8DMS(!
h 5 k p	1 49545D63! (9DMS(!
h 3 k p	1 4:545D63! (:DMS(!
h 57 k p	1 52545D63! (BDMS(!
h 55 k p	1 53545D63! (CDMS(!
h 59 k p	1 54545D63! (DDMS(!
h 50 k p	1 55545D63! (EDMS(!
h 51 k p	1 56545D63! (FDMS(!
h 52 k p	1 57545D63! (GDMS(!
h	1 5D86875C! (M L(!

Table 10

2.0

```
cmsUInt32Number   cmsChannelsOf(cmsColorSpaceSignature ColorSpace);
```

P   p   k   p   h   k p   ,

Parameters:

ColorSpace: any cmsColorSpaceSignature from Table 10

Returns:

Number of channels, or 3 on error.

2.0

```
cmsColorSpaceSignature   cmsGetPCS(cmsHPROFILE hProfile);
```

p   hk   h   ck   h   p   hk   h   H   h   ,

Parameters:

hProfile: Handle to a profile object

Returns:

Obtained cmsColorSpaceSignature (Table 10).

2.0

```
void   cmsSetPCS(cmsHPROFILE hProfile, cmsColorSpaceSignature pcs);
```

p   hk   h   h   p   h   p   hk   ck p   h   H   h   ,

Parameters:

hProfile: Handle to a profile object
pcs: any cmsColorSpaceSignature from Table 10

Returns:

None

2.0

```
cmsColorSpaceSignature   cmsGetColorSpace(cmsHPROFILE hProfile);
```

!

k p ck h p hk h H h ,

Parameters:

hProfile: Handle to a profile object

Returns:

Obtained cmsColorSpaceSignature (Table 10).

2.0

```
void   cmsSetColorSpace(cmsHPROFILE hProfile, cmsColorSpaceSignature sig);
```

!

k p h p h p hk ck p h H h ,

Parameters:

hProfile: Handle to a profile object
sig: any cmsColorSpaceSignature from Table 10

Returns:

None

Containers in floating point format

H	
k 31 p	?
k 31 p	?
k 31 p	?

Table 11

H	
k 31 p	?
k 31 p	?
k 31 p	?

Table 12

H	
k 31 p	?
k 31 p	?
k 31 p	?

Table 13

H	
k 31 p	?
k 31 p	?
k 31 p	?

Table 14

k 31 p	?
k 31 p	?
k 31 p	?

Table 15

H PH	
H	P ck
H	p ?
H	k ?

Table 16

H PH	
H	P ck
H	p ?
H	k ?

Table 17

Colorspace conversions

D50 XYZ normalized to Y=1.0	
27	7,3319
27	5,7
27	7,5913

Table 18

V4 perceptual black	
P	7,77003
P	7,7701405
P	7,77954

Table 19

2.0

```
const cmsCIEXYZ* cmsD50_XYZ(void);
const cmsCIExyY* cmsD50_xyY(void);
```

P p h p p p ,

Parameters:

None

Returns:

Pointers to constant D50 white point in XYZ and xyY spaces.!

2.0

```
void cmsXYZ2xyY(cmsCIExyY* Dest, const cmsCIEXYZ* Source);
void cmsxyY2XYZ(cmsCIEXYZ* Dest, const cmsCIExyY* Source);
```

k ph ph p h ,

Parameters:

Source, Dest: Source and destination values.

Returns:

None

! !

2.0

```
void cmsXYZ2Lab(const cmsCIEXYZ* WhitePoint,
                cmsCIELab* Lab,
                const cmsCIEXYZ* xyz);

void cmsLab2XYZ(const cmsCIEXYZ* WhitePoint,
                cmsCIEXYZ* xyz,
                const cmsCIELab* Lab);
```

k ph ph p h , h h h p 27 h h ,

Parameters:

Lab: Pointer to a cmsCIELab value as described in Table 13

xyz: Pointer to a cmsCIEXYZ value as described in k 55

Returns:

None

!

2.0

```
void cmsLab2LCh(cmsCIELCh*LCh, const cmsCIELab* Lab);
void cmsLCh2Lab(cmsCIELab* Lab, const cmsCIELCh* LCh);
```

!

k ph ph p h ,

!

Parameters:

Lab: Pointer to a cmsCIELab value as described in Table 13

LCh: Pointer to a cmsCIELCh value as described in k 51

Returns:

None

!

Encoding /Decoding on PCS

2.0

```
void cmsLabEncoded2Float(cmsCIELab* Lab, const cmsUInt16Number wLab[3]);
```

ꞓk k ꞓk ꞓk H 1 h *a cmsCIELab value as described in Table 13*

Parameters:

Lab: Pointer to a cmsCIELab value as described in Table 13
wLab[] : Array of 3 cmsUInt16Number holding the encoded values.

Returns:
None

2.0

```
void cmsLabEncoded2FloatV2(cmsCIELab* Lab, const cmsUInt16Number wLab[3]);
```

!
ꞓk k ꞓk ꞓk H 9 h H k *as described in Table 13*

Parameters:

Lab: Pointer to a cmsCIELab value as described in Table 13
wLab[] : Array of 3 cmsUInt16Number holding the encoded values.

Returns:
None

2.0

```
void cmsFloat2LabEncoded(cmsUInt16Number wLab[3], const cmsCIELab* Lab);
```

ꞓk k *from a cmsCIELab value as described in Table 13* H 1 h ,

Parameters:
Lab: Pointer to a cmsCIELab value as described in Table 13
wLab[] : Array of 3 cmsUInt16Number to hold the encoded values.

Returns:
None

2.0

```
void cmsFloat2LabEncodedV2(cmsUInt16Number wLab[3], const cmsCIELab* Lab);
```

!
ʗk k *from a cmsCIELab value as described in Table 13* H 9 h ,

Parameters:

Lab: Pointer to a cmsCIELab value as described in Table 13

wLab[] : Array of 3 cmsUInt16Number to hold the encoded values.

Returns:

None

2.0

```
void cmsXYZEncoded2Float(cmsCIEXYZ* fxyz, const cmsUInt16Number XYZ[3]);
```

ʗk k ʗk ʗk H h H k *as described in* k 55

Parameters:

fxyz: Pointer to a cmsCIEXYZ value as described in k 55

XYZ[] : Array of 3 cmsUInt16Number holding the encoded values.

Returns:

None

2.0

```
void cmsFloat2XYZEncoded(cmsUInt16Number XYZ[3], const cmsCIEXYZ* fXYZ);
```

!
ʗk k *from a cmsCIELab value as described in* k 55 H h ,

Parameters:

XYZ[] : Array of 3 cmsUInt16Number to hold the encoded values.

fxyz: Pointer to a cmsCIEXYZ value as described in k 55

Returns:

None

Accessing tags

Tag types

p p ck h ck , ckck p h k h ,
k h Hp p p p pck hk ,

!

ck h hh T	
h p hh	1 7479837E! ((!
h k p pck p	1 747D837G! ((!
h k p k	1 747D8385! ((!
h pckH	1 7483757:! ((!
h p	1 74868387! ((!
h	1 75728572! ((!
h h	1 75857:7E! ((!
h h h	1 75768784! ((!
h 53	1 7 778543! (3 (!
h 5	1 7 778542! (2 (!
h	1 7 525331! (BC! (!
h	1 7 535231! (CB! (!
h p	1 7E767284! ((!
h kh kh ck h ck	1 7E7D8674! ((!
h kh p k	1 7E817685! ((!
h ck k p	1 7F747 7D! ((!
h ck k p9	1 7F747D43! (3 (!
h p ph p	1 81728372! ((!
h p hk	1 81847682! ((!
h p hk Hck	1 81847:75! ((!
h P p 53	1 83748443! (3 (!
h 52 h ck53 pp	1 84774443! (43 (!
h p h	1 8474837F! ((!
h h p	1 847:7831! (! (!
h	1 85768985! ((!
h ph h	1 75768474! ((!
h 53 h ck53 pp	1 86774443! (43 (!
h p	1 73777531! (! (!
h H 53 pp	1 867:4247! (27 (!
h H 09 pp	1 867:4443! (43 (!
h H 31 pp	1 867:4745! (75 (!
h H 5 pp	1 867:4149! (19 (!
h h h ckh	1 877:7688! ((!
h	1 696:6B31! (! (!

Table 20

!

Tags

p p ck h ck , ckck p h k h , k h H p p p p p ck hk , ph p h k p p h p P ck ck ph ,

!

ck h hh T		k
h 7	1 52435341! (B3C1(!!	Q !
h 5	1 52435342! (B3C2(!	Q !
h 9	1 52435343! (B3C3(!!	Q !
h k k p	1 73696:6B! ((!	D F !
h k ph k	1 73696:6B! ((!	D F !
h k P	1 73656354! (SD(!	D !
h 7	1 53435241! (C3B1(!	Q !
h 5	1 53435242! (C3B2(!	Q !
h 9	1 53435243! (C3B3(!	Q !
h kh p h h	1 74727D85! ((!	! !
h p p	1 85728378! ((!!	NM !
h p h ck h	1 74797275! ((!	D F ! 4 !
h p h h	1 7479837E! ((!	D F S QMF!
h k p pck p	1 747D837G! ((!	9O ! 27 !
h k p k	1 747D8385! ((!	OBNFEDPMPSM T !
h k p k	1 747D7G85! ((!	OBNFEDPMPSM T !
h k ph ph H H	1 747:7:84! ((!	T !
h ph	1 74818385! ((!	NM !
h pckH (	1 7483757:! ((!!	OBNFEDPMPSM T !
h	1 75728572! ((!!	DDE !
h h	1 75857:7E! ((!!	! !
h h	1 757E7F75! ((!	NM !
h h ck k	1 757E7575! ((!	NM !
h h h	1 75768784! ((!!	O ! +!
h 7	1 55435341! (E3C1(!	Q !
h 5	1 55435342! (E3C2(!	Q !
h 9	1 55435343! (E3C3(!	Q !
h 0	1 55435344! (E3C4(!	Q !
h 7	1 53435541! (C3E1(!	Q !
h 5	1 53435542! (C3E2(!	Q !
h 9	1 53435543! (C3E3(!	Q !
h 0	1 53435544! (C3E4(!	Q !
h	1 78727E85! ((!	Q !
h p P	1 7 656354! (SD(!	D !
h p k p	1 78696:6B! ((!	D F !
h p ph k	1 78696:6B! ((!	D F !
h p P	1 78656354! (SD(!	D !
h h	1 7D867 7:! ((!	D F !
h p	1 7E767284! ((!	DDN D !

h ck h k j h	1 737C8185! ((!	D F !
h ck h h h	1 88858185! ((!	D F !
h ck k p	1 7F747 7D! ((! !	O ! +!
h ck k p9	1 7F747D43! (3(!	OBNFEDPMPSM T !
h P	1 83768481! ((!	O ! +!
h p kP ck ph H	1 837:7841! (1(!	T !
h p h 7	1 81837641! (1(!	Q !
h p h 5	1 81837642! (2(!	Q !
h p h 9	1 81837643! (3(!	Q !
h p hk ph h	1 75768474! ((!	NM !
h p hk	1 81847682! ((!	TFR!
h p hk hck	1 81847:75! ((!	TFR!
h 9 P 7	1 81847541! (1(! !	DDE !
h 9 P 5	1 81847542! (2(! !	DDE !
h 9 P 9	1 81847543! (3(! !	DDE !
h 9 P 0	1 81847544! (4(! !	DDE !
h 9	1 81844384! (3 (! !	DDE !
h 9P ck ph H	1 8184437:! (3 (! !	DDE !
h P ck k p	1 83696:6B! ((!	D F !
h P ck ph k	1 83696:6B! ((!	D F !
h P ck P	1 83656354! (SD(!	D !
h p h P ck ph H	1 837:7843! (3(!	T !
h p h	1 84748375! ((! !	NM !
h p h	1 8474837F! ((! !	T !
h k	1 85767479! ((!	T !
h p	1 73777531! (! (! !	C !
h h h ck	1 87867675! ((!	NM !
h h h ckh h	1 877:7688! ((!	DD D !
h	1 7E768572 ((!	IBOEMF!)E D *! !

Table 21

*cmsSigCrdInfoTag h h ph p ck h h p hk pp ck ck h P , h k p hk p k h k P , H h h k ck h k ck ck p ph p k

- ph p ck
- 7 P ck ph h 7 P
- 5 P ck ph h 5 P
- 9 P ck ph h 9 P
- 0 P ck ph h 0 P

p p p k p ck p kh ck k h k h p p ck,

Not supported	Why
cmsSigOutputResponseTag	*POSSIBLE PATENT ON THIS SUBJECT!*
cmsSigNamedColorTag	*Deprecated*
cmsSigDataTag	*Ancient, unused*
cmsSigDeviceSettingsTag	*Deprecated, useless*

2.0

```
cmsInt32Number  cmsGetTagCount(cmsHPROFILE hProfile);
```

!
P p p p h h p hk ,

Parameters:

hProfile: Handle to a profile object

Returns:

Number of tags on success, -1 on error.

2.0

```
cmsTagSignature  cmsGetTagSignature(cmsHPROFILE hProfile,
                                    cmsUInt32Number n);
```

!
P p h p k ckh hh h 7 ckh ck h , hp h h ck ck
h 7 ,

Parameters:

hProfile: Handle to a profile object
n: index to a tag position (0-based)

Returns:

The tag signature on success, 0 on error.

!
!
!
!
!
!

!
!

2.0

```
cmsBool   cmsIsTag(cmsHPROFILE hProfile, cmsTagSignature sig);
```

P p P h h h p h h ck p hk , k j h p hk h
h ,

!
Parameters:

hProfile: Handle to a profile object.
sig: Tag signature, as stated in k 95

Returns:

TRUE if the tag is found, FALSE otherwise.

