CAMThread::ThreadProc

CAMThread Class

Overridden member function in which to implement a thread.

virtual DWORD ThreadProc( );

Return Values

The meaning of this return value is not defined by the CAMThread class.

Remarks

The thread calls this member function upon startup. Derived classes must override this member function. When this member function returns, the thread terminates. This member function is protected.

CAutoLock Class

CAutoLock

The **CAutoLock** class holds a critical section for the scope of a block or function. The constructor locks the critical section and the destructor unlocks it. The object passed to the **CAutoLock** constructor must be derived from the CCritSec class. Thus, by declaring a **CAutoLock** object as a local variable in a function, a critical section can be locked without the danger of forgetting to unlock it in some of the code paths: the destructor ensures that upon exit from the function (or the scope of the declaration), the critical section will be unlocked. Member functions in this class are not designed for overriding.

```
/* Typical usage ensuring object is always unlocked correctly */

HRESULT MyFunc(IMediaSample *pSample)
{
    CAutoLock cObjectLock(m_pMyLock);

    /* Ignore samples passed when inactive */

    if (!m_bActive) {
        return NOERROR;
    }

    /* Add the sample to the pending queue */

    HRESULT hr = m_PendingList.AddTail(pSample);
    if (FAILED(hr)) {
        pSample->Release();
        return hr;
    }
    return NOERROR;
}
```

Protected Data Members

Name	Description
m_pLock	Critical section for this lock.

Member Functions

Name	Description
CAutoLock	Takes a pointer to a critical section object and locks it.

CAutoLock::CAutoLock

CAutoLock Class

Takes a pointer to a critical section object and locks it.

```
CAutoLock(
  CCritSec * plock
  );
```

Parameters

plock
Pointer to a critical section object.

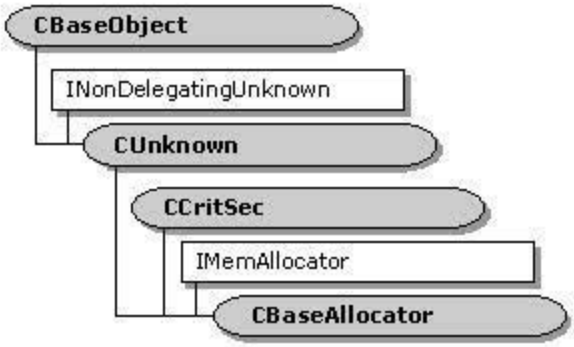
Return Values

No return value.

Remarks

The critical section is unlocked when the CAutoLock object is destroyed.

CBaseAllocator Class



CBaseAllocator is an abstract base class that implements the basic mechanisms for an allocator with a fixed number of fixed-size buffers. The number of buffers and their size can be changed using the CBaseAllocator::SetProperties member function when an input pin and an output pin negotiate the allocator between them.

The class provides the basic functionality for a memory allocator by implementing the IMemAllocator interface. It provides support for managing a list of CMediaSample objects (or objects derived from this class), including support for the IMemAllocator::Commit and IMemAllocator::Decommit methods, and blocking the IMemAllocator::GetBuffer method.

Any class derived from this class (such as CMemAllocator) must create CMediaSample objects, which this base class does not.

A signaling mechanism employing a semaphore is used so that if there are no samples, a thread can wait until there are samples or until the allocator is decommitted. The m_lFree and m_hSem member variables are used to implement this simple signaling mechanism as follows.

When a thread calls CBaseAllocator::GetBuffer and there are no samples available, m_lWaiting is incremented and the thread calls the Microsoft® Win32® WaitForSingleObject function on the semaphore indicated by m_hSem.

When a sample is freed (by the IUnknown::Release method returning the reference count to zero) or CBaseAllocator::Decommit is called and m_lWaiting is nonzero, the Win32 ReleaseSemaphore function is called on m_hSem with a release count of **m_lWaiting**, and **m_lWaiting** is reset to zero.

All member functions in this class that return HRESULT and accept a pointer as a parameter return E_POINTER when passed a null pointer.

