EDIUS Plug-In SDK

Programmer's Manual

Updated: 2003/08/08
Version: 1.87

canopus

==========

● はじめに ●

==========

■ようこそ EDIUS Plug-In SDK へ

このドキュメントは EDIUS の Plug-In を作成する為に必要な情報を掲載しております。EDIUS Plug-In は Audio や Video に効果を与える Video Filter, Audio Filter, Transition, Cross Fade, Keyer, Title Effect の６種類の 効果用 Plug-In と Expoter, Importer の２種類の入出力用 Plug-In の合計８種類の Plug-In があります。

■ご使用に当たっての留意事項

1. 本ソフトウェアは参考提供です。試用に当たってのサポートは行っておりませんのでご了承ください。
2. ご使用上の過失の有無を問わず､本ソフトウェアの運用において発生した逸失利益を含む特別､付随的､または派生的損害に対するいかなる請求があったとしても、当社はその責任を負わないものとします。

○ EDIUS Plug-In SDK について

SDK に付属する「Sample Plug-In Project」は VC++ ver6.0 (Service Pack5)にて作成しております。
EDIUS Plug-In 及び Plug-In を管理する Plug-In Manager は 全て COM like なものであり, interface と implementation を完全に分離するという考え方で開発するようになっています。
COM の詳細は web 上のリソースや MSDN をご参照ください。

○ パラメータについて

interface 仕様における 各パラメータ の先頭の in/out の意味は以下の通りです。

[in]：入力用パラメータ。この引数は method 内で値が読み出されるだけであり、値の上書き等はできません。

[out]：出力用パラメータ。この引数は method 内で値が書き込まれます。原則として書き込み用のメモリ領域は method 呼び出し側で確保してください。

○ Plug-In のインストールについて

作成した Plug-In は基本的に EDIUS の Plug-In フォルダに格納してください｡サブフォルダを作成して、その中に格納することもできます。
またショートカットを作成することによって別フォルダを指定することもできます。
EDIUS の Plug-In Manager は Plug-In フォルダとして指定されたフォルダをルートにして､サブフォルダ内を再帰的に検索します。ショートカットが存在する場合はショートカット先も検索するようになっています。(ショートカット先がフォルダである場合は、更に再帰的に検索を行います)

==========

● Plug-In DLL について ●

==========

EDIUS Plug-In DLL は Plug-In の実装を DLL としてファイル化したもので拡張子は "tpi"となります。1 つの DLL には複数の Plug-In を含めることができます。

全ての Plug-in は Plug-in Manager によって管理します。

EDIUS Plug-In DLL は特定の EDIUS Plug-In フォルダに格納され、EDIUS の起動時に Plug-In Manager は Plug-In フォルダにある全ての Plug-In DLL をロードします。

Plug-In DLL は Startup と Shutdown の２つの関数を export します。Plug-In Manager は Plug-In DLL のロード成功後直ちに Startup 関数を実行します。また、Plug-In Manager が終了する時、Plug-In DLL 全ての Shutdown 関数を実行し，その DLL を FreeLibrary() とします。

Startup 関数では、plug-in のインスタンスを生成し、Plug-In Manager に登録します。Plug-in の登録の為に、Startup 関数の引数に指定されている plug-in manager の interface を使用します。(ICtsPlugInServiceFactory)

< Plug-In DLL の export 関数 >

BOOL WINAPI Startup (ICtsPlugInServiceFactory *pServiceFactory)

- -

Paramter:

[in] ICTSPlugInServiceFactory *pServiceFactory

Plug-In Manager が提供するサービスを取得する為の interface ポインタです。これを使用して Plug-In 登録や Plug-In Manager への Notification をする為の ICtsPlugInRenderer, ICtsPlugInNotifier interface を取得します。(詳細は Plug-In Manager interface の項目を参照)

Return Value:

通常は TRUE を返し，関数が失敗した時には FALSE を返します。

Remarks:

Plug-In Manager は Plug-In DLL ロードに成功した直後に Startup 関数を GetProcAddress で取得して実行する。この関数の返り値が FALSE の場合、Plug-In Manager はその Plug-In DLL Handle を直ちに解放(::FreeLibrary)する。Plug-In DLL は Startup 関数の中で以下の手順に従って、Plug-In を登録しなければならない。(サンプル: PlugInDLL.cpp)

1. Plug-In の実体を作成する
2. ICtsPlugInRegisterer interface を取得する
3. ICtsPlugInRegisterer::RegisterPlugIn 関数で plug-in の ICtsPlugInInfo interface を登録する

BOOL WINAPI Shutdown (ICtsPlugInServiceFactory *pServiceFactory)

- -

Paramter:

[in] ICtsPlugInServiceFactory *pServiceFactory

Plug-In Manager が提供するサービスを取得する為の interface ポインタです。これを使用して Plug-In 登録や Plug-In Manager への Notification をする為の ICtsPlugInRenderer, ICtsPlugInNotifier interface を取得します。(詳細は Plug-In Manager interface の項目を参照)

Return Value:

通常は TRUE を返し、関数が失敗した時には FALSE を返します。

Remarks:

Plug-In Manager は終了する時，ロードしていた全ての Plug-In DLL の

Shutdown 関数を実行し、その後 DLL Handle を解放(::FreeLibrary)します。

==========

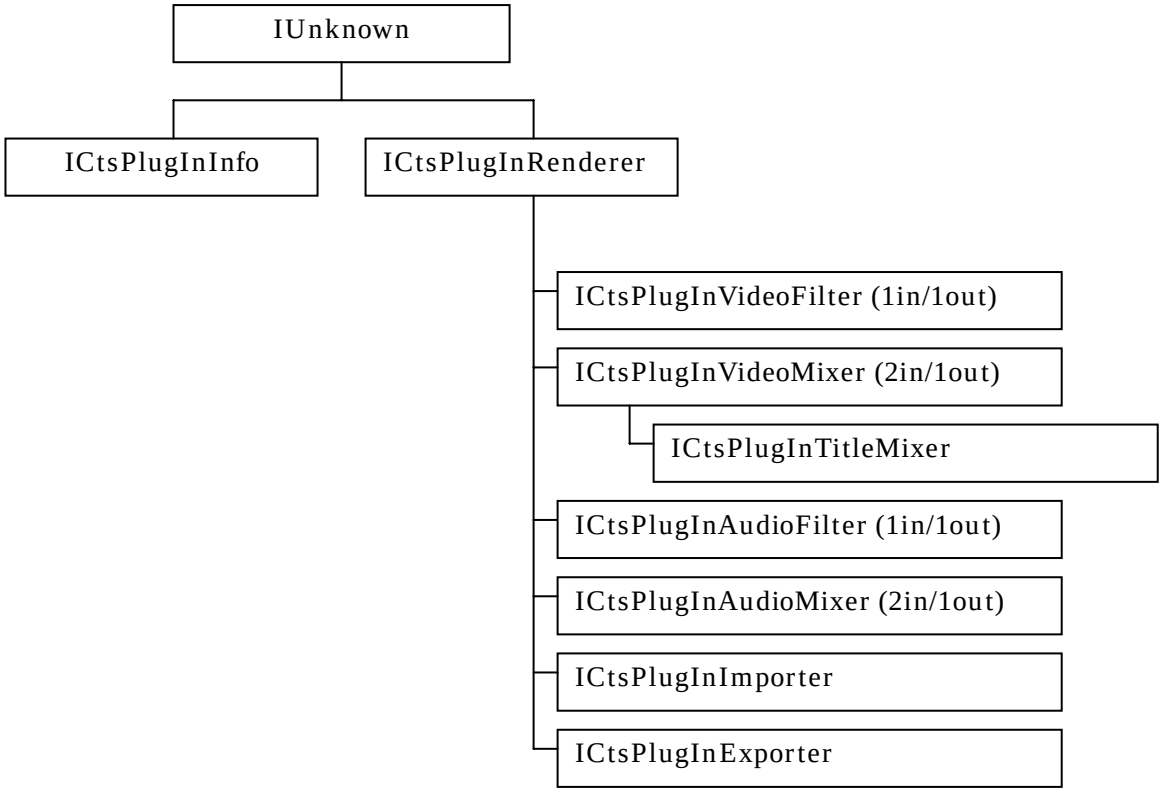
● Plug-In 概要 ●

==========

○ Plug-In の interface

- Plug-In の Interface は Implementation から完全に分離されている.

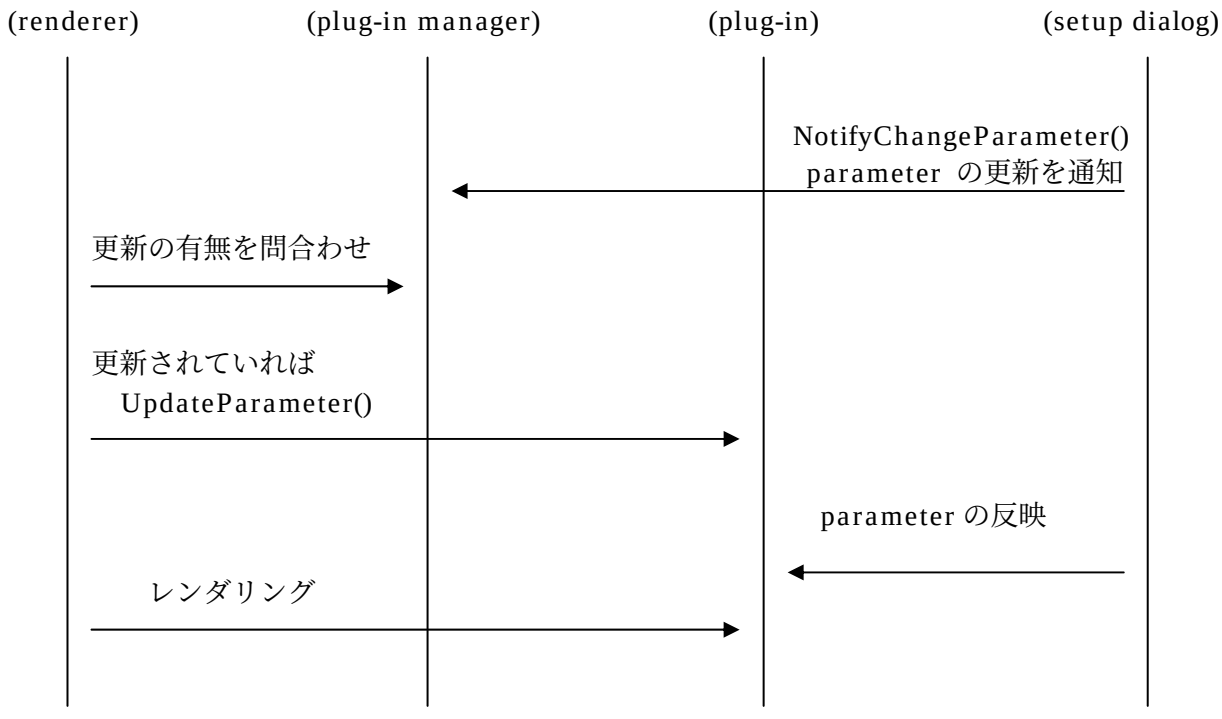
[Plug-In Interface クラス階層図]



○ Plug-In の実装
- Plug-In は上記 Interface を継承し実体を定義するものとする。
- 後述する通り、ICtsPlugInInfo interface の CreateInstance 関数を使って ICtsPlugInRenderer が取得できなければならない。
- また、ICtsPlugInRenderer interface の QueryInterface から ICtsPlugInInfo interface を取得できなければならない。
- ICtsPlugInInfo の QueryInterface を使って ICtsPlugInRenderer interface が取得できてはならない。

○ Plug-In の Setup Dialog と Dialog 上の Parameter 値
- Plug-In は必要に応じて Setup Dialog を持つことができる。
- Dialog は モードレスダイアログとする
- CreateSetup() で Dialog の実体を Create し、DestroySetup() で Dialog の実体を Destroy すること。
- CreateSetup() 関数が繰り返し呼び出されてもメモリリークが発生しないようにすること。 setup dialog 再表示の時, DestroySetup() が呼び出されずに CreateSetup() 関数が 呼び出されることがある。
- Dilaog が閉じられる時、ICtsPlugInNotifier::NotifyCloseSetup() 関数で Plug-In Manager にその旨を通知すること。
- Dialog 上のコントロールで parameter が変更された時には、 ICtsPlugInNotifier::NotifyChangedParameter()関数で Plug-In Manager に その旨を通知すること。parameter 値を直ぐに plug-in 本体に反映させてはならない。 parameter 値の反映は Updateparameter() 関数にてスレッドセーフに行うこと。 これは Dialog が表示されている最中も Renderer thread がマルチスレッドに plug-in のレンダリング処理を呼び出すことに対応する為であり、レンダリング処理の 直前に plug-in の UpdateParemter() が呼び出されることによって parameter 値の同期 が取られることを保証するものである。

[parameter update の流れ 概略図]



○ Plug-In の Packed Parameter Format
- Plug-In は CreatePackedParameter() 関数の中で，一連の Paramter 値を以下の format に従って格納するのに必要なメモリを確保し，値を書き込まなければならない。format は以下の通りである。

Packed Version (DWORD : 4)	Length of MatchName (DWORD : 4)	Match Name (Lengthof MatchName)	Plug-In Version (DWORD : 4)	Plug-In 任意の構造

- Plug-In の CreateInstance で Parameter が渡された場合、Pack された Parameter を解釈し、Parameter 値を反映した plug-in のレンダリング用インスタンスを作成しなければならない。
- Packed Version には CtsPlugInCommon.h に定義されている DWORD 定数 CtsPackedParameterVersion の値を格納する。
- Plug-In Version は上位 WORD を Major 番号、下位 WORD を Minor 番号とする DWORD 値である．Version 番号の付け方は Plug-In 作成者が任意に定めることが出来る。

* Plug-In Version を更新した場合、古い Version の packed parameter も読み出すことが出来るように下位互換性を持たせること。