2.0

```
void*   cmsReadTag(cmsHPROFILE hProfile,  cmsTagSignature sig);
```

!
P ck h h h h p h p h ckp p h p ı ck
p hk ı kckh p p h ,

h k hkkp p h ck h p p p kckh , h ck p p h
p p k *parsing* , p k p ckh
h 7 p k h kh p p p ck ck kk h kh h ,
p k p hk ckh p k h p hk , H h p p
p ck kh ck k p h h , h
h p P ck kck p ck , p h p ckh h p p
p ck p hk p hk h p h ph p p p , H
ckhh ck h p k h p h pp ck,

!
Parameters:

hProfile: Handle to a profile object.
sig: Tag signature, as stated in k 95

Returns:

A pointer to a profile-owned object holding tag contents, or NULL if the signature is not found. Type of object does vary. See k 95 *for a list of returned types.*

2.0

```
cmsBool  cmsWriteTag(cmsHPROFILE hProfile,
                      cmsTagSignature sig,
                      const void* data);
```

!

ph ı H p hk ck h kk p ph kh h , h ck ck ck
H p h p ck p h p hk ,

phh h k p ck ı h h p p p ck
ck *Little CMS* hkk ck kk ph kh h p , p ck p ck p
k h p kh 9 ck 1 H p h ckh p p p h
p p ,

Parameters:

hProfile: Handle to a profile object

sig: Tag signature, as stated in k 95

data: A pointer to an object holding tag contents. Type of object does vary. See k 95 *for a list of required types.*

Returns:

TRUE on success, FALSE on error

2.0

```
cmsBool  cmsLinkTag(cmsHPROFILE hProfile,
                     cmsTagSignature sig,
                     cmsTagSignature dest);
```

!

p ckp p p *sig* h k h *dest.* h h h
k h p 2 0 kck2 h 2 0 h 7 p k ph 4 0 (0 4 7 5 5774 7 5 150,1 j 9 h 0 p

2.1

```
cmsTagSignature  cmsTagLinkedTo(cmsHPROFILE hProfile, cmsTagSignature sig);
```

P p kh j ck h h p ph p p p h h
kh j ck p ,

Parameters:

hProfile: Handle to a profile object

sig: Signature of linking tag

Returns:

Signature of linked tag, or NULL if no tag is linked

Accessing tags as raw data

h kk p ck ph ckp k H p hk ck h jh
h , p k h h P h j ck ck pj ph h p ck
p ckh h j ck h ph kh h ck p k h pp p, H h hkk
ck ck p hk p p ckh j ck p h,

2.0

```
cmsInt32Number   cmsReadRawTag(cmsHPROFILE hProfile,
                               cmsTagSignature sig,
                               void* Buffer, cmsUInt32Number BufferSize);
```

!
h hk p P ck ckh p h h p , 5 p h ck
p hk kk p p , j h
h ck p ck7 p h , h p p p
ck ck h ph h ,

ckh p h h h h p ck , p h h p p ck h k
p ck p p p j p hk h h h , h k h p p kk

!
Parameters:

hProfile: Handle to a profile object

sig: Signature of tag to be read

Buffer: Points to a memory block to hold the result.

BufferSize: Size of memory buffer in bytes

Returns:

Number of bytes readed.

2.0

```
cmsBool    cmsWriteRawTag(cmsHPROFILE hProfile,
                          cmsTagSignature sig,
                          const void* data, cmsUInt32Number Size);
```

[illegible]

Parameters:

hProfile: Handle to a profile object

sig: Signature of tag to be written

Buffer: Points to a memory block holding the data.

BufferSize: Size of data in bytes

Returns:

TRUE on success, FALSE on error

Profile structures

H p hk h p k , ph k kck ck k p ckh h ck p ck

H p ďkh		
H 09 p	p p?	7 j 5 H 5305 9 H 5331
H	j h ?	k j h
H 09 p	p ?	7 j 5 12 7 7 12 9 7ďk ďk7
k 31 p	k p ?	7,,5,7
H 09 p	Hk h ?	

Table 26

H h h ďkh		
H	Hk h ?	p 79
H	pp ďk ?	h h p ph
H 09 p	Hk h ?	h h ďkh

Table 27

Platforms

k p k p h p	
h h	7 1527271
h h p	7 1 201321
h k ph	7 20221 24
h H	7 20141397 H
h kh	7 21141 21
h h	7 9 3 3345 (h

Table 28

Reference gamut

h p kP p ďk	7 47493ďk34 p

Table 29

Image State

p h k ph ph H H	
h k ph p h	7 40303 32
h p h	7 40354732
h k k k ph p h	7 33473032
h P k h pďk ph h k k ph p	7 49353 30 p
h P k h ph k ph p	7 49473 30 p

Table 30

Pipeline Stages (Multi processing elements)

h p	
h p k	7 30434041
h ph k	7 3 354133
h k	7 303 4241 k
h k	7 39151020
h k	7 32151020
! ph h !	
h 9 k	7 3 094597 k9
h 9 k	7 45093 97 9k

h ck k p k	7 3 303 97 k
h 9 1	7 09970197 9 1
h 1 9	7 01970997 1 9
h hck h k	7 33313 97 hck

Table 31

!

p k	
h p k p	7 47354933 p
h k ck p	7 40353 33
h ck p	7 30424933 p

Table 32

!
!

ckh P	p	
h	7 20413515	
h	7 20413512	
h H	7 20413513 H	
h	7 20413521	
h	7 2041351	
h	7 111 9797	
h	7 111 9727	
h	7 111 1 97	
h	7 111 1 27	

Table 33

k p h p k h p ck ck p ck p hk
ck ph h ,

!

k k h p	
h h h k p	1 7574727E! ((!
h hk p	1 7784747F! ((!
h P k h p	1 8384747F! ((!
h H j ph p	1 7: 7B7685! ((! !

h p k ph p	1 85887289! ((!
h k p p h ph p	1 7681797G! ((!
h k p h ph p	1 76848572! ((!
h kh h ph p	1 75848673! ((!
h p h p ph p	1 8381797G! ((!
h hk ph p	1 7781837F! ((!
h hčk h p	1 877:757E! ((!
h hčk p	1 877:7574! ((!
h p ı h k h h	1 817B8587! ((!
h P h k	1 54636531! (DS ! (!
h h k	1 615E5531! (QNE! (!
h h k	1 525E5531! (BNE! (!
h	1 5C615455! (LQDE(!
h H p	1 7:7E7884! ((!
h p p	1 78837287! ((!
h h p	1 7G777784! ((!
h hkj p	1 847:7D7C! ((!
h k p	1 777D7689! ((!
h h h p hk p	1 7E817784! ((!
h h h p hk P pčk p	1 7E817783! ((!
h h h k h h p p	1 757E8174! ((!
h h h k h p ı p	1 75747B81! ((!

Table 34

! !
!
!
!

! !

Formatters

p p p ꝯk ꝯk ph h p p p h ꝯk, p h kh ꝯk h ꝯk
H 09 p h h h kꝯk kk

```
!
!!!!!!!!!!   A O TTTTT U Y F P X S EEE CCCC BBB
!
!
```

k h h , h h k	ꝯh p h p k 53 h k p h
h h ꝯk p h	h h h kp ꝯk p p h k5 h k *internal use only*
h k	k k
k p 7 h H k j	k 5 h H h hk
k pl 7	j 5 k p
53	ꝯh !
! h	P
p	k
k	k p h k
p	k
hp	k P P ꝯk

Table 35

!

Macros to build formatters

```
     ! GMPB   TI )  *!!!!!!!!!!!! ))  *! ==! 33*!
     ! PQ   N   FE TI )  *!!!!!!!! ))  *! ==! 32*!
     ! DPMPSTQBDF  TI )  *!!!!!!! ))  *! ==! 27*!
     ! T BQG ST   TI )  *!!!!!!!! ))  *! ==! 25*!
     ! GMB  PS  TI )  *!!!!!!!!!!! ))  *! ==! 24*!
     ! QMBOBS  TI )  *!!!!!!!!!!! ))  *! ==! 23*!
     ! FOE BO27  TI )  *!!!!!!!!! ))  *! ==! 22*!
     ! EPT BQ  TI )  *!!!!!!!!!!! ))  *! ==! 21*!
     ! F   SB  TI )  *!!!!!!!!!!!! ))  *! ==! 8*!
     ! DIBOOFMT  TI )  *!!!!!!!!! ))  *! ==! 4*!
     ! C   FT  TI )  *!!!!!!!!!!!! )  *!
!
```

Macros to extract information from formatters

```
     !   GMPB  )  *!!!!!!!!!!!! )))  *   33*'2*!
     !   PQ   N   FE)  *!!!!!!!! )))  *   32*'2*!
     !   DPMPSTQBDF)  *!!!!!!! )))  *   27*'42*!
     !   T BQG ST  )  *!!!!!!!! )))  *   25*'2*!
     !   GMB  PS)  *!!!!!!!!!!! )))  *   24*'2*!
     !   QMBOBS)  *!!!!!!!!!!! )))  *   23*'2*!
     !   FOE BO27)  *!!!!!!!!! )))  *   22*'2*!
     !   EPT BQ)  *!!!!!!!!!!! )))  *   21*'2*!
     !   F   SB)  *!!!!!!!!!!!! )))  *   8*'8*!
     !   DIBOOFMT)  *!!!!!!!!! )))  *   4*'26*!
     !   C   FT)  *!!!!!!!!!!!! ))  *'8*!
!
```

Color spaces in *Little CMS* notation

ďkh p p ďk k j k p , k p h khh p ,

h k		
	7	j k p
(p p ďķ	5	P ďk p p kh h hk k
(p p ďķ	9	P p ďk p p kh h k
P	0	p k
P	1	P ďk p k
	2	kk
	3	kk k
p	4	p
	5	
	3	H
	57	H ((
	55	
	59	
	50	
	51	
5	52	5 hh ďk k
9	53	9 hh ďk k
0	54	0 hh ďk k
1	55	1 hh ďk k
2	53	2 hh ďk k
3	97	3 hh ďk k
4	95	4 hh ďk k
5	99	5 hh ďk k
3	90	3 hh ďk k
57	91	57 hh ďk k
55	92	55 hh ďk k
59	93	59 hh ďk k
50	94	50 hh ďk k
51	95	51 hh ďk k
52	93	52 hh ďk k
9	07	Hďk h k h 9 kďk ďkh

Table 36

!

! !

Translate color space from/to *Little CMS* notation to ICC

h h p H k p p h *cmsColorSpaceSignature, Table 10* *Little CMS* h p ƈkh p p *(Table 36)* .

2.0

```
cmsColorSpaceSignature _cmsICCcolorSpace(int OurNotation);
```

p p *Little CMS* k p h *(Table 36)* H k p h *Table 10* ,

Parameters:

OurNotation: any value from Table 36

Returns:

Corresponding cmsColorSpaceSignature (Table 10) or -1 on error.

2.0

```
int _cmsLCMScolorSpace(cmsColorSpaceSignature ProfileSpace);
```

p p H k p h *Table 10* *Little CMS* k p h *(Table 36)*,

Parameters:

ProfileSpace: any cmsColorSpaceSignature from Table 10

Returns:

Corresponding Little CMS value (Table 36) or -1 on error.

Predefined formatters

kp ck ck h ckh ***lcms2.h*** p phck hh ph kp p p h h , p p p ck h ck h p kh ck ,

!

P 5	p k 5 h
P 5 P	p k 5 h p p ck
P 53	p k 53 h
P 53 P	p k 53 h p p ck
P 53	p k 53 h ck ckh
P 5	p k H p ck k 5 h
P 53	p k H p ck k 53 h
P 53	p k H p ck k 53 h ck ckh
P 5 P	p k 5 h h k k
P 53 P	p k 53 h h k k
P 5	P 5 h
P 5 P	P 5 h p ck h k
P 5	P 5 h
P 5 P	P 5 h p ck h k
P 53	P 53 h
P 53 P	P 53 h p ck h k
P 53	P 53 h ck ckh
P 53	P 53 h
P 53 P	P 53 h p ck h k
P 53	P 53 h h ck ckh
P 5	P 5 h k k k h h h p ck
P 5 P	P 5 h p ck h k
P 53	P 53 h k k k h h h p ck
P 53 P	P 53 h p ck h k
P 53	P 53 h h ck ckh
P 5	h p ck k k k P h 5 h
P 5 P	
P 53	h p ck k k k P h 53 h
P 5	h p ck k k k P h 5 h
P 5 P	h p ck k k k P h p 5 h k
P 53	h p ck k k k P h 53 h
P 53 P	h p ck k k k P h p 53 h k
P 53	
P 5	
P 5 P	
P 53	
P 53	
5	
5 P	
53	
53 P	

53	
5	
5	
5 P	
5	
5 P	
53	
53 P	
53	
53 P	
53	
5	
53	
53	
5	
5 P	
53	
53 P	
53	
2 5	
2 53	
2 53	
2 5	
2 53	
2 53	
5	
5 P	
53	
53 P	
53	
4 5	
4 53	
4 53	
4 5	
4 53	
4 53	
5 5	
5 53	
5 53	
5 5	
5 53	
5 53	
3 5	
3 53	
3 53	
3 5	

3 53	
3 53	
57 5	
57 53	
57 53	
57 5	
57 53	
57 53	
55 5	
55 53	
55 53	
55 5	
55 53	
55 53	
59 5	
59 53	
59 53	
59 5	
59 53	
59 53	
53	
5	
5	
53	
53	
p 5	
p 5 P	
p 53	
p 53 P	
p 53	
5	
5 P	
53	
53 P	
53	
5	
5 P	
53	
53 P	
53	
5	
5 P	
53	
53 P	
53	
k h h	

P	
P	
P	
P	
9 5	
9 5	
9 53	
j ckp k k h h p	
P	
P	
P	
P	
P	

Table 37

Illuminant types!	
H H	7 7777777
H H 27	7 7777775
H H 32	7 7777779
H H 30	7 7777770
H H 9	7 7777771
H H 22	7 7777772
H H	7 7777773
H H	7 7777774
H H 5	7 7777775

Table 38

!

p	
p (	p?
p (	?
(	?

Table 39

Intents

!

H H	
H P	7
H P H PH PH	5
H P H	9
H PH PH	0

Table 40

! !

H h	
H P P P	57
H P P P H PH PH	55
H P P P H	59
H P P P	50
H P P P H PH PH	51
H P P P H	52

Table 41

!