Protected Data Members

Name	Description
m_bChanged	TRUE if the buffers have changed; otherwise, FALSE.
m_bCommitted	If TRUE, the allocator is in a state in which all IMemAllocator::GetBuffer methods fail. The IMemAllocator::SetProperties method is the only member function permitted to operate in this state.
m_bDecommitInProgress	If TRUE, the decommit process completes upon the return of all media samples. Until the decommit process has completed, any calls to IMemAllocator::GetBuffer return E_OUTOFMEMORY.
m_hSem	Semaphore for signaling.
m_lAlignment	Agreed alignment of the buffer.
m_lAllocated	Number of buffers actually allocated.
m_lCount	Established number of buffers to provide.
m_lFree	List of CMediaSample objects that are currently free (free list).
m_lPrefix	Agreed prefix of the buffer (precedes value returned by IMediaSample::GetPointer).
m_lSize	Size of each buffer.
m_lWaiting	Count of threads waiting for samples.

Member Functions

Name	Description
CBaseAllocator	Constructs a CBaseAllocator object.
NotifySample	Notifies a waiting thread that a sample is available on the free list.
SetWaiting	Increments the m_lWaiting data member to indicate that a thread is waiting for a sample.

Overridable Member Functions

Name	Description
Alloc	Allocates memory, instantiates CMediaSample objects, and adds them to the m_lAllocated and m_lFree data members.
Free	Decommits memory when the last buffer is freed.

Implemented IMemAllocator Methods

Name	Description
Commit	Allocates memory by calling the CBaseAllocator::Alloc member function, which you must override.
Decommit	Releases any resources and enters the inactive state. Any blocking calls to IMemAllocator::GetBuffer should return with an error value, and all further calls to GetBuffer fail when in the inactive state.
GetBuffer	Retrieves a container for a sample.
GetProperties	Determines the size, number, and alignment of blocks.
ReleaseBuffer	Releases the CMediaSample object.
SetProperties	Specifies a desired number of blocks, size of the blocks, and block alignment figure. This method returns the actual values for the same.

Implemented INonDelegatingUnknown Methods

Name	Description
NonDelegatingQueryInterface	Passes out pointers to any interfaces added to the derived filter class.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseAllocator::Alloc

CBaseAllocator Class

Allocates a media sample object.

HRESULT Alloc(void);

Return Values

Returns an HRESULT value.

Remarks

Override this member function to allocate memory, instantiate CMediaSample objects, and add them to the free list represented by the m_lFree data member. The CMemAllocator::Alloc member function is an example of an override of this member function. It calls this member function first to ensure that allocator properties have been set.

This member function is called from the CBaseAllocator::Commit member function when entering the active state. The default implementation returns an error value if the IMemAllocator::SetProperties method has not been called yet, and checks that there are no outstanding buffers.

Previous Home Topic Contents Index Next

CBaseAllocator::CBaseAllocator

CBaseAllocator Class

Constructs a CBaseAllocator object.

CBaseAllocator(
TCHAR * *pName*,
LPUNKNOWN *lpUnk*,
HRESULT * *phr*,
BOOL *bListSemaphore* **= TRUE**
);

Parameters

pName
Name of the allocator object.
lpUnk
Pointer to LPUNKNOWN.
phr
Pointer to an HRESULT for return values. This is not modified unless this member function fails.
bListSemaphore
If TRUE, the free list in the allocator has a semaphore associated with it. If FALSE, no semaphore is created for the list. Setting this to FALSE can be useful for subclassing CBaseAllocator when the semaphore is not required for blocking. The semaphore is used for the waiting and signaling mechanism.

Return Values

No return value.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseAllocator::Commit

CBaseAllocator Class

Commits the memory for the specified buffers.

HRESULT Commit(void);

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IMemAllocator::Commit method. The IMemAllocator::SetProperties method must be called before calling this member function. This member function sets m_bCommitted to TRUE and overrides any pending decommit operation. It then calls the CBaseAllocator::Alloc member function to allocate memory (which should be overridden in the derived class to call the base class member function and then allocate the memory). The IMemAllocator::GetBuffer method fails if it is called before calling this member function.