、

○ Plug-In が返すエラー値
- HRESULT 型のエラーを返す必要がある場合、基本的なエラー値として CtsErr.h に記載されているものを使用する。それ以外に WinError.h に記載のあるエラー値を使用することも出来る。
- Interface 仕様 に 「エラー値が返る」と記載のあるものは、上記に従って CtsErr.h か WinError.h に記載のあるエラー値が返ることを表している。

○ Plug-In Suite interface
- Suite interace とは、Plug-In が自身からみた外部情報を取得する為の Interface のことである。
- Plug-In は必要があれば ICtsPlugInNotifier::CreateSuite 関数を呼ぶことによって Plug-In Manager に Suite の作成を要求出来る。
- Suite interface には以下のものが存在する。
  + ICtsPlugInTimeLineSuite : Time Line の情報を得る
  + ICtsPlugInProjectSettingSuite : Project 設定の情報を得る
  + ICtsPlugInObjectSuite : Time Line 上 Object の情報を得る
  + ICtsPlugInPlatformSuite : plat form の情報を得る
  + ICtsPlugInColorSelectorSuite : 色選択ダイアログを表示する
- Suite が必要なくなった場合，メモリ節約の為積極的に Release しなければならない。

○ Plug-In と CtsCore.Dll
- Plug-In の実装において ICtsPix, ICtsSnd, ICtsMemory の実体を生成する場合，CtsCore.dll と CtsCore.lib が必要となる。使用方法は Core dll に関するドキュメントを参照のこと。また sample Plug-in のソースにも使用方法の記載がある。

==========

● Plug-In ガイドライン ●

==========

plug-in の実装に際して従う必要のあるガイドラインを以下に示します。

<dialog>

- OnOk, OnCancel の時以下の順序で関数を必ず呼び出すようにして下さい。

```
ICtsPlugInNotifier::NotifyChangeParameter()
ICtsPlugInNotifier::NotifyCloseSetup()
```

<Render 関数>

- Render 関数は ODD/EVEN 別々のスレッドから呼び出されます。
- Render 関数の中で FrameSize や AspectRate 等の情報を取得する際は
  引数の pPixResult から取得するようにして下さい。
  (例)

```
SIZE          size = pPixResult->GetFrameSize();
CtsFraction   aspect = pPixResult->GetFrameAspect();
```

==
□ interface 仕様
==

Plug-In interface

=-=
<ICtsPlugInInfo>
=-=
[Required]
  CtsPlugInInfo.h

[Interface ID]
  IID_ICtsPlugInInfo

[Derived]
  IUnknown

[Remarks]
  - 使用される文字列は全て UNICODE に対応させるものとする。
  - 使用される文字列(LPCTSTR 型)は末尾が２つ連続した NULL 文字にすること
    (example)　　_T("Canopus EDIUS Video Filter Plug-In Mozaic¥0")
  - Plug-In Manager が Plug-In リストとして保持する interface
  - Plug-In Manager には、この interface へのポインタを登録します。

[method overview]
  - Plug-In の名称や Plug-In のアイコン等の Plug-In 情報を提供する。
  - Rendering 機能を備えた Plug-In のインスタンスが生成できる。

CtsPlugInType ICtsPlugInInfo::GetType()
- -
Paramter:
  none
Return Vvalue:
  Plug-In のタイプを返す。
Remark:
  Plug-In のタイプを取得する。
  CtsPlugInType 型は CtsPlugInInfo.h に定義されている。

LPCTSTR ICtsPlugInInfo::GetMatchName()
- -
Paramter:
  none
Return Value:
  関数が成功したらローカライズされていない plug-in 名のアドレス値が返る。
  関数が失敗したら NULL が返る。
Remark:
  この関数はローカライズされていない plug-in 名を返す。
  plug-in match name を格納する為のメモリは plug-in 側で確保/解放すること。
  plug-in manager はこの関数で取得したポインタを使って delete することは無い。

この文字列はシステムがプラグインを一位に識別する為に使用する為、
他のプラグインと重複しないように命名する必要がある。
基本的に，"企業名"，"カテゴリ名"，"プラグイン名" を繋げた表記とする。
例：_T("Canopus Co., Ltd. Video Filter Plug-In Monotone¥0")など

LPCTSTR ICtsPlugInInfo::GetVisibleName()
- -
Paramter:
none
Return Value:
関数が成功したらローカライズされた plug-in 名のアドレス値が返る。
関数が失敗したら NULL が返る。
Remark:
この関数はローカライズされた plug-in 名を返す。
visible name を格納する為のメモリは plug-in 側で確保/解放すること。
plug-in manager はこの関数で取得したポインタを使って delete することは無い。
この文字列は情報パレットやエフェクトパレットに表示される。
例：_T("Monotone¥0")

LPCTSTR ICtsPlugInInfo::GetHierarchy()
- -
Paramter:
none
Return Value:
成功したら，階層構造を意味する文字列を返す。
失敗したら NULL を返す。
Remark:
エフェクトパレットはツリーコントロールを用いてエフェクト群を表示する。
エフェクトパレットはこの関数から得られる階層構造文字列を解析し
エフェクトツリー内の適当な位置にそのプラグインを表示する。

特に指定する必要が無い場合は _T("¥¥¥0") を返すこと。
初めてプラグインがロードされた時にエフェクトツリーの特定の位置に表示
させたい場合、プラグインは階層構造を表す文字列を返さなければならない。
文字列の中に ¥¥ が含まれる場合、その箇所までの文字列でフォルダを検索します。

例 1)
返値 : _T("¥¥folder¥¥¥0")
表示 : --- [ ] Video Filter
|
+--[ ] folder
|
+-- [ ] <visible name>

LPCTSTR ICtsPlugInInfo::GetDescription()
- -
Paramter:
  none
Return Value:
  plug-in の説明文をローカライズされた文字列で返す。
  関数が失敗したら NULL を返す。
Remark:
  プラグインを説明する文字列を返さなければならない。
  返すべき文字列はローカライズされた文字列とする。プラグインが多言語対応
  する必要がある場合はロケールに合わせて適当なローカライズド文字列が返る
  ようにしなければならない。

  この文字列用のメモリはプラグイン側で確保/解放するものとする。
  plug-in manager がこの関数によって得たポインタを delete することは無い。

  例：_T("monotone filter¥0")

LPCTSTR ICtsPlugInInfo::GetVendorName()
- -
Paramter:
  none
Return Value:
  plug-in のベンダー名をローカライズされた文字列で返す。
  関数が失敗したら NULL を返す。
Remark:
  プラグインの制作者を表すベンダー名を返さなければならない。
  返すべき文字列はローカライズされた文字列とする。プラグインが多言語対応
  する必要がある場合はロケールに合わせて適当なローカライズド文字列が返る。
  ようにしなければならない。

  この文字列用のメモリはプラグイン側で確保/解放するものとする。
  plug-in manager がこの関数によって得たポインタを delete することは無い。

  例：_T("Canopus Co.,Ltd.¥0")

DWORD ICtsPlugInInfo::GetVersion()
- -
Paramter:
  none
Return Value:
  Plug-In の version を返す。
  version 番号は 32bit 値で表され、上位 16bit をメジャー番号、下位 16bit を
  マイナー番号とする。MAKEWPARAM(minor, major) マクロを使用して値を返す
  と良い。
Remark:
  Plug-In の Version を取得する。

  例： version 1.0 -> MAKEWPARAM(0, 1);

HRESULT ICtsPlugInInfo::GetIcon(ICtsPix **ppIconPix)

- -

Paramter:

[out] ICtsPix **ppIconPix

Plug-In の Icon を格納する為の ICtsPix interface のポインタ

Return Value:

成功したら S_OK を返す。

失敗したらエラー値を返す。

Remark:

エフェクトパレットから開かれるプロパティダイアログに独自のアイコンを
表示させることができる。
必要であればこの関数の第１引数に独自のアイコンを意味する Pix のインターフェースを渡さなければならない。

EDIUS でサポートされているプラグイン用アイコンは以下のフォーマット
を持った Pix でなければならない。

size : 64 x 64
color : RGB 32 bit color
binary: BGRABGRABGRABGRA....
transparent color : RGB(0, 128, 128)

独自にアイコンを持つ必要が無い時、*ppIconPix に NULL を格納し返値として
S_OK を指定すること。
この時プラグインを表すアイコンとしてシステムに標準で用意されているアイコンが使用される。

HRESULT ICtsPlugInInfo::CreateInstance(ICtsCoreMemory *pParameter, DWORD objectID, ICtsPlugInRenderer **ppRenderer)

- -

Parameter:

[in] ICtsCoreMemory *pParameter

レンダリング用インスタンスを生成する時に反映させるパラメータ値を
Packed したメモリへの interface ポインタが渡される。

[in] DWORD objectID

オブジェクト ID. Plug-in が関連付けられる Time Line 上の object の ID

[out] ICtsPlugInRenderer **ppRenderer

生成したレンダリング用インスタンスを格納する。
生成に失敗した場合は *ppRenderer に NULL を設定する。

Return value:

成功したら S_OK を返す。

失敗したらエラー値を返す。

Remarks:

レンダリング用の plug-in インスタンスを取得する。
もし pParameter が NULL である場合, レンダリング用インスタンスは
デフォルト値を持ったインスタンスとして生成すること、pParameter が NULL
で無い場合は Packed された parameter を反映させたインスタンスとして
生成すること。
レンダリング用インスタンスの生成に失敗したら *ppRenderer に NULL を設定する。

例えば，Video Filter plug-in の場合、この関数は ICtsVideoFilter を継承して
実装したクラスの ICtsPlugInRenderer interface を取得する。
また、Transition plug-in の場合、この関数は ICtsVideoMixer を継承し、
実装したクラスの ICtsPlugInRenderer interface を取得する。

ICtsPlugInRenderer interface は実際のレンダリング処理や parameter 設定用
ダイアログの表示の為に使われる。

=-=
<ICtsPlugInRenderer>
=-=

[Required]
CtsPlugInRenderer.h

[Interface ID]
IID_ICtsPlugInRenderer

[Derived]
IUnknown

[Remarks]
- Plug-In が持つ Setup dialog に関する method，及び Setup Dialog によって設定される Parameter に関する method が含まれる。
- EDIUS の Renderer はこの interface を使用する。

[method overview]
- Setup Dialog の表示、Paramter の serialize

HRESULT ICtsPlugInRenderer::CreateSetup(HWND hParent)
- -

Paramter:
[in] HWND hParent
Setup 表示の親となる Window の Handle を指定する。

Return value:
成功したら S_OK を返す。
失敗したらエラー値を返す。

Remarks
setup dialog を作成する．dialog はモードレスであること。
setup dialog は ShowSetup() 関数が呼び出されるまで表示されない。

HRESULT ICtsPlugInRenderer::ShowSetup(BOOL bShow)
- -

Paramter:
[in] BOOL bShow
setup dialog の表示/非表示を指定
TRUE なら表示，FALSE なら非表示

Return value:
成功したら S_OK を返す。
失敗したらエラー値を返す。

Remarks:
setup dialog を表示/非表示にする。

HRESULT ICtsPlugInRenderer::DestroySetup()
- -

Paramter:
none

Return value:

成功したら S_OK を返す。
失敗したらエラー値を返す。
Remarks
setup dialog を破棄する。

HRESULT ICtsPlugInRenderer::UpdateParameter()
- -
Paramter:
none
Return value:
成功したら S_OK を返す。
失敗したらエラー値を返す。
Remarks
setup dialog 上の parameter 値をプラグインのレンダリング用インスタンス
本体の parameter 値に反映する。この関数は Renderer がレンダリングする前
に必ず呼び出す。即ちこの関数が呼び出される時, Renderer はその処理の
直前であるため plug-in 側で parameter の排他制御を行う必要は無い。

HRESULT ICtsPlugInRenderer::CreatePackedParameter(ICTSCoreMemory **ppParameter)
- -
Paramter:
[out] ICTSCoreMemory *pMemory
Parameter が Packed されたメモリへの interface pointer
Return value:
成功したら S_OK を返す。
失敗したらエラー値を返す。
Remarks
plug-in の parameter を pack する method 内で、必要なメモリを確保し、それへの
interface ポインタをセットする。
メモリ不足などで pack 出来なかった場合は、*pMemory に NULL を セットして
エラー値を返す。

CtsFlags ICtsPlugInRenderer::GetCapability()
- -
Parameter:
none
Return value:
plug-in が持つ機能を表すフラグを返す。
機能を表すフラグは CtsPlugInRenderer.h に定義されている
Remarks:
Plug-In がサポートする機能を取得する。

HRESULT ICtsPlugInRenderer::IsShownSetup()
- -
Parameter:
none
Return value:
setup dialog が表示されていれば S_OK を返す。

setup dialog が表示されていなければ S_FALSE を返す。
Remarks:
Setup Dialog が表示されているかどうかを調べる。

=-=
<ICtsPlugInVideoFilter>
=-=

[Required]
  CtsPlugInVideoFilter.h

[Interface ID]
  IID_ICtsPlugInVideoFilter

[Derived]
  ICtsPlugInRenderer

[method overview]
- このインターフェースには video をフィルタリングする為の関数が含まれる
- このインターフェースを継承して video filter を作成することができる。

HRESULT ICtsPlugInVideoFilter::RenderBegin(void)

- -

Paramter:
  none
Return Value:
  成功したら S_OK を返す。
  失敗したらエラー値を返す。
Remarks:
  このプラグインインスタンスを使ってレンダリングを開始する時に呼び出される。
  plug-in は必要があれば、memory の確保、lookup table の生成などを行う
  rendering sequence の開始時に一回だけ呼び出される RenderBegin の後、
  0 回以上の Render method の呼び出しがあり、その後 RenderEnd が呼び出される。
  任意の instance について RenderBegin/RenderEnd を入れ子で呼び出すことはない。

```
HRESULT ICtsPlugInVideoFilter::Render(
      ICTSPix              *pPixResult,
      ICTSPix              *pPixSource,
      RECT const           *prcRegion,
      CtsFrame const    frmDuration,
      CtsFrame const    frmFrame,
      CtsField const    field,
      BOOL const           bScrub
      )
```