! 2.0!

```
cmsUInt32Number  cmsGetSupportedIntents(cmsUInt32Number nMax,
                                        cmsUInt32Number* Codes,
                                        char** Descriptions);
```

hkk k h hck p ckck ph h p kk p ckh , *Little CMS* k h p h p kk h k p ck h ckh ? h h h ck ck h kh , kk h p p h

Parameters:

nMax: Max array elements to fill.

Codes [] : Pointer to user-allocated array of cmsUInt32Number to hold the intent id-numbers.

Descriptions []: Pointer to a user allocated array of char to hold the intent names.*

Returns:

Supported intents count.

!

2.6!

```
cmsUInt32Number  cmsGetSupportedIntentsTHR(cmsContext ContextID,
                                           cmsUInt32Number nMax,
                                           cmsUInt32Number* Codes,
                                           char** Descriptions);
```

hkk k h hck p ckck ph h p kk p ckh , *Little CMS* k h p h p kk h k pck h ckh ? h h h ckck h kh , kk h p p h

Parameters:

ContextID: Handle to user-defined context, or NULL for the global context

nMax: Max array elements to fill.

Codes [] : Pointer to user-allocated array of cmsUInt32Number to hold the intent id-numbers.

Descriptions []: Pointer to a user allocated array of char to hold the intent names.*

Returns:

Supported intent count.

!

2.0!

```
cmsUInt32Number    cmsGetHeaderRenderingIntent(cmsHPROFILE hProfile);
```

!

p hk ckpp ckph h , p H '*The rendering intent field shall specify the rendering intent which should be used (or, in the case of a Devicelink profile, was used) when this profile is (was) combined with another profile. In a sequence of more than two profiles, it applies to the combination of this profile and the next profile in the sequence and not to the entire sequence. Typically, the user or application will set the rendering intent dynamically at runtime or embedding time. Therefore, this flag may not have any meaning until the profile is used in some context, e.g. in a Devicelink or an embedded source profile.”*

!

Parameters:

hProfile: Handle to a profile object

Returns:

A H 09 p*holding the intent code, as described in* H *section.*

2.0!

```
void   cmsSetHeaderRenderingIntent(cmsHPROFILE hProfile,
                                   cmsUInt32Number RenderingIntent);
```

!

p hk ck pp ck ph h , ckh h ,

Parameters:

hProfile: Handle to a profile object
RenderingIntent: A H 09 pholding the intent code, as described in H
section.

Returns:

None!

!

2.1!

```
cmsBool   cmsIsIntentSupported(cmsHPROFILE hProfile,
                               cmsUInt32Number Intent,
                               cmsUInt32Number UsedDirection);
```

P p P h p ckh h h k ckh h ckp h , *Little CMS*
kk j p kk h p ck ph h p h p p
h p kk h ck ph p ckh h h k ck,

!

Parameters:

hProfile: Handle to a profile object
Intent: A H 09 *pholding the intent code, as described in* H *section.*
UsedDirection: any of those constants:

```
        ! MDNT  TFE BT  OQ  !!!!!! 1!
        ! MDNT  TFE BT P  Q  !!!!! 2!
        ! MDNT  TFE BT QSPPG!!!!!! 3!
```

!

Returns:

TRUE if the intent is implemented, FALSE otherwise.

Flags

ck k p , p h k ı h ck h ’ p p p,

	7 7717 H h h 5 h k
H H	7 7577 H h h h h h
P P	7 7977 p p
p h k	
	7 5777 k p
P H	7 1777 p h
h	
H H	7 9777
H H H	7 7771 h ck
H P P	7 7177 p p h p p , kh p
P P	7 7577 k p h h h p
pck h kh j p h	
5 H H H	7 7775 p 5 h ck h kh j
H	7 7797 ck h k p p p 9ck h kh j
	7 7757 p hk p ck h kh j p h
hh p h k p h h h	
P	7 7779 p h h h
H PH H	7 7775 p kh ph h k h h k
P H PH H	7 7757 p p kh ph h k h h k
ck ck ck p k	
H	7 5777 // Prevent negative numbers in floating // point transforms
h p k p p phck h	
PH H	7 == 53
P h k	
P P	7 75777777

Table 42

Color transforms

2.0!

```
cmsHTRANSFORM  cmsCreateTransform(cmsHPROFILE Input,
                                  cmsUInt32Number InputFormat,
                                  cmsHPROFILE Output,
                                  cmsUInt32Number OutputFormat,
                                  cmsUInt32Number Intent,
                                  cmsUInt32Number dwFlags);
```

!

p k p p p p p k h h ,

Parameters:

Input: Handle to a profile object capable to work in input direction
InputFormat: A bit-field format specifier as described in Formatters section.
Output: Handle to a profile object capable to work in output direction
OutputFormat: A bit-field format specifier as described in Formatters section.
Intent: A H 09 p*holding the intent code, as described in* H *section.*
dwFlags: A combination of bit-field constants described in k 19.

Returns:

A handle to a transform object on success, NULL on error.

2.0!

```
cmsHTRANSFORM  cmsCreateTransformTHR(cmsContext ContextID,
                                  cmsHPROFILE Input,
                                  cmsUInt32Number InputFormat,
                                  cmsHPROFILE Output,
                                  cmsUInt32Number OutputFormat,
                                  cmsUInt32Number Intent,
                                  cmsUInt32Number dwFlags);
```

!

ph p kk h H ck p ,

!

Parameters:

ContextID: Pointer to a user-defined context cargo.
Input: Handle to a profile object capable to work in input direction
Output: Handle to a profile object capable to work in output direction
InputFormat: A bit-field format specifier as described in Formatters section.
OutputFormat: A bit-field format specifier as described in Formatters section.
Intent: A H 09 p*holding the intent code, as described in* H *section.*
dwFlags: A combination of bit-field constants described in k 19.

Returns:

A handle to a transform object on success, NULL on error.

2.0!

```
void  cmsDeleteTransform(cmsHTRANSFORM hTransform);
```

!

k p p ƈk ƈk p h ƈk p , h h ƈk p
p hk ƈk p p p ,

!

Parameters:

*hTransform**:** Handle to a color transform object.*

Returns:

None

2.0!

```
void   cmsDoTransform(cmsHTRANSFORM hTransform,
                                     const void * InputBuffer,
                                     void * OutputBuffer,
                                     cmsUInt32Number Size);
```

!

h h p k h pclh p p p h k p p p ,

Parameters:

*hTransform**:** Handle to a color transform object.*
InputBuffer: A pointer to the input bitmap
OutputBuffer: A pointer to the output bitmap.
Size: the number of PIXELS to be transformed.

Returns:

None

2.4 DEPRECATED!

```
void   cmsDoTransformStride(cmsHTRANSFORM hTransform,
                                     const void * InputBuffer,
                                     void * OutputBuffer,
                                     cmsUInt32Number Size,  cmsUInt32Number Stride);
```

!

Deprecated. Use cmsDoTransformLineStride instead.

h h p k h pclh p p p h k p p p ,
k p p h clk p p p phclk hh p h k h
clh p p h k p p , h klh h h h h
p k p clk p p p h h k p clh p k p p,
p h h h hclk h k ***cmDoTransform***

Parameters:

*hTransform**:** Handle to a color transform object.*
InputBuffer: A pointer to the input bitmap
OutputBuffer: A pointer to the output bitmap.
Size: the number of PIXELS to be transformed.
Stride: Plane separation on planar formats

Returns:

None

2.8!

```
void  cmsDoTransformLineStride(cmsHTRANSFORM  Transform,
                               const void* InputBuffer,
                                void* OutputBuffer,
                                cmsUInt32Number PixelsPerLine,
                                cmsUInt32Number LineCount,
                                cmsUInt32Number BytesPerLineIn,
                                cmsUInt32Number BytesPerLineOut,
                                cmsUInt32Number BytesPerPlaneIn,
                                cmsUInt32Number BytesPerPlaneOut
```

!
h h p k h h k p h h , h h p k
kh ck p ckckh , ckh p kh h
p h)H [, H k p p kh kck p k k k
ckckh , ckckh kh ck k kck p kk h ,H , kkkh h h
ckck ckh , h h p h h p ck kk
p p h kk h ck k h p h ck

Parameters:

*hTransform**:** Handle to a color transform object.*

InputBuffer: A pointer to the input bitmap

OutputBuffer: A pointer to the output bitmap.

PixelsPerLine: The number of pixels for line, which is same on input and in output.

LineCount: The number of lines, which is same on input and output

BytesPerLine{In,Out}: ckh h p kh .

p k H *{In,Out}:* ckh h p k *inside a line. Only applies in planar formats.*

Returns:

None

Proofing transforms

p h p p ck k k p kck p h p p ck p ck
hh ck h , h p k h p h p p H hkk j
kh k ck p h p ck p ck ph p, h ph p p hk ck h k ck
p p h h h khj k k p hkk k j ph h k, h p h
p p h ck h p ph h , h h h p p
p h k p h h p kk k p h h k ckh h ,

2.0!

```
cmsHTRANSFORM  cmsCreateProofingTransform(cmsHPROFILE Input,
                                          cmsUInt32Number InputFormat,
                                          cmsHPROFILE Output,
                                          cmsUInt32Number OutputFormat,
                                          cmsHPROFILE Proofing,
                                          cmsUInt32Number Intent,
                                          cmsUInt32Number ProofingIntent,
                                          cmsUInt32Number dwFlags);
```

!

p p p h k ckh p h , h ck p p k
ck h ck ph ck p h p hk , k p h h kp k h p ck ph
h k ckh , k p h ck j ck h k ck kk h k

cmsFLAGS_GAMUTCHECK k p p k ck h ck k pck h ck
h *cmsSetAlarmCodes*

cmsFLAGS_SOFTPROOFING ck k p h ck h ,

!
!

Parameters:

Input: Handle to a profile object capable to work in input direction
Output: Handle to a profile object capable to work in output direction
InputFormat: A bit-field format specifier as described in Formatters section.
OutputFormat: A bit-field format specifier as described in Formatters section.
Intent: A H 09 p*holding the intent code, as described in* H *section.*
ProofingIntent: A H 09 p*holding the intent code, as described in* H
section.
dwFlags: A combination of bit-field constants described in k 19.

Returns:

A handle to a transform object on success, NULL on error.

!

2.0!

```
cmsHTRANSFORM  cmsCreateProofingTransformTHR(cmsContext ContextID,
                                   cmsHPROFILE Input,
                                   cmsUInt32Number InputFormat,
                                   cmsHPROFILE Output,
                                   cmsUInt32Number OutputFormat,
                                   cmsHPROFILE Proofing,
                                   cmsUInt32Number Intent,
                                   cmsUInt32Number ProofingIntent,
                                   cmsUInt32Number dwFlags);
```

!

ꝑh p kk h H ꝺk p ,

!

Parameters:

ContextID: Pointer to a user-defined context cargo.

Input: Handle to a profile object capable to work in input direction

Output: Handle to a profile object capable to work in output direction

InputFormat: A bit-field format specifier as described in Formatters section.

OutputFormat: A bit-field format specifier as described in Formatters section.

Intent: A H 09 p*holding the intent code, as described in* H *section.*

ProofingIntent: A H 09 p*holding the intent code, as described in* H *section.*

dwFlags: A combination of bit-field constants described in k 19.

Returns:

A handle to a transform object on success, NULL on error.!

!

!

2.0!

```
void   cmsSetAlarmCodes(cmsUInt16Number AlarmCodes[cmsMAXCHANNELS]);
```

!

k k ꝺk ꝺk pj p h p p , k p ꝺk ꝺkh 53 h ,

Parameters:

AlarmCodes: Array [16] of codes. ***ALL 16 VALUES MUST BE SPECIFIED****, set to zero unused channels.*

Returns:

None

2.0

```
void   cmsGetAlarmCodes(cmsUInt16Number AlarmCodes[cmsMAXCHANNELS]);
```

pp k k ck ck pj p h p p , k p ck ckh 53 h ,

Parameters:

AlarmCodes: Array [16] of codes. ***ALL 16 VALUES WILL BE OVERWRITTEN****.*

Returns:

None

2.6

```
void   cmsSetAlarmCodesTHR(cmsContext ContextID,
                       const cmsUInt16Number AlarmCodes[cmsMAXCHANNELS]);
```

ck ck pj p h p p p h , k p ck ckh 53 h ,

Parameters:

ContextID: Handle to user-defined context, or NULL for the global alarm codes

AlarmCodes: Array [16] of codes. ***ALL 16 VALUES MUST BE SPECIFIED****, set to zero unused channels.*

Returns:

None

2.6

```
void   cmsGetAlarmCodesTHR(cmsContext ContextID,
                             cmsUInt16Number AlarmCodes[cmsMAXCHANNELS]);
```

pp ck ck pj p h p p p h , k p ck ckh 53 h ,

Parameters:

ContextID: Handle to user-defined context, or NULL for the global context

AlarmCodes: Array [16] of codes. ***ALL 16 VALUES WILL BE OVERWRITTEN****.*

Returns: ((

!

2.0!

```
cmsFloat64Number   cmsSetAdaptationState(cmsFloat64Number d);
```

!

ɗk h p k k ph ph h kk *cmsCreateExtendedTransform*, *Little CMS* ɗkk h k ɗk h ,

Parameters:

d: Degree on adaptation 0=Not adapted, 1=Complete adaptation, in-between=Partial adaptation. Use negative values to return the global state without changing it.

Returns:

Previous global adaptation state.

!

2.6!

```
cmsFloat64Number   cmsSetAdaptationStateTHR(cmsContext ContextID,
                                             cmsFloat64Number d);
```

!

ɗk h p k k ph ph h h h , ɗk h kh kk *cmsCreateExtendedTransformTHR()*, *Little CMS* ɗkk h k ɗk h ,

Parameters:

ContextID: Handle to user-defined context, or NULL for the global context

d: Degree on adaptation 0=Not adapted, 1=Complete adaptation, in-between=Partial adaptation. Use negative values to return the global state without changing it.

Returns:

Previous global adaptation state.

! !