Call CBaseAllocator::Decommit to release memory when done with the buffers.

Previous Home Topic Contents Index Next

CBaseAllocator::Decommit

CBaseAllocator Class

Releases the memory for the specified buffers.

HRESULT Decommit(void);

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IMemAllocator::Decommit method. Any threads waiting in the IMemAllocator::GetBuffer method return with an error after this method is called. The **IMemAllocator::GetBuffer** method fails if it is called before the IMemAllocator::Commit method or after this method.

See Also

CBaseAllocator::Commit

Previous Home Topic Contents Index Next

CBaseAllocator::Free

CBaseAllocator Class

Called to decommit the memory when the last buffer is freed.

virtual void Free(void) PURE;

Return Values

No return value.

Remarks

This member function must be implemented in the derived class. It is called from CBaseAllocator::ReleaseBuffer when a decommit is pending and the allocator has put its last buffer on the free list. It is also called from CBaseAllocator::Decommit.

The CMemAllocator::Free member function is an example of how this can be implemented in the derived class. In this case, it simply returns, because the CMemAllocator class releases memory from its destructor.

This member function is protected.

Previous Home Topic Contents Index Next

CBaseAllocator::GetBuffer

CBaseAllocator Class

Retrieves a container for a sample.

HRESULT GetBuffer(
IMediaSample ** *ppBuffer*,
REFERENCE_TIME * *pStartTime*,
REFERENCE_TIME * *pEndTime*,
DWORD *dwFlags*
);

Parameters

ppBuffer
Pointer to a retrieved media sample buffer.
pStartTime
Either NULL or set to the beginning time of the sample to retrieve.
pEndTime

Either NULL or set to the ending time of the sample to retrieve.

dwFlags

The following flags are supported.

Value	Meaning
AM_GBF_PREVFRAMESKIPPED	The buffer returned will not be filled with data contiguous to any previous data sent.
AM_GBF_NOTASYNCPOINT	Dynamic format changes are not allowed on this buffer because it is not a key frame.

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IMemAllocator::GetBuffer method.

This is a blocking, synchronous call to access the next free buffer (as represented by an IMediaSample interface). Upon return, properties (such as the time and so on) are invalid, but the buffer pointer and size are correct.

If no buffers are available, **CBaseAllocator::GetBuffer** calls CBaseAllocator::SetWaiting and then calls the Microsoft® Win32® WaitForSingleObject function to wait for the list to signal that a sample is available. The list signals by calling CBaseAllocator::ReleaseBuffer, which in turn calls CBaseAllocator::NotifySample, which sets m_lWaiting to zero and calls the Win32 ReleaseSemaphore function.

This member function also takes two time parameters. These parameters are used in certain advanced buffering scenarios, when it is necessary to have an idea of the amount of time a buffer is required. The only place this is currently used is in the video renderer, when the time stamps are used as a guide to when the primary surfaces of Display Control Interface (DCI) and Microsoft® DirectDraw® should be returned (this is because filling a primary surface buffer corresponds directly to the actual rendering of the data). In all other cases, these parameters can be safely set to NULL. If one is non-NULL, both should be non-NULL; these times will not be set on the sample when it is subsequently returned.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseAllocator::GetProperties

CBaseAllocator Class

Retrieves the size, number, and alignment of blocks.

HRESULT GetProperties(
ALLOCATOR_PROPERTIES * *pProps*
);

Parameters

pProps
Structure to be filled in with allocator properties.

Return Values

Returns an HRESULT value. The default implementation returns NOERROR.

Remarks

This member function implements the IMemAllocator::GetProperties method. The default implementation fills the ALLOCATOR_PROPERTIES structure passed in with the values of m_lSize, m_lCount, m_lAlignment, and m_lPrefix.

Previous Home Topic Contents Index Next

CBaseAllocator::NonDelegatingQueryInterface

CBaseAllocator Class

Retrieves an interface and increments the reference count.

HRESULT NonDelegatingQueryInterface(
REFIID *riid*,
void ** *ppv*
);

Parameters

riid
Reference identifier.
ppv
Pointer to the interface.