- -

Paramter:
  [out] ICTSPix *pixResult
    レンダリング結果を格納する為の ICtsPix interface ポインタ。
  [in] ICTSPix *pixSource
    元画像
  [in] RECT *prcRegion
    レンダリングの対象となる Rect。
  [in] CtsFrame frmDuration

plug-in が関連付けられている TIme Line 上の Object の duration。
[in] CtsFrame frmFrame
レンダリングするフレーム番号、IN 点からの相対値
[in] CtsField field
field (Odd or Even)
[in] BOOL bScrub
スクラブ中かどうかを表すフラグ。
TRUE ならスクラブ中。
Return Value:
成功したら S_OK を返す。
失敗したらエラー値を返す。
Remarks:
レンダリングを行う。pPixSource は元画像への interface ポインタ。
レンダリング中、pPisSource の内容を変更してはならない。
pPixResult はレンダリング結果を格納するための interface ポインタ
だが, 画像イメージのデータ以外変更してはならない。
画像イメージデータは ICtsPix::GetImageBuffer() を呼ぶことで取得する
ことが出来る。

Renderer は Odd フィールドと Even フィールドを非同期にレンダリング
する。もし Even フィールドのレンダリング中に Even フィールドを参照
しなければいけないような場合は、plug-in 側で同期処理を行う必要が有。
Renderer はフレーム以上のずれが発生するような非同期処理を行わない。
Rendering はフィールド単位で行われることに注意すること。

HRESULT ICtsPlugInVideoFilter::RenderEnd(void)
- -
Paramter:
none
Return Value:
成功したら S_OK を返す。
失敗したらエラー値を返す。
Remarks:
このインスタンスを使ってレンダリングをしなくなる時に呼び出される。
RenderBegin で確保したリソースを開放するなどの処理を行う。

```
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
  <ICtsPlugInVideoMixer>
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
[Required]
  CtsPlugInVideoMixer.h

[Interface ID]
  IID_ICtsPlugInVideoMixer

[Derived]
  ICtsPlugInRenderer

[method overview]
  - この interface は video を mixing する為の関数を含む。
  - この interface を継承することによって keyer や transition プラグイン
    を作成することができる。

----------------------------------------------------------------------
HRESULT ICtsPlugInVideoMixer::RenderBegin()
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  Parameter:
    none
  Return value:
    if this function succeeds, the return value is S_OK.
        if this function fails, returns an error.
  Notes:
    このプラグインインスタンスを使ってレンダリングを開始する時に呼び出される。
    plug-in は必要があれば、memory の確保、lookup table の生成などを行う
    rendering sequence の開始時に一回だけ呼び出される RenderBegin の後、
    0 回以上の Render method の呼び出しがあり、その後 RenderEnd が呼び出さ
    れる。
    ある instance について RenderBegin/RenderEnd を入れ子で呼び出すことは
    ない。

----------------------------------------------------------------------
HRESULT ICtsPlugInVideoMixer::Render(
    ICtsPix            *pPixResult,
    ICtsPix            *pPixFromOrBelow,
    ICtsPix            *pPixToOrAbove,
    RECT const          *prcRegion,
    DWORD const          dwTransparency,
    CtsFrame const    frmDuration,
    CtsFrame const    frmFrame,
    CtsField const    field,
    BOOL const           bScrub
    )
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  Parameter:
    [out] ICtsPix *pPixResult
      レンダリング結果を格納する為の ICtsPix interface ポインタ。
    [in]  ICtsPix *pPixFromOrBelow
      元画像
```

plug-in が transition の場合、From 画像を意味する。
plug-in が keyer や title mixer の場合、 Below 画像 を意味する。

[in] ICtsPix *pPixToOrAbove
元画像
plug-in が transition の場合, To 画像を意味する。
plug-in が keyer や title mixer の場合, Above 画像 を意味する。

[in] RECT const *prcRegion
plug-in が transition の場合、この parameter は無視できる。
plug-in が keyer や title mixer の場合、レンダリング対象の
矩形を意味する。

[in] DWORD const dwTransparency
plug-in が transition の場合, この parameter は無視できる。
plug-in が keyer や title mixer の場合、透明度を表す。

[in] CtsFrame const frmDuration
plug-in が関連付けられている TIme Line 上の Object の duration

[in] CtsFrame const frmFrame
レンダリングするフレーム番号、IN 点からの相対値で

[in] CtsField const field
field (Odd or Even)

[in] BOOL const bScrub
スクラブ中かどうかを表すフラグ。
TRUE ならスクラブ中。

Return value:
成功したら S_OK を返す。
失敗したらエラー値を返す。

Notes:
レンダリングを行う。pPixFromOrBelow と pPixToOrAbove は元画像への
interface ポインタ。レンダリング中、pPixFromOrBelow や pPixToOrAbove
の内容を変更してはならない。
pPixResult はレンダリング結果を格納するための interface ポインタ
だが, 画像イメージのデータ以外変更しては成らない。
画像イメージデータは ICtsPix::GetImageBuffer() を呼ぶことで取得する
ことができる。

Renderer は Odd フィールドと Even フィールドを非同期にレンダリング
する。Even フィールドのレンダリング中に Even フィールドを参照しなけ
ればならない場合は、plug-in 側で同期処理を行う必要がある。
Renderer はフレーム以上のずれが発生するような非同期処理を行わない。
Rendering はフィールド単位で行われることに注意すること。

HRESULT ICtsPlugInVideoMixer::RenderEnd()

Parameter:
none

Return value:
成功したら S_OK を返す。
失敗したらエラー値を返す。

Notes:
このインスタンスを使ってレンダリングをしなくなる時に呼び出される。
RenderBegin で確保したリソースを開放するなどの処理を行う。

=-=
<ICtsPlugInTitleMixer>
=-=

[Required]
  CtsPlugInTitleMixer.h

[Interface ID]
  IID_ICtsPlugInTitleMixer

[Derived]
  ICtsPlugInVideoMixer

[method overview]
  - この interface は title mixing に必要な関数が含まれる。
  - この interface を継承することで title mixer plug-in を作成することができる。

HRESULT ICtsPlugInTitleMixer::SetDirection(CtsFlags direction)

- -

  Parameter:
    CtsFlags direction
      title mixing を行う方向. IN 側か OUT 側かを表す。
      CtsPlugInTitleMixerIn or CtsPlugInTitleMixerOut.
  Return value:
    成功したら S_OK を返す。
    失敗したらエラー値を返す。
  Notes:
    この関数は rendering ツリーを構築するのに必要な CoreRenderer の作成時
    に呼び出される。この関数で指定された IN/OUT に従って Render 関数にて
    適切な rendering を行うこと。

=-=
<ICtsPlugInAudioFilter>
=-=

[Required]
CtsPlugInAudioFilter.h

[Interface ID]
IID_ICtsPlugInAudioFilter

[Derived]
ICtsPlugInRenderer

[method overview]
- この interface は audio を filtering する為に必要な関数を含む。
- この interface を使用することで audio filter を作成することができる。

HRESULT ICtsPlugInAudioFilter::RenderBegin()

- -

Parameter:
none

Return value:
成功したら S_OK を返す。
失敗したらエラー値を返す。

Notes:
このプラグインインスタンスを使ってレンダリングを開始する時に呼び出される。
plug-in は必要があれば、memory の確保、lookup table の生成などを行う。
rendering sequence の開始時に一回だけ呼び出される RenderBegin の後、
0 回以上の Render method の呼び出しがあり、その後 RenderEnd が呼び出される。
ある instance について RenderBegin/RenderEnd を入れ子で呼び出すことはない。

```
HRESULT ICtsPlugInAudioFilter::Render(
      ICtsSnd             *pSndResult,
      ICtsSnd             *pSndSource,
      CtsSample const     smpDuration,
      CtsSample const     smpFrame,
      BOOL const          bScrub
      )
```

- -

Parameter:

[out] ICtsSnd *pSndResult
レンダリング結果を格納する為の ICtsSnd interface ポインタ。

[in] ICtsSnd *pSndSource
元音声

[in] CtsSample smpDuration
plug-in が関連付けられている TIme Line 上の Object の duration。

[in] CtsSample smpFrame
レンダリングするフレーム番号．IN 点からの相対値。

[in] BOOL bScrub

スクラブ中かどうかを表すフラグ。
TRUE ならスクラブ中。

Return value:
成功したら S_OK を返す。
失敗したらエラー値を返す。

Notes:
レンダリングを行う。pSndSource は元音声への interface ポインタ。
レンダリング中、pSndSource の内容を変更してはならない。
pSndResult はレンダリング結果を格納するための interface ポインタ
だが、音声のデータ以外変更してはならない。
音声データは ICtsPix::GetSoundBuffer() を呼ぶことで取得することができる。

Renderer は Odd フィールドと Even フィールドを非同期にレンダリング
する。Even フィールドのレンダリング中に Even フィールドを参照しなけ
ればならない場合は、plug-in 側で同期処理を行う必要がある。
Renderer はフレーム以上のずれが発生するような非同期処理を行わない。
Rendering はフィールド単位で行われることに注意すること。

HRESULT ICtsPlugInAudioFilterr::RenderEnd()

- -

Parameter:
none

Return value:
成功したら S_OK を返す。
失敗したらエラー値を返す。

Notes:
このインスタンスを使ってレンダリングをしなくなる時に呼び出される。
RenderBegin で確保したリソースを開放するなどの処理を行う。

=-=
<ICtsPlugInAudioMixer>
=-=

[Required]
  CtsPlugInAudioMixer.h

[Interface ID]
  IID_ICtsPlugInAudioMixer

[Derived]
  ICtsPlugInRenderer

[method overview]
- この interface にはオーディオミキシングを行うための関数が含まれる。
- この interface を使用してクロスフェードプラグインを作成することができる。

HRESULT ICtsPlugInAudioMixer::RenderBegin()

- -

Parameter:
  none

Return value:
  成功したら S_OK を返す。
  失敗したらエラー値を返す。

Notes:
  このプラグインインスタンスを使ってレンダリングを開始する時に呼び出される。
  plug-in は必要があれば、memory の確保、lookup table の生成などを行う。
  rendering sequence の開始時に一回だけ呼び出される RenderBegin の後、
  0 回以上の Render method の呼び出しがあり、その後 RenderEnd が呼び出さ
  れる。
  ある instance について RenderBegin/RenderEnd を入れ子で呼び出すことは
  ない。

```
HRESULT ICtsPlugInAudioMixer::Render(
    ICtsSnd             *pSndResult,
    ICtsSnd             *pSndFrom,
    ICtsSnd             *pSndTo,
    CtsSample const     smpDuration,
    CtsSample const     smpSample,
    BOOL const          bScrub
    )
```

- -

Parameter:
  [out] ICtsSnd *pSndResult
    レンダリング結果を格納する為の ICtsSnd interface ポインタ。
  [in] ICtsSnd *pSndFrom
    元音声．From 側。
  [in] ICtsSnd *pSndTo
    元音声．To 側。
  [in] CtsSample smpDuration
    plug-in が関連付けられている TIme Line 上の Object の duration。

[in] CtsSample smpFrame
レンダリングするフレーム番号、IN 点からの相対値で。
[in] BOOL bScrub
スクラブ中かどうかを表すフラグ。
TRUE ならスクラブ中。

Return value:
成功したら S_OK を返す。
失敗したらエラー値を返す。

Notes:
レンダリングを行う。pSndFrom と pSndTo は元画像への interface ポインタ。
レンダリング中、pSndFrom や pSndTo の内容を変更してはならない。
pSndResult はレンダリング結果を格納するための interface ポインタ
だが、画像イメージのデータ以外変更してはならない。
画像イメージデータは ICtsPix::GetSoundBuffer() を呼ぶことで取得する
ことができる。

Renderer は Odd フィールドと Even フィールドを非同期にレンダリング
する。もし Even フィールドのレンダリング中に Even フィールドを参照
しなければいけないような場合は、plug-in 側で同期処理を行う必要が有る。
Renderer はフレーム以上のずれが発生するような非同期処理を行わない。
Rendering はフィールド単位で行われることに注意すること。

HRESULT ICtsPlugInAudioMixer::RenderEnd()

- -

Parameter:
none

Return value:
成功したら S_OK を返す。
失敗したらエラー値を返す。

Notes:
このインスタンスを使ってレンダリングをしなくなる時に呼び出される。
RenderBegin で確保したリソースを開放するなどの処理を行う。

=-=
 Importer interface
=-=

* Importer PlugIn は、主にファイルからの Video/Audio データの読み込みを担当するが、カラーバーや無音黒クリップなど、アプリケーション内部で生成するデータについても、同じインターフェイスを使用する。

* Importer I/F は、Importer に関する情報を管理する。ICtsPlugInImporterInfo, PlugIn の実体である ICtsPlugInImporter, Video データを取り出す口である ICtsPlugInVideoImporter, Audio 版の ICtsPlugInAudioImporter の 4 つ

* 実際のデータ取り出し処理は、Effect の PlugIn と同様に、ImportBegin(), フレーム単位の Import(), ImportEnd() が呼ばれるディスク I/O などの処理に時間がかかるのと、最適化できるようにするために、フレーム単位のデータの取り出し要求と、実際のデータの出力は非同期にできるようになっている。
　このため、Import() では、要求されたフレームのデータを取り出す要求を発行し、最終的なデータ結果が入るであろう領域のポインタを返しておく
　Impoter を利用する Renderer 側では、実際にデータが必要になった時点でこのとき返却されたポインタにアクセスし、データを取り出す。

* 他の種類の PlugIn は、情報を取得する口である PlugInInfo に対し、PlugInRenderer はタイムライン上のオブジェクトに対して一対一で生成される。つまり、同じ PlugInRenderer 実体が複数箇所で使用されることはない
　ところが、Importer の Renderer は、ファイルに対して一対一となり、タイムライン上のオブジェクトに対して一対一であることが保証されるわけではないことに注意。同じファイルがあちこちで使用されている場合は、同じ Importer Renderer が複数箇所で使用されることになる。
　このため、Import 処理に必要な中間データを Renderer 実体のメンバとして持つことができない場合がある。このときは、Begin でコンテキスト領域を確保して、ここにデータを格納する。