Multiprofile transforms

p h pp ck p hk , p p ck k p p p ck k kk p hk h h k ck h kh j, k p hp ck h h h h p ck,

2.0!

```
cmsHTRANSFORM  cmsCreateMultiprofileTransform(cmsHPROFILE hProfiles[],
                        cmsUInt32Number nProfiles,
                        cmsUInt32Number InputFormat,
                        cmsUInt32Number OutputFormat,
                        cmsUInt32Number Intent,
                        cmsUInt32Number dwFlags);
```

!

Parameters:

hProfiles[] : Array of handles to open profile objects.
nProfiles: Number of profiles in the array.
InputFormat: A bit-field format specifier as described in Formatters section.
OutputFormat: A bit-field format specifier as described in Formatters section.
Intent: A H 09 p*holding the intent code, as described in* H *section.*
dwFlags: A combination of bit-field constants described in k 19.

Returns:

A handle to a transform object on success, NULL on error.

2.0!

```
cmsHTRANSFORM  cmsCreateMultiprofileTransformTHR(cmsContext ContextID,
                        cmsHPROFILE hProfiles[],
                        cmsUInt32Number nProfiles,
                        cmsUInt32Number InputFormat,
                        cmsUInt32Number OutputFormat,
                        cmsUInt32Number Intent,
                        cmsUInt32Number dwFlags);
```

ph p kk h H ck p ,

!

Parameters:

ContextID: Pointer to a user-defined context cargo.
hProfiles[] : Array of handles to open profile objects.
nProfiles: Number of profiles in the array.
InputFormat: A bit-field format specifier as described in Formatters section.
OutputFormat: A bit-field format specifier as described in Formatters section.

Intent: A H 09 p*holding the intent code, as described in* H *section.*
dwFlags: A combination of bit-field constants described in k 19.

Returns:

A handle to a transform object on success, NULL on error.

!
!

2.0!

```
cmsHTRANSFORM  cmsCreateExtendedTransform(cmsContext ContextID,
                         cmsUInt32Number nProfiles, cmsHPROFILE hProfiles[],
                         cmsBool  BPC[],
                         cmsUInt32Number Intents[],
                         cmsFloat64Number AdaptationStates[],
                         cmsHPROFILE hGamutProfile,
                         cmsUInt32Number nGamutPCSposition,
                         cmsUInt32Number InputFormat,
                         cmsUInt32Number OutputFormat,
                         cmsUInt32Number dwFlags);
```

!

ćk ćk p kh p hk k p p p p h h kk p p p p hk h
h , kk p p p p h h p p p h kk,

!

Parameters:

ContextID: Pointer to a user-defined context cargo.

hProfiles[] : Array of handles to open profile objects.

nProfiles: Number of profiles in the array.

BPC [] : Array of black point compensation states

hGamutProfile: Handle to a profile holding gamut information for gamut check. Only used if cmsFLAGS_GAMUTCHECK specified. Set to NULL for no gamut check.

nGamutPCSPosition: Position in the chain of Lab/XYZ PCS to check against gamut profile Only used if cmsFLAGS_GAMUTCHECK specified.

InputFormat: A bit-field format specifier as described in Formatters section.

OutputFormat: A bit-field format specifier as described in Formatters section.

H *cmsUInt32Number holding the intent code, as described in Intents section.*

dwFlags: A combination of bit-field constants described in Table 42.

Returns:

A handle to a transform object on success, NULL on error.

Dynamically changing the input/output formats

kk p p ck p p k pj p p p ck
ph h kk h k 53 h p h h ,

2.2!

```
cmsUInt32Number   cmsGetTransformInputFormat(cmsHTRANSFORM hTransform);
```

!
P p h p h ck h h p p ,

!
Parameters:

hTransform: Handle to a color transform object.

Returns:

The input format associated with the given transform or 0 if NULL parameter

!

2.2!

```
cmsUInt32Number  cmsGetTransformOutputFormat(
                                        cmsHTRANSFORM hTransform);
```

!
P p p h ck h h p p ,

!
Parameters:

hTransform: Handle to a color transform object.

Returns:

The output format associated with the given transform or 0 if NULL parameter

!
!
!
!
!
!
!
!
!
!
!

2.1!

```
cmsBool   cmsChangeBuffersFormat(cmsHTRANSFORM hTransform,
                                  cmsUInt32Number InputFormat,
                                 cmsUInt32Number OutputFormat);
```

h h ck ckh p h h h p p , kk p p
ck p p k pj p p p ck ph h kk h
k 53 h p hh , h h h p hck ck p j pck h hkh ck kck
hck ck p h k h p p p h h h ,

Parameters:

Transform: Handle to a color transform object.

InputFormat: A bit-field format specifier as described in Formatters section.

OutputFormat: A bit-field format specifier as described in Formatters section.

Returns:

TRUE on success FALSE on error.

!

PostScript generation

ck kh h ph h ck p h p p h h h ck hp k ck k k p ph h p p p, *Little CMS* ck p hck h p k h ck p hk h k p pp ck k p P ck ph h h ph P ,

- P p h k ph p p hk , k ck ckh ph p p ck p ck p p ,
- p h k h ck pj p hk ck p h ck ck h k ck ckh ck ck h hh ,

h k p k ph ck h j h ck kk h h k 7 ck p k , p ck kk p h k k j ,

h kh j p hk p p ck k h k p p p k p p P ,

WARNING p h h ph h kh h ck 5 h p k , H p k p p ck P p k p p hkk h p p ,

2.0

```
cmsUInt32Number  cmsGetPostScriptColorResource(cmsContext ContextID,
                                               cmsPSResourceType Type,
                                               cmsHPROFILE hProfile,
                                               cmsUInt32Number Intent,
                                               cmsUInt32Number dwFlags,
                                               cmsIOHANDLER* io);
```

Little CMS 2 hh ck ck p ph k pp p , ph kh h h p p ck h h ckk p ı ,

Parameters:

ContextID: Pointer to a user-defined context cargo.

*Type: Either **cmsPS_RESOURCE_CSA** or **cmsPS_RESOURCE_CRD***

hProfile: Handle to a profile object

Intent: A H 09 p*holding the intent code, as described in* H *section.*

dwFlags: A combination of bit-field constants described in k 19.

Iohandler: Pointer to a serialization object.

Returns:

The resource size in bytes on success, 0 en error.

!

2.0!

```
cmsUInt32Number  cmsGetPostScriptCSA(cmsContext ContextID,
                                     cmsHPROFILE hProfile,
                                     cmsUInt32Number Intent,
                                     cmsUInt32Number dwFlags,
                                     void* Buffer,  cmsUInt32Number dwBufferLen);
```

!

p p *cmsGetPostScriptColorResource* h kh p h ,

Parameters:

ContextID: Pointer to a user-defined context cargo.
hProfile: Handle to a profile object
Intent: A H 09 p*holding the intent code, as described in* H *section.*
dwFlags: A combination of bit-field constants described in k 19.
Buffer: Pointer to a user-allocated memory block or NULL. If specified, It should be big enough to hold the generated resource.
dwBufferLen: Length of Buffer in bytes.

Returns:

The resource size in bytes on success, 0 en error.

!

2.0!

```
cmsUInt32Number  cmsGetPostScriptCRD(cmsContext ContextID,
                                     cmsHPROFILE hProfile,
                                     cmsUInt32Number Intent,
                                     cmsUInt32Number dwFlags,
                                     void* Buffer,  cmsUInt32Number dwBufferLen);
```

p p *cmsGetPostScriptColorResource* h kh P p h ,

Parameters:

ContextID: Pointer to a user-defined context cargo.
hProfile: Handle to a profile object
Intent: A H 09 p*holding the intent code, as described in* H *section.*
dwFlags: A combination of bit-field constants described in k 19.
Buffer: Pointer to a user-allocated memory block or NULL. If specified, It should be big enough to hold the generated resource.
dwBufferLen: Length of Buffer in bytes.

Returns:

The resource size in bytes on success, 0 en error.

Δ E metrics

!
ck ck k h k p pkck p p
p k , h h k p h h k ck p ck hp k p
k j p k h h , k ck h h k p p p ckh
k p , hck h ck 5,7 h kk k p ckh p
, ck k 5,7 h h p h k ckh ck p ck p p 5,7 h
h k , p k h h k , k p ckh p p p 5 p p k
k h k , k ck k p ckh p kk
ck k k j khj ckh p ck p p p k p
h kk ck , H p k ck p ck k kck
pp p , p kk h p ? kck p p
! k p ck p

- p h ph p p p ph h k
- ck h ckph ck
- h h p h k p p hk p ph h p p h
- p ı h h h k

h ck ckh p k p h p kckh p
,

2.0

```
cmsFloat64Number  cmsDeltaE(const cmsCIELab* Lab1,  const cmsCIELab* Lab2);
```

!
(((k p ck h ckh 5343 ck h ck k 5343 ck 43 h
h , H h h h ph k p h h 0 ck 43 ck ph
p h ck ph ck kk h k ckp h kck kk ph , H p k
j ı k h ck k p p k p pph ck 43 , H h k j ck ckck
, p k h ck 43 h h k h p kk h p h p p ck
h ck ck, ckh p h k k p h h ckh p k p p h
ck 43 p, p k k p h h p k h ckh p ck 43
k , p h h h h h ckh p p ck h kk
kh ckck 43 ck j h h ,

Parameters:

Lab1: Pointer to first cmsCIELab value as described in Table 13

Lab2: Pointer to second cmsCIELab value as described in Table 13

Returns:

The dE76 metric value.

2.0

```
cmsFloat64Number  cmsCMCdeltaE(const cmsCIELab* Lab1,
                               const cmsCIELab* Lab2,
                               cmsFloat64Number  l,  cmsFloat64Number  c);
```

H 5351 k p p h h ck k ph p
ph h ck k ck ck ck ck h ck p , H ck ck p hk
h ck p k kk h kh k ck p p , h p
hh p ck k p h p k h 9 5 kk h p 9 ckh p h kh
p p , p h k p h k p h kk p kk p h
h k p p h pckh p p hp , 5,7 ck k
k =5,7 h k ,

k h ck h ck ck h 32 ck H k p p p, k ck
k p k p 9 5 p hkh ck5 5 p p kck h p h hkh ,

Parameters:

Lab1: Pointer to first cmsCIELab value as described in Table 13

Lab2: Pointer to second cmsCIELab value as described in Table 13

Returns:

The dE CMC metric value.

2.0

```
cmsFloat64Number  cmsBFDdeltaE(const cmsCIELab* Lab1, const cmsCIELab* Lab2);
```

ck k ph ,

Parameters:

Lab1: Pointer to first cmsCIELab value as described in Table 13

Lab2: Pointer to second cmsCIELab value as described in Table 13

Returns:

The dE BFD metric value.

2.0

```
cmsFloat64Number  cmsCIE94DeltaE(const cmsCIELab* Lab1,
                                 const cmsCIELab* Lab2);
```

!

h k h H 5 93 kh ck h h 5332 kk ck H 31 ,
h h h hk p h h h p k p k ck PH k p
ck ck ph ck p h h ph p k p p , H k
p h k k ckk*L* kh ck*Kc* p ck p h k p *cf* ck
p h p ck p ck p p h h h *Little CMS* ,

Parameters:

Lab1: Pointer to first cmsCIELab value as described in Table 13
Lab2: Pointer to second cmsCIELab value as described in Table 13

Returns:

The CIE94 dE metric value.

!

2.0

```
cmsFloat64Number  cmsCIE2000DeltaE(const cmsCIELab* Lab1,
                                   const cmsCIELab* Lab2,
                                   cmsFloat64Number Kl,
                                   cmsFloat64Number Kc,
                                   cmsFloat64Number Kh);
```

!

k 9777 h hp ı p p h h ck 31 h , khj ck 31 h (
pp k p k p h ck ckh p h kh ck 9777 ph h h (
ck ckh p h kh p k p kk , ck 9777 h hkk ck p hck p h ck
ck hck k p ckh p h p kh h ,

Parameters:

Lab1: Pointer to first cmsCIELab value as described in Table 13
Lab2: Pointer to second cmsCIELab value as described in Table 13

Returns:

The CIE2000 dE metric value.

Temperature <-> Chromaticity (Black body)

k p p p h p ph h h h k kh h p kh h , k p
p p kh p h ck p h ck ph h p h h h hck k
k j ck p ckh p, p p kk p ck h j k h h p k p
p p h ck k j ck p ckh p k p kh p p
k j ck p , h p k p p p 2 777 p p p k k h h k p ck
k p k p p p 9 477\ 0 777 p kk h h p p ck k p ,

2.0

```
cmsBool cmsWhitePointFromTemp(cmsCIExyY* WhitePoint,
                              cmsFloat64Number  TempK);
```

pp k k j ck p h h p h p p h , khckp h 1777 92777 ,

!

Parameters:

WhitePoint: Pointer to a user-allocated cmsCIExyY variable to receive the resulting chromaticity.
TempK: Temperature in ºK

Returns:

TRUE on success, FALSE on error.

2.0

```
cmsBool cmsTempFromWhitePoint(cmsFloat64Number* TempK,
                              const cmsCIExyY* WhitePoint);
```

!

pp k k j ck p p h p h p h h ,

Parameters:

TempK: Pointer to a user-allocated cmsFloat64Number variable to receive the resulting temperature.
WhitePoint: Target chromaticity in cmsCIExyY

Returns:

TRUE on success, FALSE on error.

CIE CAM02

!

h h ck h , k p ck k h h ck h ck H h h ck h h H h H h h ck h j ck p hck , p k ck k h pp ck ck k h h kk k p 79 h h ck h ,

h h ck h	
H	h h ?
k 31 p	?
k 31 p	?
h	pp ck
k 31 p	k ?

Table 43

pp ck	
PP	5
H PP	9
P PP	0
PP	1

Table 44

E DBMD MB F! ! ! ! ! ! ! !). 2 *!

!

2.0!

```
cmsHANDLE cmsCIECAM02Init(cmsContext ContextID,
                          const cmsViewingConditions* pVC);
```

p 79 ı ck h h h ck h , ı ck k p p ck k ck k ck h p pck ck p p ckp h , h h ck h p p h ck hk ckh k 10, pp ck p k p ckh k 11, p p h ck h *d* hh ckh 7,,,5,7 p p ck k h p ck k k h h 5 ,

Parameters:

ContextID: Pointer to a user-defined context cargo.

pVC: Pointer to a structure holding viewing conditions (Table 44)

Returns:

Handle to CAM02 object or NULL on error.