Return Values

Returns E_POINTER if *ppv* is invalid. Returns NOERROR if the query is successful or E_NOINTERFACE if it is not.

Remarks

This member function implements the INonDelegatingUnknown::NonDelegatingQueryInterface method and passes out references to the IMemAllocator and IUnknown interfaces. Override this member function to return other interfaces on the object in the derived class.

Previous Home Topic Contents Index Next

CBaseAllocator::NotifySample

CBaseAllocator Class

Notifies a waiting thread that a sample is available on the free list.

void NotifySample(void);

Return Values

No return value.

Remarks

If m_lWaiting has been incremented (is not zero), this indicates a thread is waiting. This member function checks for this condition and calls the Microsoft Win32 ReleaseSemaphore function with the semaphore value m_hSem to activate any waiting thread. It also sets **m_lWaiting** back to zero.

This member function is called from CBaseAllocator::ReleaseBuffer when putting a sample back on the free list and from CBaseAllocator::Decommit when decommitting the allocator (so that waiting threads can be denied).

Previous Home Topic Contents Index Next

CBaseAllocator::ReleaseBuffer

CBaseAllocator Class

Releases the object back to the list of free objects.

HRESULT ReleaseBuffer(

IMediaSample * *pSample*
);

Parameters

pSample
Pointer to the IMediaSample interface of the media sample object.

Return Values

No return value.

Remarks

This member function implements the IMemAllocator::ReleaseBuffer method. It adds the sample to the free list (represented by m_lFree) and calls CBaseAllocator::NotifySample to notify any blocked thread waiting for a free sample. If there is a pending CBaseAllocator::Decommit call (indicated by m_bDecommitInProgress), the pure virtual CBaseAllocator::Free member function is called to decommit memory when the last buffer is placed on the free list.

Previous Home Topic Contents Index Next

CBaseAllocator::SetProperties

CBaseAllocator Class

Determines the size, number, and alignment of blocks.

HRESULT SetProperties(
ALLOCATOR_PROPERTIES * *pRequest*,
ALLOCATOR_PROPERTIES * *pActual*
);

Parameters

pRequest
Allocator properties requested to be set.
pActual
Allocator properties actually set.

Return Values

Returns an HRESULT value.

Remarks

The *pRequest* parameter is filled in by the caller with the requested values for the count, number, and alignment as specified by the ALLOCATOR_PROPERTIES structure. The *pActual* parameter is filled in by the allocator with the closest values that it can provide for the request. This method cannot be called unless the allocator has been decommitted using the IMemAllocator::Decommit method.

The values of data members m_lSize, m_lCount, m_lAlignment, and m_lPrefix are set to the corresponding members of the *pActual* parameter's ALLOCATOR_PROPERTIES structure.

Previous Home Topic Contents Index Next

CBaseAllocator::SetWaiting

CBaseAllocator Class

Increments the m_lWaiting data member to indicate that a thread is waiting for a sample.

void SetWaiting( );

Return Values

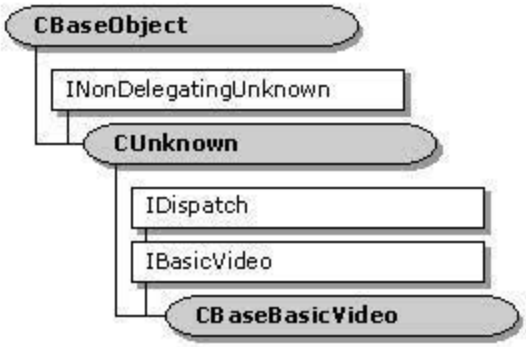
No return value.

Remarks

This member function is called from CBaseAllocator::GetBuffer if no samples are available on the free list. After calling this, **CBaseAllocator::GetBuffer** calls the Microsoft® Win32® WaitForSingleObject function to wait for the list to signal that a sample is available. The list signals by calling CBaseAllocator::ReleaseBuffer, which in turn calls CBaseAllocator::NotifySample, which sets m_lWaiting to zero and calls the Win32 ReleaseSemaphore function.