=-=

<ICtsPlugInImporterInfo>

=-=

[Required]
  CtsPlugInImporter.h

[Interface ID]
  IID_ICtsPlugInImporterInfo

[Derived]
  ICtsPlugInInfo

[method overview]
  - このインターフェースは、ファイルなどから video/audio を取り出すインポータの情報を取得するための関数を提供する。

CtsFlags ICtsPlugInImporterInfo::GetImporterCapability(void)

- -

Paramter:
  none
Return Value:
  Importer plug-in が持つ機能を表すフラグを返す。
  機能を表すフラグは CtsPlugInImporter.h に定義されている。
Description:
  Importer Plug-In がサポートする機能を取得する。
Remarks:
  PlugInInfo の情報であるため、Renderer ごとの動的な情報を格納することはできない。GetCapability は Renderer の情報であるため、実体ごとに変わる値を格納する場合には、こちらを使用する。
  静止画かどうか、アルファがあるかどうか、といった情報は GetCapability から取得する。

LPCTSTR ICtsPlugInImporterInfo::GetExtensions(int nIdex)

- -

Paramter:
  [in] int nIdex
    何番目の拡張子を取得したいかのインデックス。
Return Value:
  指定されたインデックスを持つ拡張子がなければ NULL を返す。
  それ以外の場合は拡張子の文字列ポインタを返す。
Description:
  読める可能性があるファイルの拡張子を 1 つ以上返す。
  NULL が出てくるまで順番に nIdx を上げていくと全ての拡張子が取り出せる。
Remarks:
  generator は拡張子を返してはいけない。

LPCTSTR ICtsPlugInImporterInfo::GetMenuItem(void)
- -
Paramter:
　なし。
Return Value:
　Setup を呼び出すメニュー用の項目名を返す。
Description:
　Setup を呼び出すメニュー用の項目名を返す。
Remarks:
　なし。

=-=
<ICtsPlugInImporter>
=-=

[Required]
  CtsPlugInImporter.h

[Interface ID]
  IID_ICtsPlugInImporter

[Derived]
  ICtsPlugInRenderer

[method overview]
  - このインターフェースは、インポータで主としてファイル関連を取り扱う関数を提供する。

HRESULT ICtsPlugInImporter::Open(LPCTSTR lpszFileName)
- -
Paramter:
  [in] LPCTRSTR lpszFileName
    オープンするファイル名。
Return Value:
  成功したら S_OK を返す.
  失敗したらエラー値を返す.
Descriptin:
  指定されたファイルを開く。
Remarks:
  ImporterCapability で CanOpen を返す場合はちゃんと実装すること。
  渡された lpszFileName は plug-in 内部で複製を作らなければならない。

HRESULT ICtsPlugInImporter::Create(void)
- -
Paramter:
  なし。
Return Value:
  成功したら S_OK を返す。
  失敗したらエラー値を返す。
Descriptin:
  新しい importer の実体を作成する。ImporterCapability で CanCreate を返す importer は実装しておく必要がある。
Remarks:
  Project 設定は PlugInManager の Suite から取得可能。

LPCTSTR ICtsPlugInImporter::GetFileName(void)
- -
Paramter:
  なし。
Return Value:
  ファイル名を返す。
Descriptin:
  Importer が開いているファイル名を返す。color clip などの generator は ColorClip など適当な名前の固定値を返せばよい。Generator でない場合は、適

切なファイル名を返す。

Remarks:
　主に bin のデフォルトの clip 名として使われる。

HRESULT ICtsPlugInImporter::Lock(void)
Paramter:
　なし。

Return Value:
　S_OK　　：ロック処理が正常に行えた。
　S_FALSE：ファイルが書き換わっているが VideoInfo や AudioInfo の内容
　　は変っていない場合。例えば、AVI で Codec もフレームサイズ
　　やフレームレートも変っていないが、タイムスタンプが変ってい
　　る場合。静止画でフレームサイズが変ってないときに返す。
　E_FAIL　：ファイルが書き換わっているが、ファイルがなくなっているときに返す。

Descriptin:
　Unlock()で解放されたファイルハンドルを再度開く。

　アプリケーションは、Open するファイルを基本的に排他処理し開く。
ただし、自アプリケーションがアクティブでない場合、他のアプリケーション
でそのファイルを編集する可能性があるので、アクティブでなくなった場合に
は Unlock() してファイルを解放する。
　再び自分がアクティブになった場合には、Lock() を呼んで再度ファイルを
排他モードにする。
　Unlock()されている間にファイルが書き換わる恐れがあるので、タイムス
タンプや、ファイルサイズなどを覚えておき、書き換わっていた場合は、Lock
を失敗させてアプリケーションに通知する。

Remarks:
　Unlock() 参照。

HRESULT ICtsPlugInImporter::Unlock(void)
- -
Paramter:
　なし。
Return Value:
　成功したら S_OK を返す.
　失敗したらエラー値を返す.
Descriptin:
　一時的にファイルハンドルを解放する。再度 Lock する時の為に、最終状態
のファイル･サイズやタイムスタンプを記録しておく。
Remarks:
　Lock() 参照。

HRESULT ICtsPlugInImporter::HasVideo(void)
Paramter:
　なし。
Return Value:
　Video データが読み出せる場合には S_OK
　だめな場合は E_FAIL

Descriptin:
　Video データが読み出せるかどうかを返す。具体的には、QueryInterface で ICtsPlugInVideoImporter が取り出せるときは S_OK, そうでなければ E_FAIL を返す。
Remarks:
　なし。

HRESULT ICtsPlugInImporter::HasAudio(void)
- -
Paramter:
　なし。
Return Value:
　Audio データが読み出せる場合には S_OK を返す。
　だめな場合は E_FAIL を返す。
Descriptin:
　Audio データが読み出せるかどうかを返す。具体的には、QueryInterface で ICtsPlugInAudioImporter が取り出せるときは S_OK, そうでなければ E_FAIL を返す。
Remarks:
　なし。

=-=
<ICtsPlugInVideoImporter>
=-=

[Required]
  CtsPlugInImporter.h

[Interface ID]
  IID_ICtsPlugInVideoImporter

[Derived]
  IUnknown

[method overview]
- このインターフェースは、ファイルなどから Video を取り出す関数を提供する。
- このインターフェースを継承して EDIUS に Video を供給することができる。

HRESULT ICtsPlugInVideoImporter::GetVideoInfo(ICtsVideoInfo **ppVideoInfo)
- -

Paramter:
  [out] ICtsVideoInfo **ppVideoInfo
    VideoInfo を受け取るためのポインタ。
Return Value:
  成功したら S_OK を返す。
  失敗したらエラー値を返す。
Descriptin:
  供給される Video データの情報を ICtsVideoInfo の形で取得する。
ICtsVideoInfo については、CtsVideoInfo.h を参照のこと。
Remarks:
  参照カウンタが 1 上がっているので、不要になったら解放すること。

```
HRESULT ICtsPlugInVideoImporter::ImportVideoBegin(
        ICtsCoreIoScheduler *ios,
        ICtsCoreCodecScheduler *cs,
        CtsFrame duration,
        PVOID *ppvImportContext);
```
- -

Paramter:
  [in] ICtsCoreIoScheduler *ios
    ディスクの読み出しのスケジューリングが必要な場合、このポインタを使ってスケジューラに依頼する。

  [in] ICtsCoreCodecScheduler *cs
    コーデックのスケジューリングが必要な場合、このポインタを使ってスケジューラに依頼する。

  [in] CtsFrame duration
    読み出す予定のクリップの長さをフレームで示す。

  [out] PVOID *ppvImportContext
    一回の Timeline Object 処理に固有のデータ領域が必要な場合、ここで
確保して返す。以降の ImportVideo と ImportVideoEnd ではここで返した値
が渡される。

Return Value:
　成功したら S_OK を返す。
　失敗したらエラー値を返す。

Descriptin:
　Importer からのイメージ取り出し処理を開始する。
　一つの Timeline Object に対し、一組の ImportVideoBegin - ImportVideo(複数回) - ImportVideoEnd が呼ばれる。
　メモリ領域節約のため、ファイル内容を解析して保持しておく場合など、メモリの必要な処理はなるべく Open ではなく、ImportVideoBegin で行うこと。

Remarks:
　他のタイプの PlugInRenderer は、Timeline Object に対してかならず一つの実体があるが、Importer はファイルに対して一つであり、同じファイルからのクリップが複数箇所に置かれた場合でも Importer の実体は一つになる。
　Timeline Object 単位でデータを保持する必要がある場合には、メンバ変数ではなく、引数の context を利用する。

```
HRESULT ICtsPlugInVideoImporter::ImportVideo(
        ICtsCoreIoScheduler *ios,
        ICtsCoreCodecScheduler *cs,
        FOURCC fccCompression,
        CtsFrame frame,
        PVOID *ppvImportContext,
        ICtsPlugInAsyncPix **ppAsync);
```
- -
Paramter:
　[in] ICtsCoreIoScheduler *ios
　　ディスクの読み出しのスケジューリングが必要な場合、このポインタを使ってスケジューラに依頼する。

　[in] ICtsCoreCodecScheduler *cs
　　コーデックのスケジューリングが必要な場合、このポインタを使ってスケジューラに依頼する。

　[in] FOURCC fccCompression
　　今から取り出したい fcc。通常は FCC('YUY2')。

　[in] CtsFrame frame
　　処理対象のフレーム番号。

　[in] PVOID *ppvImportContext
　　ImportVideoBegin で返したポインタがそのまま渡ってくる。

　[out] ICtsPlugInAsyncPix **ppAsync
　　実際に Video データが格納される領域。

Return Value:
　成功したら S_OK を返す。
　失敗したらエラー値を返す。

Descriptin:
　指定された位置の 1 フレーム分のデータを用意する処理を行う。非同期処理を可能にするため、実際の video data の取り出しは、絵が必要になる直前のタイミングで ppAsync から取り出される。ImportVideo では、それまでに絵が準備されるように、各スケジューラに依頼を行う処理をする。
　スケジューラを利用しない場合には、ここで実際の video data を AsyncPix に準備しても良い。

Remarks:
　ImportVideoBegin() 参照。

HRESULT ICtsPlugInVideoImporter::ImportVideoEnd(PVOID *ppvImportContext)
- -
Paramter:
　[in] PVOID *ppvImportContext
　　ImportVideoBegin で返したポインタがそのまま渡ってくる。

Return Value:
　成功したら S_OK を返す。
　失敗したらエラー値を返す。

Descriptin:
　Import Video 関連の終了処理を行う。ImportVideoBegin で確保した資源の解放、ppImportContext 領域の解放など。

Remarks:
　ImportVideoBegin() 参照。

=-=
<ICtsPlugInAudioImporter>
=-=

[Required]
  CtsPlugInImporter.h

[Interface ID]
  IID_ICtsPlugInAudioImporter

[Derived]
  IUnknown

[method overview]
- このインターフェースは、ファイルなどから Audio を取り出す関数を提供する。
- このインターフェースを継承して EDIUS に Audio を供給することができる。

HRESULT ICtsPlugInAudioImporter::GetAudioInfo(ICtsAudioInfo **ppAudioInfo)

- -

Paramter:
  [out] ICtsAudioInfo **ppAudioInfo
    AudioInfo を受け取るためのポインタ。
Return Value:
  成功したら S_OK を返す。
  失敗したらエラー値を返す。
Descriptin:
  供給される Audio データの情報を ICtsAudioInfo の形で取得する。
ICtsAudioInfo については、CtsAudioInfo.h を参照のこと。
Remarks:
  参照カウンタが 1 上がっているので、不要になったら解放すること。

```
HRESULT ICtsPlugInAudioImporter::ImportAudioBegin(
        ICtsCoreIoScheduler *ios,
        ICtsCoreCodecScheduler *cs,
        CtsSample duration,
        PVOID *ppvImportContext);
```

- -

Paramter:
  [in] ICtsCoreIoScheduler *ios
    ディスクの読み出しのスケジューリングが必要な場合、このポインタを使ってスケジューラに依頼する。

  [in] ICtsCoreCodecScheduler *cs
    コーデックのスケジューリングが必要な場合、このポインタを使ってスケジューラに依頼する。

  [in] CtsSample duration
    読み出す予定の音の長さをサンプル数で示す。

  [out] PVOID *ppvImportContext
    一回の Timeline Object 処理に固有のデータ領域が必要な場合、ここで

確保して返す。以降の ImportAudio と ImportAudioEnd ではここで返した値が渡される。

Return Value:
　成功したら S_OK を返す。
　失敗したらエラー値を返す。

Descriptin:
　Importer からのイメージ取り出し処理を開始する。
　一つの Timeline Object に対し、一組の ImportAudioBegin - ImportAudio(複数回) - ImportAudioEnd が呼ばれる。
　ファイル内容を解析して保持しておく場合など、メモリの必要な処理はなるべく Open ではなく、ImportAudioBegin で行うこと。

Remarks:
　ImportVideoBegin 参照。

```
HRESULT ICtsPlugInAudioImporter::ImportAudio(
        ICtsCoreIoScheduler *ios,
        ICtsCoreCodecScheduler *cs,
        FOURCC fccCompression,
        CtsSample sample,
        CtsSample NumberOfSamples,
        PVOID *ppvImportContext,
        ICtsPlugInAsyncSnd **ppAsync);
```

- -

Paramter:
　[in] ICtsCoreIoScheduler *ios
　　ディスクの読み出しのスケジューリングが必要な場合、このポインタを使ってスケジューラに依頼する。

　[in] ICtsCoreCodecScheduler *cs
　　コーデックのスケジューリングが必要な場合、このポインタを使ってスケジューラに依頼する。

　[in] FOURCC fccCompression
　　今から取り出したい音の形式を示す値。通常は 0(PCM)。

　[in] CtsSample sample
　　今から取り出したい Audio の位置を sample で指定する。

　[in] CtsSample NumberOfSamples
　　今から取り出したい Audio の長さを sample で指定する。

　[in] PVOID *ppvImportContext
　　ImportAudioBegin で返したポインタがそのまま渡ってくる。

　[out] ICtsPlugInAsyncPix **ppAsync
　　実際に Audio データが格納される領域。

Return Value:

成功したら S_OK を返す。
失敗したらエラー値を返す。

Descriptin:
指定された位置からしていの長さのデータを用意する処理を行う。非同期処理を可能にするため、実際のデータの取り出しは、音が必要になる直前のタイミングで ppAsync から取り出される。ImportAudio では、それまでに音が準備されるように、各スケジューラに依頼を行う処理をする。
スケジューラを利用しない場合には、ここで実際の Audio data を AsyncSnd に準備しても良い。

Remarks:
ImportVideoBegin 参照。

HRESULT ICtsPlugInAudioImporter::ImportAudioEnd(PVOID *ppvImportContext)
- -
Paramter:
[in] PVOID *ppvImportContext
ImportAudioBegin で返したポインタがそのまま渡ってくる。

Return Value:
成功したら S_OK を返す。
失敗したらエラー値を返す。

Descriptin:
Import Audio 関連の終了処理を行う。ImportAudioBegin で確保した資源の解放、ppImportContext 領域の解放を行う。

Remarks:
ImportVideoBegin 参照。

=-=
Exporter interface
=-=

* Exporter PlugIn は、ファイルへのデータ出力を担当する。Clip Rendering などの内部処理にもこの I/F を利用する。

* Exporter I/F は、Importer と同様、Exporter に関する情報を管理する ICtsPlugInExporterInfo, PlugIn である ICtsPlugInExporter とに分かれる。

* Export を行う際は、タイムラインデータを取り出すために、内部でダミーの Hal を作成し、新しい Playbacker を作成して playback を行う。Exporter はこのダミー Hal から呼び出される。ダミー Hal は、Exporter に対して実データを取り出す I/F を提供するとともに、データの準備が整った段階で、順次 ExportBegin, Export, ExportEnd を呼び出す。

* Exporter Plugin は、他の PlugIn とは異なり、Export 関数内部でフレームごとの処理を行う。つまり、フレームごとに Export() が呼び出されるのではなく、Export 内部でフレーム単位の処理をループさせることに注意しなければならない。

* Exporter 内部で、タイムラインに関する情報が必要な場合は、Suite 経由で取得する。
例: マーカ位置など。

=-=
<ICtsPlugInExporterInfo>
=-=

[Required]
CtsPlugInExporter.h

[Interface ID]
IID_ICtsPlugInExporterInfo

[Derived]
ICtsPlugInInfo

[method overview]
- このインターフェースは、エクスポータの情報を取得するための関数を提供する。

CtsFlags ICtsPlugInExporterInfo::GetExporterCapability(void)
- -

Paramter:
なし。

Return Value:
Exporter plug-in が持つ機能を表すフラグを返す。
機能を表すフラグは CtsPlugInExporter.h に定義されている。

Description:
Exporter Plug-In がサポートする機能を取得する。

Remarks:
PlugInInfo の情報であるため、Renderer ごとの動的な情報を格納することはできない。

LPCTSTR ICtsPlugInExporterInfo::GetExtensions()
- -

Paramter:
なし。

Return Value:
その Exporter で出力する標準の拡張子を返す。

Description:
その Exporter で出力する標準の拡張子を返す。

Remarks:
なし。

LPCTSTR ICtsPlugInExporterInfo::GetMenuItem(void)

Paramter:
なし。

Return Value:

Setup を呼び出すメニュー用の項目名を返す。

Description:

Setup を呼び出すメニュー用の項目名を返す。

Remarks:

なし。

=-=
<ICtsPlugInExporter>
=-=
[Required]
  CtsPlugInExporter.h

[Interface ID]
  IID_ICtsPlugInExporter

[Derived]
  ICtsPlugInRenderer

[method overview]
  - このインターフェースは、タイムライン・データをファイル出力するための関数を定義する。

```
HRESULT ICtsPlugInExporter::ExportBegin(
                        CtsFrame duration,
                        ICtsExporterData *pED);
```
- -
Paramter:
  [in] CtsFrame duration
    出力予定のタイムライン・データの長さをフレーム単位で指定する。
  [in] ICtsExporterData *pED
    タイムライン･データを取り出す I/F へのポインタ。
Return Value:
  成功したら S_OK を返す。
  失敗したらエラー値を返す。
Descriptin:
  Export 開始処理を行う。通常は、出力対象のファイルをオープンするなどの作業を行う。
Remarks:
  なし。

```
HRESULT ICtsPlugInExporter::Export(void)
```
- -
Paramter:
  なし。
Return Value:
  成功したら S_OK を返す。
  失敗した場合はエラー値を返す。
Descriptin:
  Export 処理を行う。ExportBegin で渡された I/F を指定された回数分呼び出して、Video/Audio データを取得し、必要ならばコンバートしてファイルに書き出す。
  エラーが発生した場合には上位にエラーを返す。エラーになった場合でも ExportEnd は呼び出されるので、ファイルのクローズなどの処理は ExportEnd で行うことができる。
Remarks:
  ICtsExporterData の GetVideoData, GetAudioData は、Timeline 上の In 点から Out 点まで(存在しない場合には最初から最後まで)をフレーム単位で

返してくる。バッファリングや、フレームのスキップなどが必要な場合には、
Exporter 側でこれらの処理を行う。
　またユーザーからのキャンセルがあった場合には GetVideoData/GetAudioData
からエラーが返るので、その場合には Export 処理を中断して上位に帰る。

HRESULT ICtsPlugInExporter::ExportEnd(void)
- -
Paramter:
　なし。
Return Value:
　成功したら S_OK を返す.
　失敗したらエラー値を返す.
Descriptin:
　Export 終了処理を行う。出力対象のファイルのクローズなど。
Remarks:
　なし。

HRESULT ICtsPlugInExporter::SetFileName(LPCTSTR lpszFileName)
- -
Paramter:
　[in] LPCTSTR lpszFileName
　　出力ファイル名を指定する。
Return Value:
　成功したら S_OK を返す。
　失敗したらエラー値を返す。
Descriptin:
　出力対象となるファイル名をセットする。Setup ダイアログを開かずにファ
イル出力したい場合に使用する。

Remarks:
　主に Clip Rendering 等の内部処理に使う予定。

LPCTSTR ICtsPlugInExporter::GetFileName(void)
- -
Paramter:
　なし。
Return Value:
　ファイル名を返す。
Descriptin:
　Exporter が出力するファイル名を返す。Setup Dialog か、SetFileName か
どちらかで設定された名前がかえる。
Remarks:
　なし。

HRESULT ICtsPlugInExporter::GetVideoInfo(ICtsVideoInfo **ppVideoInfo)
- -
Paramter:
　[out] ICtsVideoInfo **ppVideoInfo
　　出力するタイムラインのプロジェクト設定の VideoInfo を返す。
Return Value:
　成功したら S_OK を返す。
　失敗したらエラー値を返す。

Descriptin:
　出力するタイムラインのプロジェクト設定の VideoInfo を返す。この設定を使ってダミーの playback が行われる。
Remarks:
　なし。

HRESULT ICtsPlugInExporter::GetAudioInfo(ICtsAudioInfo **ppAudioInfo)
- -
Paramter:
　[out] ICtsAudioInfo **ppAudioInfo
　　出力するタイムラインのプロジェクト設定の AudioInfo を返す。
Return Value:
　成功したら S_OK を返す。
　失敗したらエラー値を返す。
Descriptin:
　出力するタイムラインのプロジェクト設定の AudioInfo を返す。
Remarks:
　なし。

=-=
<ICtsExporterData>
=-=

[Required]
CtsPlugInExporter.h

[Interface ID]
IID_ICtsExporterData

[Derived]
IUnknown

[method overview]
- このインターフェースは、Exporter にタイムライン･データを供給するための関数を定義する。

HRESULT ICtsExporterData::GetVideoData(ICtsPix **ppPix)

Paramter:
[out] ICtsPix **ppPix
出力するタイムラインの Video データを返す。

Return Value:
成功したら S_OK を返す。
失敗したらエラー値を返す。

Descriptin:
出力するタイムラインの Video データを返す。タイムライン上に In 点、Out 点が設定されている場合はその間を返し、ない場合は先頭から最後までのデータを返す。ExportBegin 時にリセットされ、順次次のフレームを返す。
ユーザが処理を中断した場合や、何らかのエラーが発生して処理を中断したい場合には、エラーを返す。

Remarks:
キャンセル時、エラー時でも ExportEnd の呼び出しは行える。

HRESULT ICtsExporterData::GetAudioData(ICtsSnd **ppSnd)

Paramter:
[out] ICtsSnd **ppSnd
出力するタイムラインの Audio データを返す。

Return Value:
成功したら S_OK を返す。
失敗したらエラー値を返す。

Descriptin:
出力するタイムラインの Audio データを 1 フレーム分返す。タイムライン上に In 点、Out 点が設定されている場合はその間を、ない場合は先頭から最後までのデータを順次返す。
ユーザが処理を中断した場合や、何らかのエラーが発生して処理を中断したい場合には、エラーを返す。

Remarks:
なし。

=-=

<ICtsPlugInSyncObject>

=-=

[Required]
  CtsPlugInSyncObject.h

[Interface ID]
  IID_ICtsPlugInSyncObject

[Derived]
  IUnknown

[Remarks]
- plug-in の Render 処理の中で ODD field と EVEN field の同期を取る為に使用できる同期オブジェクトインターフェース。
- plug-in の Render() 関数の中でのみ使用することができる。

HRESULT ICtsPlugInSyncObject::Sync(CtsField field)

Parameter:
  [in] CtsField field
    同期を取ろうとしている rendering thread の Field

Return Value:
  Odd Field と Even Field の同期がとれた時，S_OK を返す。
  失敗したらエラー値を返す。

Remarks:
  この関数を plug-in の Render() 内で呼び出した箇所で Odd と Even の同期を取ることができる。Render() 関数の中で任意の回数呼び出すことができる.

```
(example)
CCtsXXXXTransition::Render(
    ...
    CtsField     field,
    ...
    )
{

    m_pSyncObject->Sync(field);         // 1st synchronous point
                                        // The thread is the wait state till the sync object
                                        // coonfirms that the ODD/EVEN rendering thread is synchronized.
    .....
    .....
    .....

    m_pSyncObject->Sync(field);         // 2nd synchronous point

    .....
    .....

    m_pSyncObject->Sync(field);         // 3rd synchronous point
```

```
        .....
        .....

        return S_OK;
    }
-----------------------------------------------------------------
```

==
● Plug-In Manager interface ●
==

=-=
<ICtsPlugInServiceFactory>
=-=
[Required]
CtsPlugInServiceFactory.h

[Interface ID]
IID_ICtsPlugInServiceFactory

[Derived]
IUnknown

[Remarks]
- plug-in DLL で export されている Startup/Shutdown 関数の引数で
  取得できる interface
- この interace に対して QueryInterface することで他のインターフェース
  を引き出すことができる。(ICtsPlugInRegisterer, ICtsPlugInNotifier)
- この interface は特別な interface を持たない。plug-in や plug-in DLL
  が使う interface を引き出す目的の為だけの interface である。

```
HRESULT ICtsPlugInServiceFactory::CreateSuite(
    ICtsPlugInInfo    *pTHIS,
    DWORD             objectID,
    REFIID            suiteIID
    void              **ppvSuite)
```
Paramter:
[in] ICtsPlugInInfo *pTHIS
  Plug-In の Info インターフェースポインタ。
[in] DWORD objectID
  Plug-In が関連づけられている objectID を指定. objectID は CreateInstance
  が呼び出される時に与えられるものである。
[in] REFIID suiteIID
  所望の Suite interface for Plug-In の IID を指定。
[out] void **ppvSuite
  生成された Suite interface のポインタが格納される。
Return Value:
  成功したら S_OK を返す。
  失敗したらエラー値を返す。
Remarks:
  所望の Suite for Plug-In を作成し、その interface ポインタを取得する。
  suiteIID で指定した IID を持つ Suite の interface ポインタが ppvSuite に格納される。
  Suite の作成に失敗した場合、*ppvSuite には NULL が代入され、エラー値が返る。

HRESULT ICtsPlugInServiceFactory::CreateSyncObject(ICtsPlugInSyncObject **pSyncObject)
Paramter:
[out] ICtsPlugInSyncObject **pSyncObject

生成された同期オブジェクトの interface が格納される。

Return Value:

成功したら S_OK を返す。

失敗したらエラー値を返す。

Remarks:

同期オブジェクトを生成する。取得した interface は必要が無くなれば必ず解放すること。

同期オブジェクトの interface については <ICtsPlugInSyncObject> の項目を参照のこと。

```
HRESULT ICtsPlugInServiceFactory::CreateNullClipImporter(
        ICtsPlugInRenderer *pTHIS,
        ICtsPlugInRenderer** ppNullClipImporter
        );
```

Parameter:

[in] ICtsPlugInRenderer *pTHIS

plug-in のインターフェースポインタ。

[out] ICtsPlugInRenderer** ppNullClipImporter

NULL clip importer へのポインタ。

Return Value:

成功したら S_OK を返す。

失敗したらエラー値を返す。

Remarks:

NULL clip importer を取得する。

=-=

<ICtsPlugInServiceFactory15>

=-=

[Required]
  CtsPlugInServiceFactory15.h

[Interface ID]
  IID_ICtsPlugInServiceFactory

[Derived]
  ICtsPlugInServiceFactory

[Remarks]
  この interface は現在使用していない。

HRESULT ICtsPlugInRegisterer::RegisterPlugIn(ICtsPlugInInfo *pTHIS)
Parameter:
  [in] ICtsPlugInInfo * pTHIS
    Plug-In Info interface へのポインタ
Return Value:
  成功したら S_OK を返す。
  失敗したらエラー値を返す。
Remarks:
  Plug-In manager へ Plug-In を登録する。

HRESULT ICtsPlugInServiceFactory15::CreateMemory(ICtsMemory **ppMemory)

- -

[parameter]
  ICtsMemory **ppMemory
    新しいメモリインターフェースへ格納するためのポインタ
    *ppMemory は NULL になるよう初期化されなければならない。