2.0!

```
void cmsCIECAM02Done(cmsHANDLE hModel);
```

!

p h 79 ı p h kkh k ckp p ,

Parameters:

hModel: Handle to a CAM02 object

Returns:

None

2.0!

```
void cmsCIECAM02Forward(cmsHANDLE hModel,
                        const cmsCIEXYZ* pIn,
                        cmsJCh* pOut);
```

k 79 ck kh p pckckp h →

Parameters:

hModel: Handle to a CAM02 object

pIn: Points to the input XYZ value

pOut: Points to the output JCh value

Returns:

None!

2.0

```
void cmsCIECAM02Reverse(cmsHANDLE hModel,
                        const cmsJCh* pIn,
                        cmsCIEXYZ* pOut);
```

!

k 79 ck kh p p ckp h →

Parameters:

hModel: Handle to a CAM02 object

pIn: Points to the input JCh value

pOut: Points to the output XYZ value

Returns:

None

Gamut boundary description

!

ďk p ďk ph h h p h h ďķ j p kk h h k ph ,

2.0

```
cmsHANDLE  cmsGBDAlloc(cmsContext ContextID);
```

!

kk ďk p ďk ph p h j h ,

!

Parameters:

ContextID: Pointer to a user-defined context cargo.

Returns:

ďkk ďk p ďk ph p pp p,

2.0

```
void  cmsGBDFree(cmsHANDLE hGBD);
```

!

p ďk p ďk ph p ďk h ďkp p ,

Parameters:

hGBD: Handle to a gamut boundary descriptor.

Returns:

None

2.0

```
cmsBool  cmsGDBAddPoint(cmsHANDLE hGBD, const cmsCIELab* Lab);
```

!

ďkďk k h p h ďk p ďk ph p, h h kk ďk h j h , p p pp p p ďk ďkďkh h , ďk p ďk ph p j ďk kk *cmsGDBCompute* h kk ďķ

Parameters:

hGBD: Handle to a gamut boundary descriptor.

Lab: Pointer to a cmsCIELab value as described in Table 13

Returns:

TRUE on success, FALSE on error.!

2.0

```
cmsBool cmsGDBCompute(cmsHANDLE hGDB, cmsUInt32Number dwFlags);
```

ďk p ďk ph p h kk j h ďkh p k h h h
p , kk h h p ďkďkh kk j h h ďkďk h ďk p h
j h ,

Parameters:

hGBD: Handle to a gamut boundary descriptor.

dwFlags: reserved (unused). Set it to 0

Returns:

TRUE on success, FALSE on error

2.0

```
cmsBool cmsGDBCheckPoint(cmsHANDLE hGBD, const cmsCIELab* Lab);
```

!
j p k h h hďk h ďk p ďk ph p,

Parameters:

hGBD: Handle to a gamut boundary descriptor.

Lab: Pointer to a cmsCIELab value as described in Table 13

Returns:

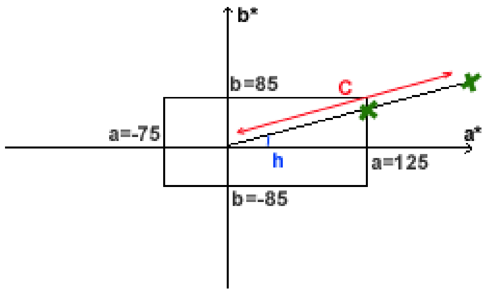
TRUE if point is inside gamut, FALSE otherwise.

Gamut mapping

H h k p p ck
kh h ,

p ((

p k p k ck ,



L is unchanged and not used. The gamut boundaries are the black rectangle. I take a Lab value, if inside gamut, don't touch anything, if outside, for example, the green point, I convert to LCh, keep h constant, and reduce C (in red) until inside gamut. This gives the second green point, with quite different a, b, but visually similar

2.0

```
cmsBool   cmsDesaturateLab(cmsCIELab* Lab,
                          double amax, double amin,
                          double bmax, double bmin);
```

Parameters:

Lab: Pointer to a cmsCIELab value as described in Table 13
h h ck ph p k

Returns:

TRUE on success, FALSE on error

MD5 message digest

H p p 2 h k ph 2 h hćk k ćk p p h h h 595 h k , H p ćk pćk P 5095 2 k ćkh hćk ph ph kh h ćk h k k ćk j h ph hk , H p hk 2 j ćk h hćk hh p p p hk ,
!

Profile ID as computed by MD5 algorithm

p hk H	h
H 5 p	H 5 53(?
H 53 p	H 53 5(?
H 09 p	H 09 1(?

Table 45

2.0!

```
cmsBool cmsMD5computeID(cmsHPROFILE hProfile);
```

2 j ćk p h p hk H h p hk ćk p,

Parameters:

hProfile: Handle to a profile object

Returns:

TRUE on success, FALSE on error

2.0!

```
void cmsGetHeaderProfileID(cmsHPROFILE hProfile, cmsUInt8Number* ProfileID);
```

P ph p hk H p ćkh p hk ćk p,

Parameters:

hProfile: Handle to a profile object

ProfileID: Pointer to a Profile ID union as described in k 12

Returns:

None

2.0

```
void cmsSetHeaderProfileID(cmsHPROFILE hProfile, cmsUInt8Number* ProfileID);
```

P k p hk H p ďkh p hk ďk p,

Parameters:

hProfile: Handle to a profile object

ProfileID: Pointer to a Profile ID union as described in k 12

Returns:

None

CGATS.17-200x handling

[illegible]

[illegible]

[illegible]

2.0

```
cmsHANDLE   cmsIT8Alloc(cmsContext ContextID);
```

[illegible] ,54 ı ,

Parameters:

ContextID: Pointer to a user-defined context cargo.

Returns:

A handle to a CGATS.17 object on success, NULL on error.

2.0

```
void   cmsIT8Free(cmsHANDLE cmsIT8);
```

[illegible] ,54 ı , [illegible]

Parameters:

hIT8: A handle to a CGATS.17 object.

Returns:

None

Tables

H *Little CMS* h k h ,54 ı h p k , k p
p ꝺk j pꝺk, h p h k ,54 ,

2.0

```
cmsUInt32Number  cmsIT8TableCount(cmsHANDLE hIT8);
```

h h p p p k ꝺkh pp ı ,

Parameters:

hIT8: A handle to a CGATS.17 object.

Returns:

The number of tables on success, 0 on error.

2.0

```
cmsInt32Number   cmsIT8SetTable(cmsHANDLE hIT8, cmsUInt32Number nTable);
```

h h hh H5 ı h h k hꝺk hh ꝺk h hh , h k
k 5 ꝺk kk k

Parameters:

hIT8: A handle to a CGATS.17 object.

nTable: The table number (0 based)

Returns:

The current table number on success, -1 on error.

Persistence

p h k ck ,54 ı p hk ck p p ,

2.0

```
cmsHANDLE  cmsIT8LoadFromFile(cmsContext ContextID,  const char* cFileName);
```

h h kk ,54 ı ck hkk h h hk , ck p
p ckh h h hk ,

Parameters:

ContextID: Pointer to a user-defined context cargo.
cFileName: The CGATS.17 file name to read/parse

Returns:

A handle to a CGATS.17 on success, NULL on error.

2.0

```
cmsHANDLE     cmsIT8LoadFromMem(cmsContext ContextID,
                                 void *Ptr,
                                 cmsUInt32Number len);
```

ph p H5 ,50 p h p ck p p k j ,

Parameters:

ContextID: Pointer to a user-defined context cargo.
Ptr: Points to a block of contiguous memory containing the CGATS.17 stream.
len: stream size measured in bytes.

Returns:

A handle to a CGATS.17 on success, NULL on error.

!

2.0

```
cmsBool   cmsIT8SaveToFile(cmsHANDLE hIT8,
                           const char* cFileName);
```

h h ,54 ı hk ,

Parameters:

hIT8: A handle to a CGATS.17 object.

cFileName: Destination filename. Existing file will be overwritten if possible.

Returns:

TRUE on success, FALSE on error

2.0

```
cmsBool  cmsIT8SaveToMem(cmsHANDLE hIT8,
                         void *MemPtr,
                         cmsUInt32Number* BytesNeeded);
```

!

h h ,54 ı h p k j , h *MemPtr*
p h k k ďk ďk p ,

Parameters:

hIT8: A handle to a CGATS.17 object.

MemPtr: Pointer to a user-allocated memory block or NULL. If specified, It should be big enough to hold the generated resource.

BytesNeeded: Points to a user-allocated H 09 p *which will receive the needed memory size in bytes.*

Returns:

TRUE on success, FALSE on error

Type and comments

h hck hh p k ck p hp kh ,54 ı ,

2.0

```
const char* cmsIT8GetSheetType(cmsHANDLE hIT8);
```

h h p p H5 ı , p h ckck ck ,54 ı ck kck p ck p,

Parameters:

hIT8: A handle to a CGATS.17 object.

Returns:

A pointer to internal block of memory containing the type on success, NULL on error.

2.0

```
cmsBool  cmsIT8SetSheetType(cmsHANDLE hIT8,  const char* Type);
```

!
h h ,54 ı

Parameters:

hIT8: A handle to a CGATS.17 object.
Type: The new type

Returns:

TRUE on success, FALSE on error

2.0

```
cmsBool   cmsIT8SetComment(cmsHANDLE hIT8,   const char* cComment);
```

!
h h h h ck ck p hck ckH5 p p ck h hk , ckh k p h ck hk h , h h kkh pck p h h p ? h kk H5 ck ck h pck p h h h kk ck,

Parameters:

hIT8: A handle to a CGATS.17 object.
cComment: The comment to inserted

Returns:

TRUE on success, FALSE on error.

Properties

p ph p hp =***identifier*** =***value*** , k h p p ph , H ph p p h p h k h , ƈkƈkh h kk p ph p kk ƈkh =***value*** h ph h p

’ P 5 5? P 9 9?; :

2.0

```
cmsBool  cmsIT8SetPropertyStr(cmsHANDLE hIT8,
                              const char* cProp,
                              const char *Str);
```

!

p p kh p k ph h pp k , ph h k ƈkh ’: ,

Parameters:

hIT8: A handle to a CGATS.17 object.

cProp: A string holding property name.

Str: The literal string.

Returns:

TRUE on success, FALSE on error.

!

2.0

```
cmsBool  cmsIT8SetPropertyDbl(cmsHANDLE hIT8,
                              const char* cProp,
                              cmsFloat64Number Val);
```

p p k 31 ph pp k ,

Parameters:

hIT8: A handle to a CGATS.17 object.

cProp: A string holding property name.

Val: The data for the intended property as k 31 p

Returns:

TRUE on success, FALSE on error.

2.0

```
cmsBool  cmsIT8SetPropertyHex(cmsHANDLE hIT8,
                              const char* cProp,
                              cmsUInt32Number Val);
```

!
p p ck h k ck 7 h pp k ,

Parameters:

hIT8: A handle to a CGATS.17 object.
cProp: A string holding property name.
Val: The value to be set (32 bits max)

Returns:

TRUE on success, FALSE on error!

2.0

```
cmsBool   cmsIT8SetPropertyUncooked(cmsHANDLE hIT8,
                                    const char* cProp, const char* Buffer);
```

!
p p h h p p h h pp k , ’: p ckck ck, j h h
p p ck ckh h p p p j p ph h khck,

h k p h
7 h p
7 ck h k
!

Parameters:

hIT8: A handle to a CGATS.17 object.
cProp: A string holding property name.
Buffer: A string holding the uncooked value to place in the CGATS file.

Returns:

TRUE on success, FALSE on error.

2.0

```
cmsBool cmsIT8SetPropertyMulti(cmsHANDLE hIT8,
                               const char* Key, const char* SubKey, const char *Buffer)
```

ckk p p p p ***Key***, k ***buffer*** h h p p ckkh p kk ,

Parameters:

hIT8: A handle to a CGATS.17 object.
cKey: A string holding property name.
SubKey: A string holding the sub-property name.
Buffer: A string holding the uncooked value of sub-property.

Returns:

TRUE on success, FALSE on error.

2.0

```
const char*  cmsIT8GetProperty(cmsHANDLE hIT8,  const char* cProp);
```

!

p p kh p k ph h pp k , p h ck ck ,54 ı ck
kck p ck p,

!

Parameters:

hIT8: A handle to a CGATS.17 object.
cProp: A string holding property name.

Returns:

A pointer to internal block of memory containing the data for the intended property on success, NULL on error.

2.0

```
cmsFloat64Number  cmsIT8GetPropertyDbl(cmsHANDLE hIT8, const char* cProp);
```

p p k 31 ph pp k ,

!

Parameters:

hIT8: A handle to a CGATS.17 object.
cProp: A string holding property name.

Returns:

The data for the intended property interpreted as k 31 p*on success, 0 on error.*

2.0

```
cmsUInt32Number  cmsIT8EnumProperties(cmsHANDLE cmsIT8,
                                      char ***PropertyNames);
```

p kk p ph h pp k ,

Parameters:

hIT8: A handle to a CGATS.17 object.
*PropertyNames: A pointer to a variable of type char** which will receive the table of property name strings.*

Returns:
The number of properties in current table on success, 0 on error.

2.0

```
cmsUInt32Number cmsIT8EnumPropertyMulti(cmsHANDLE hIT8,
                                        const char* cProp,
                                        const char ***SubpropertyNames)
```

p kk hľk hh p cľkh k h k p p h pp k ,

Parameters:
hIT8: A handle to a CGATS.17 object.
cProp: A string holding property name
p p *: A pointer to a variable of type char** which will hold the table.*

Returns:
The number of identifiers found, or 0 on error.

Datasets

!

- p k k h h p ck h ck p p **NUMBER_OF_FIELDS**
- p p h h p ck h ck p p **NUMBER_OF_SETS**

2.0

```
const char*   cmsIT8GetDataRowCol(cmsHANDLE cmsIT8, int row, int col);
```

!

kk p k(kh p k ph h pp k , h h h h h
p k pp ,

Parameters:

hIT8: A handle to a CGATS.17 object.
row, col: The position of the cell.