Previous Home Topic Contents Index Next

CBaseBasicVideo Class



The **CBaseBasicVideo** class handles the IDispatch component of the IBasicVideo interface and leaves the properties and methods pure virtual.

The IDispatch::GetIDsOfNames, IDispatch::GetTypeInfo, IDispatch::GetTypeInfoCount, and IDispatch::Invoke methods are standard implementations of the IDispatch interface using the CBaseDispatch class (and a type library) to parse the commands and pass them to the pure virtual methods of the IBasicVideo interface.

Member Functions

Name	Description
CBaseBasicVideo	Constructs a CBaseBasicVideo object.

Implemented INonDelegatingUnknown Methods

Name	Description
NonDelegatingQueryInterface	Returns a specified reference-counted interface.

Implemented IDispatch Methods

Name	Description
GetIDsOfNames	Maps a single member and an optional set of parameters to a corresponding set of integer dispatch identifiers, which can be used during subsequent calls to the IDispatch::Invoke method.
GetTypeInfo	Retrieves a type-information object, which can retrieve the type information for an interface.
GetTypeInfoCount	Retrieves the number of type-information interfaces provided by an object.
Invoke	Provides access to properties and methods exposed by an object.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseBasicVideo::CBaseBasicVideo

CBaseBasicVideo Class

Constructs a CBaseBasicVideo object.

CBaseBasicVideo(
const TCHAR * *pName*,
LPUNKNOWN *pUnk*
);

Parameters

pName
Name of the object for debugging purposes.
pUnk
Pointer to the owner of this object.

Return Values

No return value.

Remarks

Allocate the *pName* parameter in static memory. This name appears on the debugging terminal upon creation and deletion of the object.

Previous Home Topic Contents Index Next

CBaseBasicVideo::GetIDsOfNames

CBaseBasicVideo Class

Maps a single member function and an optional set of parameters to a corresponding set of integer dispatch identifiers, which can be used upon subsequent calls to the CBaseBasicVideo::Invoke member function.

HRESULT GetIDsOfNames(

REFIID *riid*,
OLECHAR ** *rgszNames*,
UINT *cNames*,
LCID *lcid*,
DISPID * *rgdispid*
);

Parameters

riid
Reference identifier. Reserved for future use. Must be NULL.
rgszNames
Passed-in array of names to be mapped.
cNames
Count of the names to be mapped.
lcid
Locale context in which to interpret the names.
rgdispid
Caller-allocated array, each element of which contains an ID corresponding to one of the names passed in the *rgszNames* array. The first element represents the member name; the subsequent elements represent each of the member's parameters.

Return Values

Returns one of the following values.

Value	Meaning
S_OK	Success.
E_OUTOFMEMORY	Out of memory.
DISP_E_UNKNOWNNAME	One or more of the names were not known. The returned DISPIDs contain DISPID_UNKNOWN for each entry that corresponds to an unknown name.
DISP_E_UNKNOWN_CLSID	The CLSID was not recognized.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseBasicVideo::GetTypeInfo

CBaseBasicVideo Class

Retrieves a type-information object, which can retrieve the type information for an interface.

HRESULT GetTypeInfo(
UINT *itinfo*,

LCID *lcid*,
ITypeInfo ** *pptinfo*
);

Parameters

itinfo
Type information to return. Pass zero to retrieve type information for the IDispatch implementation.
lcid
Locale ID for the type information. An object might be able to return different type information for different languages. This is important for classes that support localized member names. For classes that do not support localized member names, this parameter can be ignored.
pptinfo
Pointer to the type-information object requested.

Return Values

Returns an E_POINTER if *pptinfo* is invalid. Returns TYPE_E_ELEMENTNOTFOUND if *itinfo* is not zero. Returns S_OK if is successful. Otherwise, returns an HRESULT from one of the calls to retrieve the type. The **HRESULT** indicates the error and can be one of the following standard constants, or other values not listed:

Value	Meaning
E_FAIL	Failure.
E_POINTER	Null pointer argument.
E_INVALIDARG	Invalid argument.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseBasicVideo::GetTypeInfoCount

CBaseBasicVideo Class

Retrieves the number of type-information interfaces provided by an object.