[return value]
  成功したら S_OK を返す。
  失敗したらエラー値を返す。

[note]
  新しいメモリインターフェースを取得する。.
  この関数は ICtsPlugInRenderer::CreatePackedParameter() で使用することができる。

HRESULT ICtsPlugInServiceFactory15::CreateVideoInfo(ICtsVideoInfo **ppVideoInfo);

- -

[parameter]
  ICtsVideoInfo **ppVideoInfo
    new video interface へ格納するためのポインタ
    *ppVideoInfo は NULL になるよう初期化されなければならない。

[return value]
成功したら S_OK を返す。
失敗したらエラー値を返す。

[note]
new video information interface を生成するためのポインタ.
必要なくなった場合はこのインターフェースは解放する必要がある。

HRESULT ICtsPlugInServiceFactory15::CreateAudioInfo(ICtsAudioInfo **ppAudioInfo);
- -
[parameter]
ICtsAudioInfo **ppAudioInfo
new audio interface へ格納するためのポインタ
*ppAudioInfo は NULL になるよう初期化されなければならない。

[return value]
成功したら S_OK を返す。
失敗したらエラー値を返す。

[note]
new audio information interface を生成するためのポインタ
必要なくなった場合はこのインターフェースは解放する必要がある。

HRESULT ICtsPlugInServiceFactory15::CreatePix(ICtsVideoInfo *pVideoInfo, CtsPixFlag pixflag, ICtsPix **ppPix);
- -
[parameter]
ICtsVideoInfo *pVideoInfo
pix parameter を詳細定義するための video information interface
例： frame size, pixel depth 等

CtsPixFlag pixflag
pix 生成時にフラグを定義する（CtsPix.h も参照すること）

CtsPix **ppPix
new pix interface へ格納するためのポインタ
*ppPix は NULL になるよう初期化されなければならない。

[return value]
成功したら S_OK を返す。
失敗したらエラー値を返す。

[note]
新しい pix を生成する。
この関数によりインターフェースを解放する必要がある。
例)
{
CComPtr<ICtsPix> pPix = NULL;
pServiceFactory15->CreatePix(vinfo, flag, &pPix);

```
        }                   // <---- auto release in going out of scope
```

HRESULT ICtsPlugInServiceFactory15::CreateSnd(ICtsAudioInfo *pAudioInfo, CtsSample sample, ICtsSnd **ppSnd);

- -

[parameter]

ICtsAudioInfo *pAudioInfo

pix parameter を詳細定義するための video information interface

例： frame size, pixel depth 等

CtsSample sample

New snd のサンプル数を定義する。

ICtsSnd **ppSnd

new snd interface へ格納するためのポインタ

*ppSnd は NULL になるよう初期化されなければならない。

[return value]

成功したら S_OK を返す。

失敗したらエラー値を返す。

[note]

new snd を生成する。

この関数の取得によりインターフェースを解放する必要がある。

例)

```
        {
            CComPtr<ICtsSnd> pSnd = NULL;
            pServiceFactory15->CreateSnd(vinfo, flag, &pSnd);
        }               // <---- auto release in going out of scope
```

HRESULT ICtsPlugInServiceFactory15::CreatePixOperator(ICtsCorePixOperator **ppPixOperator);

- -

[parameter]

ICtsCorePixOperator **ppPixOperator

new core pix operator interface へ格納するためのポインタ

*ppPixOperator は NULL になるよう初期化されなければならない。

[return value]

成功したら S_OK を返す。

失敗したらエラー値を返す。

[note]

YUY2 から RGB への変換等を行う core pix operater を生成する。

このインターフェースを使用する際は CtsCorePixOperator.h をインクルードする必要がある。

HRESULT ICtsPlugInServiceFactory15::CreateTimeLineSuite(ICtsPlugInRenderer *pTHIS, DWORD

dwObjectID, ICtsPlugInTimeLineSuite **ppTimeLineSuite);
- -
[parameter]
ICtsPlugInRenderer *pTHIS
Plug-in 自体のポインタを定義する。

DWORD dwObjectID
ICtsPlugInInfo::CreateInstance() 関数を呼び出すための object ID を定義する。

ICtsPlugInTimeLineSuite **ppTimeLineSuite
new time line suite interface へ格納する（CtsPlugInTimeLineSuite.h も参照のこと)。
*ppTimeLineSuite は NULL. になるよう初期化されなければならない。

[return value]
成功したら S_OK を返す。
失敗したらエラー値を返す。

[note]
new time line suite interface を生成する。
この関数の取得によりインターフェースを解放する必要がある。

HRESULT ICtsPlugInServiceFactory15::CreateObjectSuite(ICtsPlugInRenderer *pTHIS, DWORD dwObjectID, ICtsPlugInObjectSuite **ppObjectSuite);
- -
[parameter]
ICtsPlugInRenderer *pTHIS
plug-in 自体のポインタを定義する。

DWORD dwObjectID
ICtsPlugInInfo::CreateInstance() 関数を呼び出すための object ID を定義する。

ICtsPlugInObjectSuite **ppObjectSuite
new object suite interface へ格納する（CtsPlugInObjeceSuite.h も参照のこと）
*ppObjectSuite は NULL になるよう初期化しなければならない。

[return value]
成功したら S_OK を返す。
失敗したらエラー値を返す。

[note]
new object suite interface を生成する。
この関数の取得によりインターフェースを解放する必要がある。

HRESULT ICtsPlugInServiceFactory15::CreatePlatformSuite(ICtsPlugInPlatformSuite **ppPlatformSuite);
- -
[parameter]
ICtsPlugInPlatformSuite **ppPlatformSuite
new platform suite interface へ格納する（CtsPlugInPlatformSuite.h も参照のこと）

*ppPlatformSuite は NULL になるよう初期化されなければならない。.

[return value]
成功したら S_OK を返す。
失敗したらエラー値を返す。

[note]
new platform suite interface を生成する。
この関数の取得によりインターフェースを解放する必要がある。

HRESULT ICtsPlugInServiceFactory15::CreateProjectSettingSuite(ICtsPlugInProjectSettingSuite **ppProjectSettingSuite);

- -

[parameter]
ICtsPlugInProjectSettingSuite **ppProjectSettingSuite
new project setting suite interface へ格納する（CtsPlugInProjectSettingSuite.h も参照のこと）
*ppProjectSettingSuite は NULL になるよう初期化されなければならない。

[return value]
成功したら S_OK を返す。
失敗したらエラー値を返す。

[note]
new project setting suite interface を生成する。
この関数の取得によりインターフェースを解放する必要がある。

HRESULT ICtsPlugInServiceFactory15::CreateColorSelectorSuite(ICtsPlugInColorSelectorSuite **ppColorSelectorSuite);

- -

[parameter]
ICtsPlugInColorSelectorSuite **ppColorSelectorSuite
new color selector suite interface へ格納する（CtsPlugInColorSelectorSuite.h も参照のこと）
*ppColorSelectorSuite は NULL. になるよう初期化されなければならない。

[return value]
成功したら S_OK を返す。
失敗したらエラー値を返す。

[note]
new color selector suite interface を生成する。
この関数の取得によりインターフェースを解放する必要がある。

HRESULT ICtsPlugInServiceFactory15::CreateMasterPresetDBSuite(ICtsPlugInMasterPresetDBSuite **ppMasterPresetDBSuite);

- -

[parameter]

[return value]
[note]
　　この関数は現在使用していない。

HRESULT ICtsPlugInServiceFactory15::CreatePresetWrapper(CtsPlugInType type, DWORD dwObjectID, ICtsPlugInRenderer **ppPiRenderer);
- -
[parameter]
[return value]
[note]
　　この関数は現在使用していない。

HRESULT ICtsPlugInServiceFactory15::CreatePreset(ICtsPreset **ppPreset);
- -
[parameter]
[return value]
[note]
　　この関数は現在使用していない。

HRESULT ICtsPlugInServiceFactory15::CreatePresetNestedMgr(ICtsPlugInRenderer *pTHIS, DWORD dwObjectID, ICtsPresetNestedMgr **ppPresetNestedMgr, CtsFlags capabilities);
- -
[parameter]
[return value]
[note]
　　この関数は現在使用していない。

=-=
<ICtsPlugInRegisterer>
=-=

[Required]
CtsPlugInRegisterer.h

[Interface ID]
IID_ICtsPlugInRegisterer

[Derived]
IUnknown

[Remarks]
- Plug-In DLL が Plug-In Manager に Plug-In を登録/削除する時に使う Interface。
- Plug-In DLL の Startup で引数に渡される ICtsPlugInServiceFactory の interface pointer で IID に IID_ICtsPlugInRegisterer を指定して QueryInterface することによって取得できる。
- 必要がなくなれば 解放（Release）しなければならない。

HRESULT ICtsPlugInRegisterer::RegisterPlugIn(ICtsPlugInInfo *pTHIS)

Paramter:
[in] ICTSCRPlugInInfo * pTHIS
Plug-In の Info インターフェースポインタ。

Return Value:
成功したら S_OK を返す。
失敗したらエラー値を返す。

Remarks:
Plug-In Manager に Plug In を登録する。

HRESULT ICtsPlugInRegisterer::UnRegisterPlugIn(ICtsPlugInInfo *pTHIS)

Paramter:
[in] ICTSCRPlugInInfo * pTHIS
Plug-In の Info インターフェースポインタ。

Return Value:
成功したら S_OK を返す。
失敗したらエラー値を返す。

Remarks:
Plug-In Manager に登録されている Plug In を削除する。

=-=
<ICtsPlugInNotifier>
=-=
[Required]
  CtsPlugInNotifier.h

[Interface ID]
  IID_ICtsPlugInNotifier

[Derived]
  IUnknown

[Remarks]
  - Plug-In が Plug-In Manager に Notification する為に使う Interface。
  - Plug-In DLL の Startup で引数に渡される ICtsPlugInServiceFactory の interface pointer で IID に IID_ICtsPlugInNotifier を指定して QueryInterface することによって取得できる。
  - 必要がなくなれば解放（Release） しなければならない。

HRESULT ICtsPlugInNotifier::NotifyChangeParameter(ICtsPlugInRenderer *pTHIS)

Paramter:
  [in] ICtsPlugInRenderer *pTHIS
    Plug-In の インターフェースポインタ。

Return Value:
    成功したら S_OK を返す。
    失敗したらエラー値を返す。

Remarks:
  SetupDialog 等で Plug-In の Paramter 値が変更されたことを Plug-In Manager に通知する。

HRESULT ICtsPlugInNotifier::NotifyCloseSetup(ICtsPlugInRenderer *pTHIS, BOOL bCloseFlag)

Paramter:
  [in] ICtsPlugInRenderer *pTHIS
    Plug-In の インターフェースポインタ。
  [in] BOOL bCloseFlag
    setup dialog が [Ok] ボタンのクリックによって閉じようとしているのか
    [Cancel] ボタンのクリックによって閉じようとしているのかを示す。
    [Ok]ボタンによる場合は TRUE を指定。
    [Cancel]ボタンによる場合は FLASE を指定。

Return Value:
    成功したら S_OK を返す。
    失敗したらエラー値を返す。

Remarks:
  Plug-In の SetupDialog が閉じようとしていることを Plug-In Manager に通知する。

HRESULT ICtsPlugInNotifier::NotifyUpdateNestedSetup(
        ICtsPlugInRenderer *pTHIS,
        ICtsPlugInRenderer *pNestedPlugIn

);
Remarks;
現在は使用していない。

HRESULT ICtsPlugInNotifier::NotifyOpenNestedSetup(
ICtsPlugInRenderer *pTHIS,
ICtsPlugInRenderer *pNestedPlugIn,
HWND hParentWnd
);
Remarks:
現在は使用していない。

HRESULT ICtsPlugInNotifier::NotifyCloseNestedSetup(
ICtsPlugInRenderer *pTHIS,
ICtsPlugInRenderer *pNestedPlugIn
);
Remarks
現在は使用していない。

HRESULT ICtsPlugInNotifier::EnumPlugIns(
ICtsPlugInInfo *pTHIS,
CtsPlugInType type,
int nIndex,
ICtsPlugInInfo **ppPlugInInfo)
Paramter:
[in] ICtsPlugInInfo *pTHIS
Plug-In の Info インターフェースポインタ。
[in] CtsPlugInType type
Plug-In カテゴリを指定する。
CtsPlugInTypeUnknown を指定した場合，カテゴリに関係なく全ての Plug-In を enum することを意味する。
[in] int nIndex
type で指定されたカテゴリの何番目の Plug-In を取得するのかを指定。
[out] ICtsPlugInInfo **ppPlugInInfo
Enum で見つかった Plug-In Info へのポインタが格納される。NULL が格納されていた場合、type で指定されたカテゴリのエフェクトのうち nIndex 番目のエフェクトは存在しないことを意味する。

Return Value:
成功したら S_OK を返す。
失敗したらエラー値を返す。
Remarks:
この関数を使用して取得した ICtsPlugInInfo インターフェースポインタは必要が無くなった時点で、参照カウンタを decrement する為に解放（Release）しなければならない。.