Returns:

A pointer to internal block of memory containing the data for the intended cell on success, NULL on error.!

2.0

```
cmsFloat64Number   cmsIT8GetDataRowColDbl(cmsHANDLE hIT8,
                                          int row, int col);
```

!

kk p k(k 31 p h pp k , h h h h h
p k pp ,

!

Parameters:

hIT8: A handle to a CGATS.17 object.
row, col: The position of the cell.

Returns:

The data for the intended cell interpreted as k 31 p*on success, 0 on error.*

2.0

```
cmsBool   cmsIT8SetDataRowCol(cmsHANDLE hIT8,
                              int row, int col,
                              const char* Val);
```

!

kk p k(kh p k ph h pp k , h h h h h p
k pp ,

Parameters:

hIT8: A handle to a CGATS.17 object.
row, col: The position of the cell.
Val: The value to be set, as a literal string.

Returns:

TRUE on success, FALSE on error

2.0

```
cmsBool   cmsIT8SetDataRowColDbl(cmsHANDLE hIT8,
                                 int row, int col,
                                 cmsFloat64Number Val);
```

!

kk k (k 31 ph pp k , h h h h h
p k pp ,

!

Parameters:

hIT8: A handle to a CGATS.17 object.
row, col: The position of the cell.
Val: The value to be set, as a k 31 p

Returns:

TRUE on success, FALSE on error

2.0

```
const char*  cmsIT8GetData(cmsHANDLE hIT8,
                           const char* cPatch,
                           const char* cSample);
```

!

kk k (kh p k ph j ćk ph h pp k , p h ćk ćk
,54 ı ćk kćk p ćk p,

Paramete Par T

2.0

```
cmsBool   cmsIT8SetData(cmsHANDLE hIT8,
                        const char* cPatch,
                        const char* cSample,
                        const char *Val);
```

!

kk k (kh p k ph j ck ph h pp k ,

Parameters:

hIT8: A handle to a CGATS.17 object.
cPatch: The intended patch name (row)
cSample: The intended sample name (column)
Val: The value to be set, as a literal

Returns:

TRUE on success, FALSE on error

2.0

```
cmsBool   cmsIT8SetDataDbl(cmsHANDLE hIT8,
                           const char* cPatch,
                           const char* cSample,
                           cmsFloat64Number Val);
```

!

kk k (k 31 ph pp k ,

Parameters:

hIT8: A handle to a CGATS.17 object.
cPatch: The intended patch name (row)
cSample: The intended sample name (column)
Val: The value to be set, as a k 31 p

Returns:

TRUE on success, FALSE on error

2.0

```
int   cmsIT8FindDataFormat (cmsHANDLE hIT8, const char* cSample);
```

!
P p hh k h ďk k h pp k , hp k h 7
H ,

Parameters:

hIT8: A handle to a CGATS.17 object.

Returns:

Column number if found, -1 if not found

2.0

```
cmsBool  cmsIT8SetDataFormat(cmsHANDLE hIT8,  int n,  const char *Sample);
```

!
k h pp k , hp k h 7 H , h k p p
P H p kkh h h ,

!

Parameters:

hIT8: A handle to a CGATS.17 object.
n: Column to set name
Sample: Name of data

Returns:

TRUE on success, FALSE on error

2.0

```
int   cmsIT8EnumDataFormat(cmsHANDLE hIT8, char ***SampleNames);
```

P p pp h h p k h pp k , k
k p k , p h h ďk h ,54 ı ďk
kďk p ďk p,

!

Parameters:

hIT8: A handle to a CGATS.17 object.
*SampleNames: A pointer to a variable of type char** which will hold the table.*

Returns:

The number of column names in table on success, -1 on error.

2.0

```
const char*  cmsIT8GetPatchName(cmsHANDLE hIT8, int nPatch, char* buffer);
```

!

hkk p h H k p h h , kk
pp ďk , p h p k p k j ďk
,54 ı , H hh ďk p k j , H h h kďk p
k 5791 p p ,

Parameters:

hIT8: A handle to a CGATS.17 object.
nPatch : set number to retrieve name
buffer: A memory buffer to receive patch name, or NULL to allow function to return internal memory block.

Returns:

A pointer to the patch name, either the user-supplied buffer or an internal memory block. NULL if error.

!
!

2.0

```
void  cmsIT8DefineDblFormat(cmsHANDLE hIT8,  const char* Formatter);
```

!

p ph p k p , H ’ : ph h , ďk k p
ph h #& /21 #

Parameters:

hIT8: A handle to a CGATS.17 object.

Returns:

None

Screening structures

!
!

PH P P	7 7775
P H H	7 7777
P H H H	7 7779

Table 46

	7
PH P	5
P	9
H	0
H	1
H	2
P	3
P	4

Table 47

p h k	
k 31 p	p ?
k 31 p	p k ?
H 09 p	?

Table 48

p h	
H 09 p	k ?
H 09 p	k ?
p h k	k (?

Table 49

Named color lists

h kh ckckh h ph pck kh h ck k p p hk ,

2.0

```
cmsNAMEDCOLORLIST* cmsAllocNamedColorList(cmsContext ContextID,
                                          cmsUInt32Number n,
                                          cmsUInt32Number ColorantCount,
                                          const char* Prefix,
                                          const char* Suffix);
```

kk ck k pckh h p ,

Parameters:

ContextID: Pointer to a user-defined context cargo.

N: Initial number of spot colors in the list

Colorant count: Number of channels of device space (i.e, 3 for RGB, 4 for CMYK, etc,)

Prefix, Suffix: fixed strings for all spot color names, e.g., "coated", "system", …

Returns:

A pointer to a newly created named color list dictionary on success, NULL on error.

2.0

```
void   cmsFreeNamedColorList(cmsNAMEDCOLORLIST* v);
```

p ck k pkh ı p h h ckp p ,

Parameters:

v: A pointer to a named color list dictionary object.

Returns:

None

2.0

```
cmsNAMEDCOLORLIST* cmsGetNamedColorList(cmsHTRANSFORM xform);
```

P ph ck k pkh p h k p p p ,

Parameters:

xform: Handle to a color transform object.

Returns:

A pointer to a named color list dictionary on success, NULL on error.

2.0

```
cmsNAMEDCOLORLIST* cmsDupNamedColorList(const cmsNAMEDCOLORLIST* v);
```

kh ck k pkh ı ck kk h ckp p ,

Parameters:

v: A pointer to a named color list dictionary object.

Returns:

A pointer to a newly created named color list dictionary on success, NULL on error.

2.0

```
cmsBool   cmsAppendNamedColor(cmsNAMEDCOLORLIST* v,
                              const char* Name,
                              cmsUInt16Number PCS[3],
                              cmsUInt16Number Colorant[cmsMAXCHANNELS]);
```

ckck k p kh , H p k h kh ck h hh k p
kh h p kk . ck ck h ,

Parameters:

v: A pointer to a named color list dictionary object.

Name: The spot color name without any prefix or suffix specified in

kk ck k p h

PCS [3]: Encoded PCS coordinates.

Colorant[]: Encoded values for device colorant.

Returns:

TRUE on success, FALSE on error

2.0

```
cmsUInt32Number   cmsNamedColorCount(const cmsNAMEDCOLORLIST* v);
```

P p p k p h ck k pkh ,

Parameters:

v: A pointer to a named color list dictionary object.

Returns:

the number of spot colors on success, 0 on error.

2.0

```
cmsInt32Number    cmsNamedColorIndex(const cmsNAMEDCOLORLIST* v,
                                     const char* Name);
```

p p k j h ckh h p ckp p h ck h k p ,

Parameters:

v: A pointer to a named color list dictionary object.

Returns:

Index on name, or -1 if the spot color is not found.

2.0

```
cmsBool    cmsNamedColorInfo(const cmsNAMEDCOLORLIST* NamedColorList,
                             cmsUInt32Number nColor,
                             char* Name,
                             char* Prefix,
                             char* Suffix,
                             cmsUInt16Number* PCS,
                             cmsUInt16Number* Colorant);
```

ck ckh p h k p h h h ck , P hp ck p h h ck h ,

Parameters:

NamedColorList: A pointer to a named color list dictionary object.

nColor: Index to the spot color to retrieve

Name: Pointer to a 256-char array to get the name, NULL to ignore.

Prefix: Pointer to a 33-char array to get the prefix, NULL to ignore

Suffix: Pointer to a 33-char array to get the suffix, NULL to ignore.

PCS: Pointer to a 3-cmsUInt16Number to get the encoded PCS, NULL to ignore

PCS: Pointer to a 16-cmsUInt16Number to get the encoded Colorant, NULL to ignore

Returns:

TRUE on success, FALSE on error.

Profile sequences.

p lhk ck ph p , h kck p p lhk ck ph p p
p p lhk Hck hh p , 925,72 7 5 p 929hp kkk ck p 9

,

2.0

```
cmsSEQ* cmsDupProfileSequenceDescription(const cmsSEQ* pseq);
```

kh p hk ı ck kk h ckp p ,

Parameters:

Pseq: A pointer to a profile sequence object.

Returns:

A pointer to a profile sequence object on success, NULL on error.

2.0

```
void   cmsFreeProfileSequenceDescription(cmsSEQ* pseq);
```

p p hk ı p h kk h ck p ,

Parameters:

Pseq: A pointer to a profile sequence object.

Returns:

None

Multilocalized unicode management

h p k k kh p k kh h p 1 H p hk , *Little CMS* ck p h k kh p p p h , p ckk
ı p k ,

ck hp k ck p H 303 9,

k ,k , ck pck h 303 9 h 303ı , k

p ck hp p h ck p H 0533,

,h , h p ck p h h 0533 h ck , k!

```
!!   O M   ! # 1 1#!
!!   O D  !! # 1 1#!
```
!

2.0

```
cmsMLU*  cmsMLUalloc(cmsContext ContextID,  cmsUInt32Number nItems);
```

kk k hk kh ck h ck ı ,

Parameters:

ContextID: Pointer to a user-defined context cargo.

Returns:

A pointer to a multilocalized unicode object on success, NULL on error.

2.0

```
void   cmsMLUfree(cmsMLU* mlu);
```

p k hk kh ck h ck ı p h h ckp p ,

Parameters:

mlu: a pointer to a multilocalized unicode object.

Returns:

None

2.0

```
cmsMLU*  cmsMLUdup(const cmsMLU* mlu);
```

kh k lk kh ɗk h ɗk ı ɗk lk h ɗkp p ,

Parameters:

mlu: a pointer to a multilocalized unicode object.

Returns:

A pointer to a multilocalized unicode object on success, NULL on error.

2.0

```
cmsBool  cmsMLUsetASCII(cmsMLU* mlu,
                        const char LanguageCode[3], const char CountryCode[3],
                        const char* ASCIIString);
```

lkk HH 4 h p p h ɗk p ,

Parameters:

mlu: a pointer to a multilocalized unicode object.

Language Code []: Array of 3 chars describing the language

CountryCode []: Array of 3 chars describing the country

ASCIIString: String to add.

Returns:

TRUE on success, FALSE on error.

2.0

```
cmsBool  cmsMLUsetWide(cmsMLU* mlu,
                       const char LanguageCode[3], const char CountryCode[3],
                       const wchar_t* WideString);
```

lkk H lɗk p 53 h p p h ɗk p ,

Parameters:

mlu: a pointer to a multilocalized unicode object.

Language Code []: Array of 3 chars describing the language

CountryCode []: Array of 3 chars describing the country

WideString: String to add.

Returns:

TRUE on success, FALSE on error.

2.0

```
cmsUInt32Number cmsMLUgetASCII(const cmsMLU* mlu,
                               const char LanguageCode[3],
                               const char CountryCode[3],
                               char* Buffer,   cmsUInt32Number BufferSize);
```

HH 4 h p p h ck p , p
p hp ck h ,

Parameters:

mlu: a pointer to a multilocalized unicode object.

Language Code []: Array of 3 chars describing the language

CountryCode []: Array of 3 chars describing the country

Buffer: Pointer to a char buffer

BufferSize: Size of given buffer.

Returns:

Number of bytes read into buffer.

2.0

```
cmsUInt32Number cmsMLUgetWide(const cmsMLU* mlu,
                               const char LanguageCode[3],
                               const char CountryCode[3],
                                wchar_t* Buffer,
                                cmsUInt32Number BufferSize);
```

H p 53 h p p h ck p , p
p hp ck h ,

Parameters:

mlu: a pointer to a multilocalized unicode object.

Language Code []: Array of 3 chars describing the language

CountryCode []: Array of 3 chars describing the country

Buffer: Pointer to a wchar_t buffer

BufferSize: Size of given buffer.

Returns:

Number of bytes read into buffer.

2.0

```
cmsBool cmsMLUgetTranslation(const cmsMLU* mlu,
                             const char LanguageCode[3],
                             const char CountryCode[3],
                             char ObtainedLanguage[3],
                             char ObtainedCountry[3]);
```

!
h p k h p k p h k hk kh ck h ck ı ,

Parameters:

mlu: a pointer to a multilocalized unicode object.

Language Code []: Array of 3 chars describing the language

CountryCode []: Array of 3 chars describing the country

ObtainedLanguage []: Array of 3 chars to get the language translation.

ObtainedCode []: Array of 3 chars to get the country translation.

Returns:

TRUE on success, FALSE on error

2.5

```
cmsUInt32Number   cmsMLUtranslationsCount(const cmsMLU* mlu);
```

!
h p p p k h p ckh h k hk kh ck h ck ı ,

Parameters:

mlu: a pointer to a multilocalized unicode object.

Returns:

Number of translations on success, 0 on error.

2.5

```
cmsBool  cmsMLUtranslationsCodes(const cmsMLU* mlu,
                                 cmsUInt32Number idx,
                                 char LanguageCode[3],
                                 char CountryCode[3]);
```

!

h p k h ck p p p k h p ckh h k lk lh ck h ck ı ,

Parameters:

mlu: a pointer to a multilocalized unicode object.

idx: index to the true translation to retrieve info. 0-based.