HRESULT GetTypeInfoCount(
UINT * *pctinfo*
);

Parameters

pctinfo

Pointer to the location that receives the number of type-information interfaces that the object provides. If the object provides type information, this number is 1; otherwise, the number is 0.

Return Values

Returns E_POINTER if *pctinfo* is invalid; otherwise, returns S_OK.

‹Previous Home Topic Contents Index Next›

CBaseBasicVideo::Invoke

CBaseBasicVideo Class

Provides access to properties and methods exposed by an object.

HRESULT Invoke(
DISPID *dispidMember*,
REFIID *riid*,
LCID *lcid*,
WORD *wFlags*,
DISPPARAMS * *pdispparams*,
VARIANT * *pvarResult*,
EXCEPINFO * *pexcepinfo*,
UINT * *puArgErr*
);

Parameters

dispidMember
Identifier of the member. Use CBaseBasicVideo::GetIDsOfNames or the object's documentation to obtain the dispatch identifier.
riid
Reserved for future use. Must be IID_NULL.
lcid
Locale context in which to interpret arguments.
wFlags
Flags describing the context of the **CBaseBasicVideo::Invoke** call.
pdispparams
Pointer to a structure containing an array of arguments, an array of argument dispatch IDs for named arguments, and counts for number of elements in the arrays.
pvarResult
Pointer to where the result is to be stored, or NULL if the caller expects no result.
pexcepinfo
Pointer to a structure containing exception information.
puArgErr
Index of the first argument, within the *rgvarg* array, that has an error.

Return Values

Returns DISP_E_UNKNOWNINTERFACE if *riid* is not IID_NULL. Returns one of the error codes from CBaseBasicVideo::GetTypeInfo if the call fails. Otherwise, returns the HRESULT from the call to IDispatch::Invoke.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseBasicVideo::NonDelegatingQueryInterface

CBaseBasicVideo Class

Returns a specified reference-counted interface.

```
HRESULT NonDelegatingQueryInterface(
  REFIID riid,
  void **ppv
  );
```

Parameters

riid
Reference identifier.

ppv
Pointer to the interface.

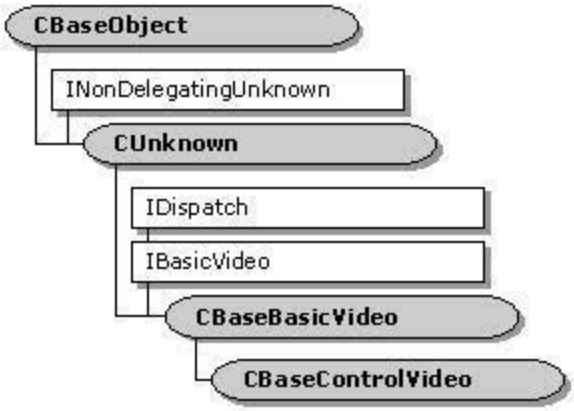
Return Values

Returns E_POINTER if *ppv* is invalid. Returns NOERROR if the query is successful or E_NOINTERFACE if it is not.

Remarks

Returns pointers to the IBasicVideo and IUnknown interfaces by default. Override this method to publish any additional interfaces implemented by the derived class.

CBaseControlVideo Class



The **CBaseControlVideo** class implements the IBasicVideo dual interface and controls the video properties of a generic video window. Generally, a **CBaseControlVideo** object is a video renderer that draws video into a window on the display.

The **CBaseControlVideo** class supports both properties and methods. Properties are more easily accessible from many Automation controllers (such as the Microsoft® Visual Basic® programming system). However, some operations require applications to be able to change several properties simultaneously; for this reason, methods are provided that enable a number of related properties to be changed.

Many **CBaseControlVideo** member functions require only that the video renderer be connected to a filter graph. If it is not connected, member functions will return VFW_E_NOT_CONNECTED. Properties set on a video renderer persist between successive connections and disconnections. All applications should ensure that they reset the renderer properties before starting a presentation.