実際に使用する時は以下の例を参考に記述すること。

```
ICtsPlugInInfo * pPlugInInfo;
int nIndex = 0;
do{
    HRESULT hr = pPlugInFactory->EnumPlugIns(this, CtsPlugInTypeVideoFilter, nIndex++,
&pPlugInInfo);
    .... // pPlugInInfo に対する何かの処理
}while(pPlugInInfo!=NULL);
```

```
HRESULT ICtsPlugInNotifier::GetPlugIn(
        ICtsPlugInRenderer* pTHIS,
        LPCTSTR                                 matchName,
        ICtsPlugInInfo          **ppPlugInInfo)
```

[parameter]

ICtsPlugInRenderer* pTHIS

この関数を呼び出す plug-in の plug-in renderer インターフェースポインタを指定する。

LPCTSTR matchName

取得したいプラグインの match name を指定する。

ICtsPlugInInfo **ppPlugInInfo

matchName で指定された match name と同一の plug-in のインターフェースポインタが格納される出力用ポインタ。*ppPlugInInfo は NULL で初期化されていなければならない。

[return value]

取得に成功した場合は S_OK．失敗した場合はエラー値を返す。

[note]

match name を指定してプラグインを取得する。
プラグイン内で更にプラグインを指定したり参照したりする必要がある場合にのみ使用する。それ以外の用途では使えない。

取得したインターフェースは正しく Release() しておくこと．解放し忘れた場合メモリリークが発生する。

```
HRESULT ICtsPlugInNotifier::RegistToNotifyCallback(
        CtsFlags                callbackTypes,
        ICtsPlugInRenderer      *pTHIS,
        DWORD                   dwObjectID,
        FnCtsPlugInCallback     fnCallback,
        LPVOID                  pUserParam
        );
```

Parameter:

[in] CtsFlags callbackTypes

コールバックのタイプを CtsPiCallbackType の組み合わせで指定する。
タイプは以下の通り

- CtsPiCallbackTypeTLCursor : Time line Cursor が動いた時。
- CtsPiCallbackTypeStatus : Playback の状態が変わった時。

[in] ICtsPlugInRenderer *pTHIS
plug-in の インターフェースポインタ。
[in] DWORD dwObjectID,
plug-in の CreateInstance で渡される objectID。
[in] FnCtsPlugInCallback fnCallback
コールバック関数のポインタ。
コールバック関数の定義は Remarks を参照。
[in] LPVOID pUserParam
コールバック関数が呼び出される時に渡される引数。

Return Value:
成功したら S_OK。
失敗したらエラー値を返す。

Remarks:
コールバック関数を登録する．登録されたコールバック関数は
指定されたタイプに従って呼び出される。
コールバック関数の定義は以下の通り。
コールバック定義の詳細は以下の "Plug-in Callback Specification" の項目を参照のこと。

```
HRESULT ICtsPlugInNotifier::UnRegistToNotifyCallback(
        CtsFlags                callbackTypes,
        ICtsPlugInRenderer      *pTHIS,
        DWORD                   dwObjectID,
        FnCtsPlugInCallback     fnCallback,
        LPVOID                  pUserParam
        );
```

Parameter:
[in] CtsFlags callbackTypes
削除したいコールバック関数のタイプを CtsPiCallbackType の組み合わ
せで指定する。コールバック関数を登録する時に指定していないタイプ
は指定しても意味は無い。
[in] ICtsPlugInRenderer *pTHIS
plug-in の インターフェースポインタ。
[in] DWORD dwObjectID,
plug-in の CreateInstance で渡される objectID。
[in] FnCtsPlugInCallback fnCallback
コールバック関数のポインタ。
関数の定義は以下の通り。
typedef HRESULT (CALLBACK *FnCtsPlugInCallback)(CtsPiCallbackType, WPARAM, LPARAM, LPVOID);

Return Value:
成功したら S_OK。
失敗したらエラー値を返す。

Remarks:
コールバック関数を削除する。
コールバック定義の詳細は以下の "Plug-in Callback Specification" の項目を参照のこと。

```
HRESULT ICtsPlugInNotifier::OpenSetup(ICtsPlugInRenderer *pPlugIn, HWND hParentWnd)
```

Parameter:

ICtsPlugInRenderer *pPlugIn

plgu-in renderer interface 自体のポインタ

HWND hParentWnd

parent window をハンドルする。

Return Value:

成功したら S_OK。

失敗したらエラー値を返す。

Remarks:

==========

● Plug-in Callback specification ●

==========

Plug-in Callback の種類は以下の通り。

- o CtsPiCallbackTypeMarker
  -> マーカーが編集された時。
- o CtsPiCallbackTypeProjectSetting
  -> プロジェクト設定が変更された時。

- o CtsPiCallbackTypePrvMouse
  -> Preview 上で MouseEvent が発生した時(Move, Leave, Down)。
- o CtsPiCallbackTypeTLStop
  -> Time Line が STOP された時。
- o CtsPiCallbackTypeTLPlay
  -> Time Line が PLAY された時。
- o CtsPiCallbackTypeTLScrub
  -> Time Line Cursor が Scrub された時。

- o CtsPiCallbackTypeRegisteredPi
  -> プラグインが新たに登録された時。
- o CtsPiCallbackTypeLoadedPiDLL
  -> plug-in Manager によりプラグインＤＬＬが新たにロードされた時。
- o CtsPiCallbackTypeUpdatedPi
  -> プラグインのパラメタが更新された時。
- o CtsPiCallbackTypeClosedPi
  -> プラグインの Setup Dialog が閉じられた時。

Plug-in Callback 関数の定義は次頁の通り。

HRESULT CALLBACK (*func)(CtsPiCallbackType, WPARAM, LPARAM, LPVOID);
[parameter]
CtsPiCallbackType : Plug-in Callback のタイプ
WPARAM, LPARAM : 以下の表を参照

Callback Type	WPARAM	LPARAM
Marker	NOT IMPLEMENTED YET	-
ProjectSetting	NOT IMPLEMENTED YET	-
PrvMouse	CCtsPiCbInfoPrevMouse : Mouse 情報	NULL
TLStop	CtsFrame : 現在の絶対フレーム番号	NULL
TLPlay	CtsFrame : 現在の絶対フレーム番号	NULL
TLScrub	CtsFrame : 現在の絶対フレーム番号	NULL
RegisteredPi	ICtsPlugInRenderer* : 登録された plug-in	NULL
LoadedPiDLL	LPTSTR : loaded plug-in dll's name	int* : index of the dll count
UpdatedPi	ICtsPlugInRenderer* : 更新された plug-in	NULL
ClosedPi	ICtsPlugInRenderer* : Setup が閉じた plug-in	BOOL* : whether Ok or Cancel

LPVOID
Receives the parameter data passed to the function using the pUserParam parameter of the RegistToNotifyCallback() function.
//-------------------------

[notice]
Plug-in コールバック関数は、あらゆる plug-in からの Regist, Loaded, Update, Closed かに関わらず全て呼び出される。従って、自分が登録した plug-in に関するコールバックであるかどうかは WPARAM に格納されている ICtsPlugInRenderer* のポインタアドレスを，自分が持つメンバと比較することで判別する必要がある。例は以下の通り。

(example)
HRESULT CALLBACK ClosedPiCallback(CtsPiCallbackType type, WPARAM wparam, LPARAM lparam, LPVOID pUserParam)
{
  if(type != CtsPiCallbackTypeClosedPi){
    return E_FAIL;
  }

  // Is this plug-in mine ?
  ICtsPlugInRenderer *pRenderer = reinterpret_cast<ICtsPlugInRenderer*>(wparam);
  if(pRenderer == g_pMyPlugInRenderer){
    ::OutputDebugString(_T("コールバックを登録した自分のプラグインだ！¥n"));
  }
  else{
    ::OutputDebugString(_T("コールバックを登録した自分のプラグインではない！¥n"));
  }
}

==
● Plug-in Callback information class specification ●
==

一部のコールバック関数で得られる WPARAM/LPARAM は callback 用に複数の
上が格納されたクラスの型の時がある．ここでは，そのような callback 用
information クラスの説明をする

[PrvMouse] callback の information class

=-=
<ICtsPiCbInfoPrevMouse>
=-=
[Required]
  CtsPiCbInfoPrevMouse.h
[Interface ID]
  IID_ICtsPiCbInfoPrevMouse

[Derived]
  IUnknown

[methods overview]
  - preview 上でマウスカーソルイベントが起こった際に呼ばれる callback 関数
    でマウス情報やマウスカーソルが指す位置の色情報等の各種情報を提供する
    ための interface クラス。

[Remarks]
  - CtsPiCallbackTypePrvMouse を指定して登録したコールバックの引数 WPARM
    をこの interface 型にキャストして使用すること。

COLORREF ICtsPiCbInfoPrevMouse::GetRGB(void)
Paramter:
  none
Return Value:
  マウスカーソルイベントが起こった時のカーソル位置にあるピクセルの色
  を RGB で返す。
Remarks:
  カーソル位置の色情報を RGB で取得する。

COLORREF ICtsPiCbInfoPrevMouse::GetYUV(void)
Paramter:
  none
Return Value:
  マウスカーソルイベントが起こった時のカーソル位置にあるピクセルの色
  を YUV で返る。COLORREF 構造体で返ってくるが，以下のように値が対応
  している。
    R -> Y
    G -> U
    B -> V
Remarks:

カーソル位置の色情報を YUV で取得する。

POINT　　ICtsPiCbInfoPrevMouse::GetPoint(void)
Paramter:
　none
Return Value:
　マウスカーソルイベントが起こった時のカーソル位置座標を返す。
Remarks:
　カーソル位置座標を取得する。

HRESULT ICtsPiCbInfoPrevMouse::GetPix(ICtsPix** ppPix)
Parameter:
　　ICtsPix** ppPix
　　　　stroed pix data of preview overlay
Return Value:
　　if this function succeeds, returns S_OK,
　　if not, returns an error value.

CtsPiCbInfoPrvMouse ICtsPiCbInfoPrevMouse::GetMouseStatus(void)
Paramter:
　none
Return Value:
　マウスカーソルイベントのタイプを取得する。
　現在取得できるタイプは以下の通り。
　　- CtsPiCbInfoPrvMouseMove ： マウスが Preview 上で動いた時。
　　- CtsPiCbInfoPrvMouseLeave　： マウスが Preview 上を離れた時。
　　- CtsPiCbInfoPrvMouseLDown　： マウスが Preview 上で押下された時。
Remarks:

=-=
<ICtsPiCbMarker>
=-=

[Required]
CtsPiCbMarker.h

[Interface ID]
IID_ICtsPiCbMarker

[Derived]
IUnknown

[method overview]
- this interface provides the marker callback informations

[Remarks]
- please cast the WPARAM passed by the callback function to this
  interface pointer.

CtsFrame ICtsPiCbMarker::GetEventTC()

Parameter:
none

Return value:
マーカーイベントが発生したフレーム番号を絶対フレーム番号で取得する。

Note:

CtsPiCbMarkerEventType ICtsPiCbMarker::GetType()

Parameter:
none

Return value:
マーカーイベントのタイプを取得する．イベントのタイプについては
CtsPiCbMarker.h ファイルを参照のこと。

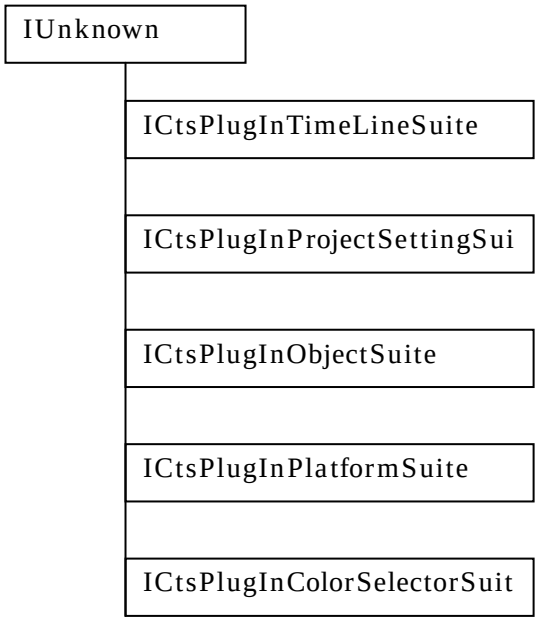
Remarks:
this function provide you the marker event type.
types is defined CtsPiCbMarkerEventType written above.

==========

● Suite Interface for Plug-In ●

==========

□ Suite for Plug-In Interface のクラス階層図



=-=
<ICtsPlugInTimeLineSuite>
=-=
[Required]
CtsPlugInTimeLineSuite.h

[Interface ID]
IID_ICtsPlugInTimeLineSuite

[Derived]
IUnknown

[methods overview]
- Time Line の情報を取得/設定する。

[Remarks]
- ICtsPlugInServiceFactory::CreateSuite 関数を IID_ICtsPlugInTimeLineSuite を引数にして呼び出すことにより作成出来る。

```
HRESULT ICtsPlugInTimeLineSuite::GetTimeLineInOut(
    CtsFrame *in,
    CtsFrame *out)
```

Paramter:
[out] CtsFrame *in
タイムライン上 In 点フレーム番号のポインタ。
[out] CtsFrame *out
タイムライン上 Out 点フレーム番号のポインタ。

Return Value:
成功したら S_OK を返す。
失敗したらエラー値を返す。

Remarks:
Plug-In に関連づけられている TimeLine 上 Object の In 点/Out 点 をフレーム番号で取得する。
*in か *out かのいずれかに CtsFrameConstInvalid が指定されている場合はこの関数は TimeLine 全体の In/Out を返す。ただし、TimeLine 全体の In/Out の取得に失敗した場合(TimeLine 上にクリップが一つもない場合など)は In/Out 共に 0 が代入されて返ってくる。

```
CtsFrame ICtsPlugInTimeLineSuite::GetCurrentFrame()
```

Paramter:
none

Return Value:
Time Line Cursor 現在位置の相対フレーム番号が返る。

Remarks:
Time Line Cursor 現在位置の相対フレーム番号を取得する。
Time Line Cursor の現在位置が plug-in の適用されている clip object よりも前にある場合、CtsFrameConstMinusInfinite が返る。
Time Line Cursor の現在位置が plug-in の適用されている clip object よりも後にある場合、CtsFrameConstInfinite が返る。

HRESULT ICtsPlugInTimeLineSuite::SetCurrentFrame(CtsFrame frmFrame)
Paramter:
  CtsFrame frmFrame
    設定したい TimeLine Cursor の位置を frame 番号で指定。
Return Value:
  成功したら S_OK を返す。
  失敗したらエラー値を返す。
Remarks:
  TimeLine Cursor の現在位置を変更する。
  ただし、この suite を作成する際に指定された object ID が不正な値で
  であった場合は何も起こらない。

---
int ICtsPlugInTimeLineSuite::GetNumberOfMarkers()
Paramter:
  none
Return Value:
  TimeLine の現在のマーカーの数を返す。
Remarks:
  TimeLine の現在のマーカーの数を取得する。