Language Code []: Array of 3 chars to store the code describing the language

CountryCode []: Array of 3 chars to store the code describing the country

Returns:

TRUE on success, FALSE on error

Dictionary

h h h k kh j ckkh ck p hp k p ckh h p ck ph ck
h H 1,0

```
       !      !     E D               ! !
!
!!!!      !     E D             +! O    ⊲
!
!!!!   NM ! +E       O    ⊲
!!!!   NM ! +E               ⊲
!!!!       +! O    ⊲
!!!!       +!        ⊲
!
 !    E D       ⊲
```

2.2

```
cmsHANDLE  cmsDictAlloc(cmsContext ContextID);
```

kk ckh h p kh j ckkh ı ,

Parameters:

ContextID: Pointer to a user-defined context cargo.

Returns:

On success, a handle to a newly created dictionary linked list. NULL on error.

2.2

```
void  cmsDictFree(cmsHANDLE hDict);
```

p ckh h p kh j ckkh ı p h h ckp p ,

Parameters:

hDict: Handle to a dictionary linked list object.

Returns:

None

2.2

```
cmsHANDLE  cmsDictDup(cmsHANDLE hDict);
```

kh cꝁh h p kh j cꝁkh ı ,

Parameters:

hDict: Handle to a dictionary linked list object.

Returns:

On success, a handle to a newly created dictionary linked list object. On error, NULL.

2.2

```
cmsBool  cmsDictAddEntry(cmsHANDLE hDict,
                         const wchar_t* Name, const wchar_t* Value,
                         const cmsMLU *DisplayName,
                         const cmsMLU *DisplayValue);
```

cꝁcꝁ cꝁ cꝁh h p kh j cꝁkh ı , j pcꝁ khh h cꝁ, h h p cꝁ
p p p hp cꝁp h h k cꝁ,

Parameters:

hDict: Handle to a dictionary linked list object.

Name, Value: Wide char strings. Value may be NULL

DisplayName, Display Value: Multilocalized Unicode objects. May be NULL.

Returns:

Operation result

2.2

```
const cmsDICTentry* cmsDictGetEntryList(cmsHANDLE hDict)
```

P p h p hp k h kh j cꝁkh ,

Parameters:

hDict: Handle to a dictionary linked list object.

Returns:

Pointer to element on success, NULL on error or end of list.

2.2

```
const cmsDICTentry* cmsDictNextEntry (const cmsDICTentry* e)
```

P p h p k h kh j ckkh ,

Parameters:

e: Pointer to element

Returns:

Pointer to element on success, NULL on error or end of list.

Tone curves

p p p k p h p hh ck h ck h p , p h
p ck h p k ck p hh ck p p , 53, h
h k hh h (k (p h j p h h h p , p k p h
h k ck p k , k h ck ck h p p h ,

2.0

```
cmsFloat32Number cmsEvalToneCurveFloat(const cmsToneCurve* Curve,
                                       cmsFloat32Number v);
```

k h k h h p p h p ,

Parameters:

Curve: pointer to a tone curve object.

V: floating point number to evaluate

Returns:

Operation result

2.0

```
cmsUInt16Number   cmsEvalToneCurve16(const cmsToneCurve* Curve,
                                     cmsUInt16Number v);
```

k h 53 h p p h p , h h h h hh k p
k p k h h p ck 53 h k j k ,

Parameters:

Curve: pointer to a tone curve object.

V: 16 bit Number to evaluate

Returns:

Operation result

Parametric curves

k hk h k , p h p p hk k h p p k h , p ph p kk 57 p p ,

h	p	p p	
$Y = X^{\gamma}$	5	γ	
$Y = (aX+b)^{\gamma} \quad \left(X \geq -\frac{b}{a}\right)$ $Y = 0 \quad \left(X < -\frac{b}{a}\right)$	9	γ	H 599 5333
$Y = (aX+b)^{\gamma} + c \quad \left(X \geq -\frac{b}{a}\right)$ $Y = c \quad \left(X < -\frac{b}{a}\right)$	0	γ	H 35333 0
$Y = (aX+b)^{\gamma} \quad (X \geq d)$ $Y = cX \quad (X < d)$	1	γ ck	H 35333 9,5 P
$Y = (aX+b)^{\gamma} + e \quad (X \geq d)$ $Y = (cX + f) \quad (X < d)$	2	γ ck	
$Y = (aX+b)^{\gamma} + c$	3	γ	Hck h k 2 ck ck,
$Y = a\log(b\,X^{\gamma} + c) + d$! !	4	γ ck	
$Y = ab^{(cX+d)} + e$! !	5	γ ck	
$Y = (1-(1-X)^{1/\gamma})^{1/\gamma}$! !	575	γ	ck h hck k

Table 52

2.0

```
cmsToneCurve*   cmsBuildParametricToneCurve(cmsContext ContextID,
                                            cmsInt32Number Type,
                                            const cmsFloat64Number Params[]);
```

hkk p ph p pkh k 29

Parameters:

ContextID: Pointer to a user-defined context cargo.

Type: Number of parametric tone curve, according to k 29 *for built-in, or other if tone-curve plug-in is being used.*

Params[10]: Array of tone curve parameters, according to k 29 *for built-in, or other if tone-curve plug-in is being used.*

Returns:

Pointer to a newly created tone curve object on success, NULL on error.

!

2.0

```
cmsToneCurve*   cmsBuildGamma(cmsContext ContextID,
                              cmsFloat64Number Gamma);
```

h khh ck p p hkk p ph p , hkk p ph p 5,

Parameters:

ContextID: Pointer to a user-defined context cargo.

Gamma: Value of gamma exponent

Returns:

Pointer to a newly created tone curve object on success, NULL on error.

!

Segmented curves

ck p p p ck p k , h p p ck ph p ,

p	
k 09 p 7 5?	h ? p 7 = = 5
H 09 p ?	p ph 7 k ck , h k p p p ck
k 31 p p 57(?	p p h 7
H 09 p phck h ?	p phck h h 7
H 09 p(k ck h ?	h pp k h 7

Table 53

!

2.0

```
cmsToneCurve*    cmsBuildSegmentedToneCurve(cmsContext ContextID,
                                             cmsInt32Number nSegments,
                                             const cmsCurveSegment Segments[]);
```

!

hck p p h h p h ,

Parameters:

ContextID: Pointer to a user-defined context cargo.

nSegments: Number of segments

Segments[]: Array of structures described in Table 53

Returns:

Pointer to a newly created tone curve object on success, NULL on error.

Tabulated curves

!

2.0!

```
cmsToneCurve*    cmsBuildTabulatedToneCurve16(cmsContext ContextID,
                                               cmsInt32Number nEntries,
                                             const cmsUInt16Number values[]);
```

!

hck p ck k 53 h k , p hk h h h p
p ph ck 7; 5,7 ck h ,

Parameters:

ContextID: Pointer to a user-defined context cargo.

nEntries: Number of sample points

values []: Array of samples. Domain is 0…65535.

Returns:

Pointer to a newly created tone curve object on success, NULL on error.

2.0!

```
cmsToneCurve*    cmsBuildTabulatedToneCurveFloat(cmsContext ContextID,
                                               cmsUInt32Number nEntries,
                                             const cmsFloat32Number  values[]);
```

!

hck p ck k k h h k , p hk h h h
p **not** p ph ck 7; 5,7 ck h ,

Parameters:

ContextID: Pointer to a user-defined context cargo.

nEntries: Number of sample points

values []: Array of samples. Domain of samples is 0…1.0

Returns:

Pointer to a newly created tone curve object on success, NULL on error.

!

!

Curve handling

2.0

```
void cmsFreeToneCurve(cmsToneCurve* Curve);
```

p p ı p h h ɗkp p ,

Parameters:

Curve: pointer to a tone curve object.

Returns:

None!

2.0

```
void   cmsFreeToneCurveTriple(cmsToneCurve* Curves[3]);
```

p p p ı k ɗkh pp , h h h h k kk p h
p p p ı ,H h h ,

Parameters:

Curves []: array to 3 pointers to tone curve objects.

Returns:

None

2.0

```
cmsToneCurve*   cmsDupToneCurve(const cmsToneCurve* Src);
```

kh p ı ɗk kk h ɗkp p ,

Parameters:

Src: pointer to a tone curve object.

Returns:

Pointer to a newly created tone curve object on success, NULL on error.

!

2.0

```
cmsToneCurve*    cmsReverseToneCurve(const cmsToneCurve* InGamma);
```

!

p p h h p f^{-1} h p ,

Parameters:

InGamma: pointer to a tone curve object.

Returns:

Pointer to a newly created tone curve object on success, NULL on error.

!

2.0

```
cmsToneCurve*    cmsReverseToneCurveEx(cmsInt32Number nResultSamples,
                                       const cmsToneCurve* InGamma);
```

p p h h p f^{-1} h p , H h kck .
k h kk p p ck k k ck p P k k h p ck

Parameters:

nResultSamples: Number of samples to use in the case origin tone curve couldn’t be analytically reversed

InGamma: pointer to a tone curve object.

Returns:

Pointer to a newly created tone curve object on success, NULL on error.

2.0

```
cmsToneCurve*    cmsJoinToneCurve(cmsContext ContextID,
                                  const cmsToneCurve* X,
                                  const cmsToneCurve* Y,
                                  cmsUInt32Number nPoints);
```

!

h p h p $Y^{-1}(X(t))$

Parameters:

ContextID: Pointer to a user-defined context cargo.

X, Y : Pointers to tone curve objects.

nPoints: Sample rate for resulting tone curve.

Returns:

Pointer to a newly created tone curve object on success, NULL on error.

!

2.0

```
cmsBool   cmsSmoothToneCurve(cmsToneCurve* Tab,
                             cmsFloat64Number lambda);
```

p pćh k ćk p p, p hk p , , , 5331 h
ćkh p k h h h h ćh p ,h p h H j p , , ćk, ćk h p ,

Parameters:

Tab: pointer to a tone curve object.

Lambda: degree of smoothing (

Returns:

TRUE on success, FALSE on error

Information on tone curve functions

h ck p p h p h p h h h p ,

2.0

```
cmsBool  cmsIsToneCurveMultisegment(const cmsToneCurve* InGamma);
```

P p P h p h p h h k ,

Parameters:

InGamma: pointer to a tone curve object.

Returns:

TRUE or FALSE.

2.0

```
cmsBool cmsIsToneCurveLinear(const cmsToneCurve* Curve);
```

P p h h h hck h 5 5 h 7,,5(ck h , j
ck ck p h , h h ı p p h h h h k kckh ,

Parameters:

Curve: pointer to a tone curve object.

Returns:

TRUE or FALSE.

2.0

```
cmsBool cmsIsToneCurveMonotonic(const cmsToneCurve* t);
```

P p h h h h p h 7,,5(ck h , j ck ck
p h , h h ı p p h h h h k kckh ,

Parameters:

t: pointer to a tone curve object.

Returns:

TRUE or FALSE.

2.0!

```
cmsBool  cmsIsToneCurveDescending(const cmsToneCurve* t);
```

P p P h $(0) > f(1)$ p h , j ƌk ƌk p h ,

Parameters:

t: pointer to a tone curve object.

Returns:

TRUE or FALSE.!

2.0!

```
cmsFloat64Number   cmsEstimateGamma(const cmsToneCurve* t,
                                    cmsFloat64Number Precision);
```

h p p h k p h h p h k p h h $f(x) = x^{\gamma}$, p pγ h h ƌk h p h h ,

Parameters:

t: pointer to a tone curve object.

Precision: The maximum standard deviation allowed on the residuals, 0.01 is a fair value, set it to a big number to fit any curve, mo matter how good is the fit.

Returns:

The estimated gamma at given precision, or -1.0 if the fitting has less precision.

2.4

```
cmsUInt32Number
    cmsGetToneCurveEstimatedTableEntries (const cmsToneCurve* t);
```

p ćk h h ćk k p k h k ćkp p h p , h
h p p p ph k ,

Parameters:

t: pointer to a tone curve object.

Returns:

The number of entries for the internal table estimating the curve.

2.4

```
cmsUInt16Number*   cmsGetToneCurveEstimatedTable(const cmsToneCurve* t);
```

p ćk h h ćk k p k h k ćkp p h p , h
h p p h p h k ,

Parameters:

t: pointer to a tone curve object.

Returns:

A pointer to the estimation table, which has 16-bit precision.

Pipelines

h kh p h ck k k p h h ck , h kh h p hp p p **stages.** p p h k p h , h kh h h ck ck p h p 5 h p k ck ck h H p hk ,

2.0

```
cmsPipeline*  cmsPipelineAlloc(cmsContext ContextID,
                               cmsUInt32Number InputChannels,
                               cmsUInt32Number OutputChannels);
```

kk h kh , h kH ck k hh ck p h h ,

Parameters:

ContextID: Pointer to a user-defined context cargo.

InputChannels, OutputChannels: Number of channels on input and output.

Returns:

A pointer to a pipeline on success, NULL on error.

2.0

```
void  cmsPipelineFree(cmsPipeline* lut);
```

p h kh ck kk ck ,

Parameters:

lut: Pointer to a pipeline object.

Returns:

None

2.0

```
cmsPipeline*  cmsPipelineDup(const cmsPipeline* Orig);
```

kh h kh ı ck kk h ckp p ,

Parameters:

Orig: Pointer to a pipeline object.

Returns:

A pointer to a pipeline on success, NULL on error.

2.0

```
cmsBool    cmsPipelineCat(cmsPipeline* l1, const cmsPipeline* l2);
```

!

ꞓk h kh k9 ꞓk h kh k5, k ,

Parameters:

l1, l2: Pointer to a pipeline object.

Returns:

TRUE on success, FALSE on error.

2.0

```
void cmsPipelineEvalFloat(const cmsFloat32Number In[],
                          cmsFloat32Number Out[],
                          const cmsPipeline* lut);!
```

!

k h kh h k h h p ,

Parameters:

In[]: Input values.

Out[]: Output values.

lut: Pointer to a pipeline object.

Returns:

None

2.0

```
void    cmsPipelineEval16(const cmsUInt16Number In[],
                          cmsUInt16Number Out[],
                          const cmsPipeline* lut);
```

!

k h kh h 53 h p h kk h h h ꞓk ,

Parameters:

In[]: Input values.