When working with video, the application can select a portion of the video to use. This portion is the source rectangle that the **CBaseControlVideo** object controls. **CBaseControlVideo** enables your application to set and retrieve the source rectangle. All the rectangles that **CBaseControlVideo** uses employ top, left, width, and height rather than top, left, right, and bottom, which is favored in Microsoft Win32® programming. When no source rectangle has been set, the properties of the source rectangle return the full, native video size.

Protected Data Members

Name	Description
m_pFilter	Pointer to an owning media filter.
m_pInterfaceLock	Externally defined critical section.
m_pPin	Control of the media types for connection.

Member Functions

Name	Description
CBaseControlVideo	Constructs a CBaseControlVideo object.
CopyImage	Creates a memory copy of a video image.
GetImageSize	Retrieves video image size information.
SetControlVideoPin	Sets the pin with which this object should synchronize.

Overridable Member Functions

Name	Description
CheckSourceRect	Determines if a source rectangle is valid.
CheckTargetRect	Determines if a target rectangle is valid.
GetSourceRect	Retrieves the current source video rectangle (pure virtual).
GetStaticImage	Returns the current image in a memory buffer (pure virtual).
GetTargetRect	Retrieves the current target video rectangle (pure virtual).
GetVideoFormat	Retrieves the VIDEOINFOHEADER structure containing the video format.
IsDefaultSourceRect	Determines if the renderer is using the default source rectangle (pure virtual).
IsDefaultTargetRect	Determines if the renderer is using the default target rectangle (pure virtual).
OnUpdateRectangles	Called when the source or target rectangle changes.
OnVideoSizeChange	Passes EC_VIDEO_SIZE_CHANGED to the application.
SetDefaultSourceRect	Sets the default source video rectangle (pure virtual).
SetDefaultTargetRect	Sets the default target video rectangle (pure virtual).
SetSourceRect	Sets the current source video rectangle (pure virtual).
SetTargetRect	Sets the current target rectangle (pure virtual).

Implemented IBasicVideo Methods

Name	Description
get_AvgTimePerFrame	Returns an approximate average time per frame.
get_BitErrorRate	Returns an approximate bit error rate.
get_BitRate	Returns an approximate bit rate for the video.
GetCurrentImage	Returns a memory rendering of the current image.
get_DestinationHeight	Retrieves the current destination rectangle's height.
get_DestinationLeft	Retrieves the current destination rectangle's left coordinate.
GetDestinationPosition	Retrieves the current destination position.
get_DestinationTop	Retrieves the current destination rectangle's top coordinate.
get_DestinationWidth	Retrieves the current destination rectangle's width.
get_SourceHeight	Retrieves the current source rectangle's height.
get_SourceLeft	Retrieves the current source rectangle's left coordinate.
GetSourcePosition	Retrieves the current source position.
get_SourceTop	Retrieves the current source rectangle's top coordinate.
get_SourceWidth	Retrieves the current source rectangle's width.
get_VideoHeight	Retrieves the native video height.
GetVideoPaletteEntries	Retrieves a range of palette entries for the video.
GetVideoSize	Retrieves the width and height of the native video.
get_VideoWidth	Retrieves the native video width.

IsUsingDefaultDestination	Determines if the renderer is using the default destination window.
IsUsingDefaultSource	Determines if the renderer is using the default source window.
put_DestinationHeight	Sets the destination rectangle's height.
put_DestinationLeft	Sets the destination rectangle's left coordinate.
put_DestinationTop	Sets the destination rectangle's top coordinate.
put_DestinationWidth	Sets the destination rectangle's width.
put_SourceHeight	Sets the source rectangle's height.
put_SourceLeft	Sets the source rectangle's left coordinate.
put_SourceTop	Sets the source rectangle's top coordinate.
put_SourceWidth	Sets the source rectangle's width.
SetDefaultDestinationPosition	Sets the default destination position again.
SetDefaultSourcePosition	Sets the default source position again.
SetDestinationPosition	Sets the destination rectangle position.
SetSourcePosition	Sets the source rectangle position.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseControlVideo::CBaseControlVideo

CBaseControlVideo Class

Constructs a CBaseControlVideo object.

```
CBaseControlVideo(
  CBaseFilter *pFilter,
  CCritSec *pInterfaceLock,
  TCHAR *pName,
  LPUNKNOWN pUnk,
  HRESULT *phr
  );
```

Parameters

pFilter
Owning media filter object.
pInterfaceLock
Critical section to use for locking.
pName
Object description.
pUnk
Typical COM ownership.

phr
COM return value.