CtsFrame ICtsPlugInTimeLineSuite::GetMarkerPosition(int nMarkerIndex)
Paramter:
  [in] int nMarkerIndex
    マーカーの index 番号。
Return Value:
  TimeLine に設定されているマーカーのうち、nMarkerIndex 番目のマーカー
  が指している位置を frame Number で返す。
Remarks:
  TimeLine に設定されているマーカーのうち、nMarkerIndex 番目のマーカー
  が指している位置を frame Number で取得する。

CtsTimeCode ICtsPlugInTimeLineSuite::ConvertToTimeCode(CtsFrame nFrameNumber)
Paramter:
  [in] CtsFrame nFrameNumber
     IN 点からの相対フレーム番号。
Return Value:
  フレーム番号を Time code に変換する。
Remarks:
  フレーム番号を Time Code に変換する。

CtsSample ICtsPlugInTimeLineSuite::ConvertToNumberOfSamples(CtsFrame nFrameNumber)
Paramter:
  [in] CtsFrame nFrameNumber
    IN 点からの相対フレーム番号。
Return Value:
  フレーム番号の位置での sampling 数を返す。

Remarks:
　フレーム番号に該当する位置の sampling 数を取得する。

HRESULT ICtsPlugInTimeLineSuite::Play(CtsTLPlayOption mode, CtsFraction rate)
- -
Parameters:
 CtsTLPlayOption mode
　再生モードを指定する．指定できるモードは
　　CtsTLPlayOptionNormal　　：標準再生。
　　CtsTLPlayOptionPreRoll　　: PreRoll を前後に追加して再生。
　　CtsTLPlayOptionLoop　　：ループ再生。
　　CtsTLPlayOptionPreRollLoop　: PreRoll を前後に追加してループ再生
　の４種類。

 CtsFraction rate
　再生速度を分数で表した値。

Return Value:
　成功したら S_OK を返す。
　失敗したらエラー値を返す。

Remarks:
 この関数を使うことで plug-in が適用されている TimeLine Object の IN/OUT 点間
 (or PreRoll を加えた区間)の再生が可能。
 再生中にこの関数を呼び出した場合、一時停止となる。
　ただし、この suite を作成する際に指定された object ID が不正な値で
　であった場合は何も起こらない。

HRESULT ICtsPlugInTimeLineSUite::Stop(void)
- -
Parameters:
 なし

Return Value:
　成功したら S_OK を返す。
　失敗したらエラー値を返す。

Remarks:
 再生中の Timline を停止させる。
 Timeline が停止中の時に呼び出しても何も起こらない。(S_OK が返る)
　ただし、この suite を作成する際に指定された object ID が不正な値で
　であった場合は何も起こらない。

CtsFlags ICtsPlugInTimeLineSuite::GetRenderingTargetType(void)
- -
Parameter:
　なし
Return Value:

Core Renderer が何を対象に Rendering しようとしているのかを表す
値が返る。現在、以下の２通りの値が返る。
- CtsTSRTargetTypeOutputFile ：ファイル出力。
- CtsTSRTargetTypeUnknown ：それ以外(Play, Scrub 中など)。

Remarks:
Core Renderer が何を対象に Rendering しようとしているかを調べる。

=-=
<ICtsPlugInProjectSettingSuite>
=-=

[Required]
  CtsPlugInProjectSettingSuite.h

[Interface ID]
  IID_ICtsPlugInProjectSettingSuite

[Derived]
  IUnknown

[methods overview]
  - Project 設定の情報を取得する。

[Remarks]
  - ICtsPlugInServiceFactory::CreateSuite 関数を IID_ICtsPlugInProjectSettingSuite
    を引数にして呼び出すことにより作成できる。

HRESULT ICtsPlugInProjectSettingSuite::GetVideoInfo(ICtsVideoInfo **ppVideoInfo)
- -

Parameter:
  [out] ICtsVideoInfo **ppVideoInfo
    Video Information の interface ポインタが格納される。

Return Value:
  成功したら S_OK を返す。
  失敗したらエラー値を返す。

Remarks:
  Video に関する情報がおさまった Video Information class への interface を
  取得する。Video Infrmation からは FrameSize や Aspect Ratio, OverScan Size
  等を取得することができる。詳細については ICtsPix.h ファイルを参照のこと。

HRESULT ICtsPlugInProjectSettingSuite::GetAudioInfo(ICtsAudioInfo **ppAudioInfo)
- -

Parameter:
  [out] ICtsAudioInfo **ppAudioInfo
    Audio Information の interface ポインタが格納される。

Return value:
  成功したら S_OK を返す。
  失敗したらエラー値を返す。

Remarks:
  Audio に関する情報がおさまった Audio Information class への interface を
  取得する。Audio Infrmation からは Bit rate や Channel 数等を取得することが
  できる。詳細については ICtsSnd.h ファイルを参照のこと。

LPCTSTR ICtsPlugInProjectSettingSuite::GetProjectFileName()
- -

Parameter:
  なし

Return Value:

成功したら S_OK を返す。

失敗したらエラー値を返す。

Remarks:

project filename を取得できる。

この関数の取得によりポインタを削除する必要はない。

.

=-=

<ICtsPlugInObjectSuite>

=-=

[Required]
  CtsPlugInObjectSuite.h

[Interface ID]
  IID_ICtsPlugInObjectSuite

[Derived]
  IUnknown

[methods overview]
  - Clip の情報を取得する。

[Remarks]
  - ICtsPlugInServiceFactory::CreateSuite 関数を IID_ICtsPlugInObjectSuite
    を引数にして呼び出すことにより作成出来る。

```
HRESULT ICtsPlugInObjectSuite::GetVideoSource(
        CtsFrame    offsetFrame,
        int         nSubTrack,
        ICtsPix**   ppPix)
```

- -

Parameter:
  [in] CtsFrame frmFrame
    取得したい画像のフレームの番号を Time Line Object の offset 値で指定する。
  [in] int nSubTrack
    取得する画像の種類を指定する。
    VideoSuite を作成した Plug-In のカテゴリにより意味が異なる。
     - Video Filter, Title Effect
        無視出来る。
     - Transition
        0 : From 側の画像を取得する。
        1 : To 側の画像を取得する。
     - Keyer
        0 : Below 側の画像を取得する。
        1 : Above 側の画像を取得する。
  [out] ICtsPix** ppPix
    取得した画像の ICtsPix が格納される。

Return Value:
  成功したら S_OK を返す。
  失敗したらエラー値を返す。

Remarks:
  Plug-In が Bind されている TimeLine 上 Object の Plug-In 自身が効果を
  与える直前までのレンダリング結果の画像データを取得する。
  nSubTrack に指定された種類の画像データを取得する。
  画像データが取得できなかった場合はエラー値が返り、なおかつ *ppPix に
  NULL が設定される。

HRESULT ICtsPlugInObjectSuite::GetVideoSourceRGB(
    CtsFrame   offsetFrame,
    int   nSubTrack,
    ICtsPix**   ppPix)
- -
Remarks:
  基本的に GetVideoSource() 関数と機能は同じ．RGB 列を返す。

HRESULT ICtsPlugInObjectSuite::GetVideoSourceBySearch(
    CtsFrame   offsetFrame,
    int   nSubTrack,
    ICtsPix**   ppPix
    );
- -
Remarks:
  基本的に GetVideoSource() と機能は同じ。
  この関数は plug-in が CreateInstance される時に指定される Object ID が
  不正な値、あるいは分からない場合に利用すると良い。plug-in manager を
  介して Middle Layer に plug-in instance の情報を検索してくれる。
  (但し，setup dialog が表示されている時のみ有効)

HRESULT ICtsPlugInObjectSuite::GetAudioSource(
    CtsSample   offsetSample,
    int   nSubTrack,
    ICtsSnd**   ppSnd)
- -
Parameter:
  [in] CtsSample offsetSample
    Time Line Object の offset 値を指定する。
  [in] int nSubTrack
    取得する音声の種類を指定する。
    AudioSuite を作成した Plug-In のカテゴリにより意味が異なる。
     - Audio Filter
        無視出来る。
     - Cross Fade
        0 : 'From' 側の音声を取得する。
        1 : 'To' 側の音声を取得する。
  [out] ICtsPix** ppPix
Return Value:
  成功したら S_OK を返す。
  失敗したらエラー値を返す。
Remarks:
  Plug-In が Bind されている TimeLine 上 Object の Plug-In 自身が効果を
  与える直前までのレンダリング結果の音声データを取得する。
  nSubTrack に指定された種類の音声データを取得する。
  音声データが取得できなかった場合はエラー値が返り、なおかつ *ppSnd に
  NULL が設定される。

POINT ICtsPlugInObjectSuite::GetOriginPoint(void);
- -
Parameter:
　なし
Return Value:
　Object の表示位置を取得する。
Remarks:
　Object の表示位置を取得する。
　この関数はレイアウター plug-in で使用する。

HRESULT ICtsPlugInObjectSuite::SetOriginPoint(POINT pnt);
- -
Parameter:
　POINT pnt
　　Object の表示位置の座標を指定する。
Return Value:
　成功したら S_OK を返す。
　失敗したらエラー値を返す。
Remarks:
　Object の表示位置を設定する。
　この関数はレイアウター plug-in で使用する。

=-=
<ICtsPlugInPlatformSuite>
=-=

[Required]
  CtsPlugInPlatformSuite.h

[Interface ID]
  IID_ICtsPlugInPlatformSuite

[Derived]
  IUnknown

[methods overview]
  - plug-in が動作している PC や Application についての情報を取得する

[Remarks]

HRESULT ICtsPlugInPlatformSuite::CPUHasMMX()
- -

Parameter:
  なし
Return Value:
  CPU が MMX をサポートしている場合、TRUE を返す。
  サポートしていない場合、FALSE を返す。
Remarks:
  CPU が MMX をサポートしているかどうか調べる。

HRESULT ICtsPlugInPlatformSuite::CPUHasMMX2()
- -

Parameter:
  なし
Return Value:
  CPU が MMX2 をサポートしている場合、TRUE を返す。
  サポートしていない場合、FALSE を返す。
Remarks:
  CPU が MMX2 をサポートしているかどうか調べる。

HRESULT ICtsPlugInPlatformSuite::CPUHasSSE()
- -

Parameter:
  なし
Return Value:
  CPU が SSE をサポートしている場合、TRUE を返す。
  サポートしていない場合、FALSE を返す。
Remarks:
  CPU が SSE をサポートしているかどうか調べる。

HRESULT ICtsPlugInPlatformSuite::CPUHasSSE2()

Parameter:
  なし
Return Value:
  CPU が SSE2 をサポートしている場合、TRUE を返す。
  サポートしていない場合。FALSE を返す。
Remarks:
  CPU が SSE2 をサポートしているかどうか調べる。

HRESULT ICtsPlugInPlatformSuite::CPUHas3DNow()
- -
Parameter:
  なし
Return Value:
  CPU が 3DNow をサポートしている場合、TRUE を返す。
  サポートしていない場合、FALSE を返す。
Remarks:
  CPU が 3DNow をサポートしているかどうか調べる。

HRESULT ICtsPlugInPlatformSuite::CPUHasE3DNow()
- -
Parameter:
  なし
Return Value:
  CPU が 拡張 3DNow をサポートしている場合、TRUE を返す。
  サポートしていない場合、FALSE を返す。
Remarks:
  CPU が 3DNow をサポートしているかどうか調べる。

HRESULT BOOL ICtsPlugInPlatformSuite::CPUHasHT()
- -
Paramter:
  なし
Return Value:
  CPU が HyperThreading をサポートしている場合、TRUE を返す。
  サポートしていない場合、FALSE を返す。
Remarks:
  CPU が HyperThreading をサポートしているかどうかを調べる。

CtsFlags ICtsPlugInPlatformSuite::GetEdititon(void)
- -
Parameter:
  なし
Return Value:
  アプリケーションの Edition を表す値を返す。
  現在，返される可能性のある値は以下の２種類。
      CtsPFEditionReg ：正規版アプリケーション
      CtsPFEditionDemo：デモ版アプリケーション

HRESULT ICtsPlugInPlatformSuite::SetStatusBarText(LPCTSTR text)

- -

Parameter:

　なし

Return Value:

　常に S_OK を返す。

Remarks

　タイムラインウィンドウの下にあるステータスバーに任意の文字列を表示する。

=-=
<ICtsPlugInColorSelectorSuite>
=-=

[Required]
CtsPlugInColorSelectorSuite.h

[Interface ID]
IID_ICtsPlugInColorSelectorSuite

[Derived]
IUnknown

[methods overview]
- 色選択ダイアログを表示して選択された色情報を取得する。

[Remarks]

HRESULT ICtsPlugInPlatformSuite::Show(HWND hParent)
- -

Parameter:
HWND hParent
親となる window の window handle を指定する。

Return Value:
成功したら S_OK を返す。
失敗した場合はエラー値を返す。

Remarks:
EDIUS 風色選択ダイアログを表示する。

COLORREF ICtsPlugInPlatformSuite::GetRGB(void)
- -

Parameter:
なし

Return Value:
色選択ダイアログで選択された色を COLORREF 構造体で返す。

Remarks:
色選択ダイアログで選択された色を RGB 値で取得する。

COLORREF ICtsPlugInPlatformSuite::GetYUV(void)
- -

Parameter:
なし

Return Value:
色選択ダイアログで選択された色を COLORREF 構造体で返す。
COLORREF 構造体の RGB はそれぞれ以下のように対応している。
R -> Y
G -> U
B -> V

Remarks:
色選択ダイアログで選択された色を YUV 値で取得する。

---
void ICtsPlugInPlatformSuite::SetYUV(COLORREF yuv)
- -
Parameter:
 COLORREF yuv
  YUV 値を COLORREF 構造体で指定する。
   R ： Y
   G ： U
   B ： V
Return Value:
  なし
Remarks:
 色選択ダイアログが表示された時の色を YUV 値で設定する。

---
void ICtsPlugInPlatformSuite::SetRGB(COLORREF rgb)
- -
Parameter:
 COLORREF rgb
  RGB 値を COLORREF 構造体で指定する。
Return Value:
  なし
Remarks:
 色選択ダイアログが表示された時の色を RGB 値で設定する。