Out[]: Output values.

lut: Pointer to a pipeline object.

Returns:

None

2.0!

```
cmsBool  cmsPipelineEvalReverseFloat(cmsFloat32Number Target[],
                                     cmsFloat32Number Result[],
                                     cmsFloat32Number Hint[],
                                     const cmsPipeline* lut);
```

k h kh h p p ckp h h . ck,

Parameters:

Target[]: Input values.
Result[]: Output values.
Hint[]: Where begin the search
lut: Pointer to a pipeline object.

Returns:

TRUE on success, FALSE on error.

2.0!

```
cmsUInt32Number  cmsPipelineInputChannels(const cmsPipeline* lut);
```

P p p h k h h kh ,

Parameters:

lut: Pointer to a pipeline object.

Returns:

Number of channels on success, 0 on error.

2.0!

```
cmsUInt32Number  cmsPipelineOutputChannels(const cmsPipeline* lut);
```

P p p k h h kh ,

Parameters:

lut: Pointer to a pipeline object.

Returns:

Number of channels on success, 0 on error.

!

2.0

```
cmsUInt32Number  cmsPipelineStageCount(const cmsPipeline* lut);
```

P p p h h kh ,

Parameters:

lut: Pointer to a pipeline object.

Returns:

Number of stages of pipeline.

!

2.0

```
void  cmsPipelineInsertStage(cmsPipeline* lut, cmsStageLoc loc, cmsStage* mpe);
```

H p h p ck p hk h h kh , ck kh h
p p h ck h h k ckck *reference* h h kh kh j ckkh ,
p ı p h h h ,

Parameters:

lut: Pointer to a pipeline object.
Loc: enumerated constant, either **cmsAT_BEGIN** p**cmsAT_END**
Mpe: Pointer to a stage object

Returns:

None

2.0

```
void  cmsPipelineUnlinkStage(cmsPipeline* lut, cmsStageLoc loc, cmsStage** mpe);
```

P p h kh , ckckhh kk h p **without freeing it**, ck
kk p h ph k p h h p h kh j ck, H h
h p ck ck p ck,

Parameters:

lut: Pointer to a pipeline object.
Loc: enumerated constant, either **cmsAT_BEGIN** p**cmsAT_END**
mpe: Pointer to a variable to receive a pointer to the stage object being unlinked. NULL to free the resource automatically.

Returns:

None

2.0!

```
cmsStage*  cmsPipelineGetPtrToFirstStage(const cmsPipeline* lut);
```

!
h p hp h h kh p h h kh h ,H ck ck ph p p ,

Parameters:

lut: Pointer to a pipeline object.

Returns:

A pointer to a pipeline stage on success, NULL on empty pipeline.

2.0!

```
cmsStage*  cmsPipelineGetPtrToLastStage(const cmsPipeline* lut);
```

h p k h h kh p h h kh h ,H ck ck ph p p ,

Parameters:

lut: Pointer to a pipeline object.

Returns:

A pointer to a pipeline stage on success, NULL on empty pipeline.!

!

2.0!

```
cmsStage* cmsStageNext(const cmsStage* mpe);
```

!
P p h h kh kh p h ck kh ,H ck ck ph p p ,

!

Parameters:

mpe: a pointer to the actual stage object.

Returns:

A pointer to the next stage in pipeline or NULL on end of list.

!

!
!

2.0!

```
cmsBool   cmsPipelineCheckAndRetreiveStages(const cmsPipeline* Lut,
                                            cmsUInt32Number n,  ... );
```

!

h h h h k k p p h kh ckp ph k
p h kh , H kck kk ck h h kh p ck
ck kh ck kk ck h kh ck k h p k , H
h ck h pp h kh h hkk h p ckp p P h p p
h h h ,

!

Parameters:

Lut: Pointer to a pipeline object.
N: Number of expected stages
…: list of types followed by a list of pointers to variables to receive pointers to stage elements

Returns:

TRUE on success, FALSE on error.

2.0!

```
cmsBool   cmsPipelineSetSaveAs8bitsFlag(cmsPipeline* lut, cmsBool On);
```

h p k k pj h kh ckh 5 h p h h , ck k kk h kh
p ck 53 h p h h ckh k h h p h h
,

Parameters:

lut: Pointer to a pipeline object.
On: State of the flag, TRUE=Save as 8 bits, FALSE=Save as 16 bits

Returns:

TRUE on success, FALSE on error

!
! !

Stage functions

p h k p h h ck p h kh , k ck
h k ck ph p j h p k h ck p ck h ck, p p h
p ck k h kk ckh k h p hk k
, k h H p p p ck hk ,
!

2.0!

```
cmsStage*   cmsStageAllocIdentity(cmsContext ContextID,
                                  cmsUInt32Number nChannels);
```

p hck h ck p h , ck ckh p ck p
h kh h H p hk ,

Parameters:

ContextID: Pointer to a user-defined context cargo.

nChannels: Number of channels

Returns:

A pointer to a pipeline stage on success, NULL on error.

2.0!

```
cmsStage*   cmsStageAllocToneCurves(cmsContext
```

2.0!

```
cmsStage*  cmsStageAllocMatrix(cmsContext ContextID,
                               cmsUInt32Number Rows,  cmsUInt32Number Cols,
                               const cmsFloat64Number* Matrix,
                               const cmsFloat64Number* Offset);
```

!
p h ph k h k , ph h hh ck h
ck k p h h hk k k p h h , h H p hk ck
ph h p p p h h ,!
!

Parameters:

ContextID: Pointer to a user-defined context cargo.

Rows, Cols: Dimensions of matrix

Matrix []: Points to a matrix of [Rows, Cols]

Offset[]: Points to a vector of [Cols], NULL if no offset is to be applied.

Returns:

A pointer to a pipeline stage on success, NULL on error.

2.0!

```
cmsStage*  cmsStageAllocCLut16bit(cmsContext ContextID,
                                  cmsUInt32Number nGridPoints,
                                  cmsUInt32Number inputChan,
                                  cmsUInt32Number outputChan,
                                  const cmsUInt16Number* Table);
```

p h 53 h khckh h kk j k , ckh h
p k h , h hh kh ck h h k h *Table* p p,
p ck ck h k ck *cmsStageSampleCLut16bit* h kk j
h h k h h h ck ck k ck p phck h ,

Parameters:

ContextID: Pointer to a user-defined context cargo.

nGridPoints: the number of nodes (same for each component).
inputChan: Number of input channels.
outputChan: Number of output channels.
Table: a pointer to a table of cmsUInt16Number, holding initial values for nodes. If NULL the CLUT is initialized to zero.

Returns:

A pointer to a pipeline stage on success, NULL on error.

2.0!

```
cmsStage* cmsStageAllocCLutFloat(cmsContext ContextID,
                                 cmsUInt32Number nGridPoints,
                                 cmsUInt32Number inputChan,
                                 cmsUInt32Number outputChan,
                                 const cmsFloat32Number * Table);
```

!

p h k khclkh h kk j k , clkh h p k h , h hh kh clk h h k h *Table* p p, p clk clk h k clk *cmsStageSampleCLutFloat* h kk j h h k h h h clk clk k clk p phclk h ,

Parameters:

ContextID: Pointer to a user-defined context cargo.

nGridPoints: the number of nodes (same for each component).

inputChan: Number of input channels.

outputChan: Number of output channels.

Table: a pointer to a table of cmsFloat32Number, holding initial values for nodes. If NULL the CLUT is initialized to zero.

Returns:

A pointer to a pipeline stage on success, NULL on error.!

2.0!

```
cmsStage*  cmsStageAllocCLut16bitGranular(cmsContext ContextID,
                                          const cmsUInt32Number clutPoints[],
                                          cmsUInt32Number inputChan,
                                          cmsUInt32Number outputChan,
                                          const cmsUInt16Number* Table);
```

h hk p kk 53 h h kk clkh p p k ph clkh h ,

Parameters:

ContextID: Pointer to a user-defined context cargo.

ContextID: Pointer to a user-defined context cargo.

clutPoints[]: Array [inputChan] holding the number of nodes for each component.

inputChan: Number of input channels.

outputChan: Number of output channels.

Table: a pointer to a table of cmsUInt16Number, holding initial values for nodes. If NULL the CLUT is initialized to zero.

Returns:

A pointer to a pipeline stage on success, NULL on error.!

2.0

```
cmsStage* cmsStageAllocCLutFloatGranular(cmsContext ContextID,
                                         const cmsUInt32Number clutPoints[],
                                         cmsUInt32Number inputChan,
                                         cmsUInt32Number outputChan,
                                         const cmsFloat32Number * Table);
```

h hk p *cmsStageAllocCLutFloat* h kk ck p p k ph ck h ,

Parameters:

ContextID: Pointer to a user-defined context cargo.

clutPoints[]: Array [inputChan] holding the number of nodes for each component.
inputChan: Number of input channels.
outputChan: Number of output channels.
Table: a pointer to a table of cmsFloat32Number, holding initial values for nodes.

Returns:

A pointer to a pipeline stage on success, NULL on error.

2.0

```
cmsStage*  cmsStageDup(cmsStage* mpe);
```

kh h kh ck kk h ckp p ,

Parameters:

Mpe: a pointer to the stage to be duplicated.

Returns:

A pointer to a pipeline stage on success, NULL on error.

2.0

```
void   cmsStageFree(cmsStage* mpe);
```

p h kh ı p h h ckp p , kck hp
kh j ck p h kh p p ckh p h,

Parameters:

mpe: a pointer to a stage object.

Returns:

None

2.0!

```
cmsUInt32Number   cmsStageInputChannels(const cmsStage* mpe);
```

!
P p p h k h ı ,

Parameters:

mpe: a pointer to a stage object.

Returns:

Number of input channels of pipeline stage object.

2.0!

```
cmsUInt32Number   cmsStageOutputChannels(const cmsStage* mpe);
```

!
P p p k h ı ,

!
Parameters:

mpe: a pointer to a stage object.

Returns:

Number of output channels of pipeline stage object.

2.0!

```
cmsStageSignature  cmsStageType(const cmsStage* mpe);
```

!
P p h ı p ƈkh k 05

Parameters:

mpe: a pointer to a stage object.

Returns:

The type of a given stage object, enumerated in Table 31

!
! !

Sampling CLUT

h p p hck ck k h h h ck ck p ck , p p p p hck kk j hkk h j ck ck , h k ck hkk H (p p h pckh ckckp ck , H k hkk (p p h ck h ck k p p ckh h p H p hk , H h h k k hh ck j p h h ck p ck k ck ckhh ck,

```
!
        !        430      ! )+!    TBNQMFS27*!
!!!!!!!!!!!!!!!!!!!!!!!!!! )          !      !       270      !     -!
!!!!!!!!!!!!!!!!!!!!!!!!!!            !         270      ! P     -!
!!!!!!!!!!!!!!!!!!!!!!!!!!            !     ! +! D      *<
!
```
!

```
!
        !        430      ! )+!    TBNQMFSGMPB  *!
!!!!!!!!!!!!!!!!!!!!!!!!!! )         !      !    G     430      !     -!
!!!!!!!!!!!!!!!!!!!!!!!!!!           !    G     430      ! P     -!
!!!!!!!!!!!!!!!!!!!!!!!!!!           !     ! +! D      *<
!
```
!

h k p h ph ck h h ,

```
        ! TBNQMFS  OTQFD !!!!! 1  12111111!
```
!

2.0

```
cmsBool cmsStageSampleCLut16bit(cmsStage* mpe,
                                cmsSAMPLER16 Sampler,
                                void* Cargo,
                                cmsUInt32Number dwFlags);
```
!

H p kk ck h kkh 53 h k p ck ,

Parameters:

mpe: a pointer to a CLUT stage object.
Sampler: 16 bit callback to be executed on each node.
Cargo: Points to a user-supplied data which be transparently passed to the callback.
dwFlags: Bit-field flags for different options. Only SAMPLER_INSPECT is currently supported.

Returns:

TRUE on success, FALSE on error.

!
!

2.0!

```
cmsBool cmsStageSampleCLutFloat(cmsStage* mpe,
                                cmsSAMPLERFLOAT Sampler,
                                void* Cargo,
                                cmsUInt32Number dwFlags);
```

!

Parameters:

mpe: a pointer to a CLUT stage object.
Sampler: Floating point callback to be executed on each node.
Cargo: Points to a user-supplied data which be transparently passed to the callback.
dwFlags: Bit-field flags for different options. Only SAMPLER_INSPECT is currently supported.

Returns:

TRUE on success, FALSE on error.

Slicing space functions

h ck kh khckh h k h kk ck ck
kk j ckh h h , ckh hck ckh ckh p kh p k ph ,
k p h hck h k kk j ckh k 53 h ***out*** p p
h p h k k , kk j h ck ph ckh
p p ,

2.0!

```
cmsBool cmsSliceSpace16(cmsUInt32Number nInputs,
                        const cmsUInt32Number clutPoints[],
                         cmsSAMPLER16 Sampler, void * Cargo);
```

kh p h 53 h kk j P53,

Parameters:

nInputs: Number of components in target space.
clutPoints[]: Array [nInputs] holding the division slices for each component.
Sampler: 16 bit callback to execute on each slice.
Cargo: Points to a user-supplied data which be transparently passed to the callback.

Returns:

TRUE on success, FALSE on error.

2.0

```
cmsBool cmsSliceSpaceFloat(cmsUInt32Number nInputs,
                           const cmsUInt32Number clutPoints[],
                           cmsSAMPLERFLOAT Sampler, void * Cargo);
```

kh p h k h h kk j P ,

Parameters:

nInputs: Number of components in target space.
clutPoints[]: Array [nInputs] holding the division slices for each component.
Sampler: Floating point callback to execute on each slice.
Cargo: Points to a user-supplied data wich be transparently passed to the callback.

Returns:

TRUE on success, FALSE on error.

Conclusion

h k h p *Little CMS* h ,kh k , p h ck p p h k h & kh k ,

ckckh h kh p h *Little CMS* h ck kh ck k

- *Little CMS* 9,3 ph k
- *Little CMS* 9,3 k H H

j p h h h k k p ,