Return Values

No return value.

Remarks

The object implements the IBasicVideo control interface.

All the interface methods from IBasicVideo that this class implements require that the filter be connected correctly. For this reason, the class is passed a pin with which it should synchronize with. Whenever an interface method is called, the object determines that the pin is still connected.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseControlVideo::CheckSourceRect

CBaseControlVideo Class

Determines if a source rectangle is valid.

virtual HRESULT CheckSourceRect(
RECT **pSourceRect*
);

Parameters

pSourceRect
Source rectangle to check.

Return Values

Returns E_INVALIDARG if not valid; otherwise, returns NOERROR (S_OK).

Remarks

This member function checks that the source rectangle requested does not exceed the available source video. The left and top coordinates cannot be negative, and the width and height cannot exceed the right and bottom of the video.

Previous Home Topic Contents Index Next

CBaseControlVideo::CheckTargetRect

CBaseControlVideo Class

Determines if a target rectangle is valid.

virtual HRESULT CheckTargetRect(
RECT *pTargetRect*
);

Parameters

pTargetRect
Target rectangle to check.

Return Values

Returns E_INVALIDARG if not valid; otherwise, returns NOERROR (S_OK).

Remarks

This member function determines if the target rectangle requested is valid. Because the destination rectangle specifies a position in the logical client of the window, the coordinates can be negative, although the overall width and height cannot be zero or a negative value.

Previous Home Topic Contents Index Next

CBaseControlVideo::CopyImage

CBaseControlVideo Class

Creates a memory copy of an image.

HRESULT CopyImage(
IMediaSample *pMediaSample*,
VIDEOINFOHEADER *pVideoInfo*,
LONG *pBufferSize*,
BYTE *pVideoImage*,

RECT **pSourceRect*
);

Parameters

pMediaSample
Sample containing the video image.
pVideoInfo
Format representing the video image.
pBufferSize
Size of the output buffer.
pVideoImage
Pointer to the output buffer.
pSourceRect
Source video rectangle.

Return Values

If the *pVideoImage* parameter is NULL, the *pBufferSize* parameter is filled in with the number of bytes the output buffer requires to store the image. If the buffer passed in is too small or the member function fails to allocate sufficient memory, the member function returns E_OUTOFMEMORY.

Remarks

The member function retrieves the image from the sample and copies it into the output buffer. The section of video copied into the output buffer reflects the source rectangle that is set through the IBasicVideo interface (although it does not reflect the destination rectangle).

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseControlVideo::get_AvgTimePerFrame

CBaseControlVideo Class

Retrieves the average time per frame.

HRESULT get_AvgTimePerFrame(
REFTIME **pAvgTimePerFrame*
);

Parameters

pAvgTimePerFrame

Average time per frame.

Return Values

Returns NOERROR if successful or E_OUTOFMEMORY if there is not enough memory available.

Remarks

This member function implements the IBasicVideo::get_AvgTimePerFrame method. It calls the pure virtual CBaseControlVideo::GetVideoFormat member function to retrieve the VIDEOINFOHEADER structure from the derived class.

Previous Home Topic Contents Index Next

CBaseControlVideo::get_BitErrorRate

CBaseControlVideo Class

Returns an approximate bit error rate for the video.

HRESULT get_BitErrorRate(
long **pBitErrorRate*
);

Parameters

pBitErrorRate
Bit error rate (one error for approximately this many bits).

Return Values

Returns NOERROR if successful or E_OUTOFMEMORY if there is not enough memory available.

Remarks

This member function implements the IBasicVideo::get_BitErrorRate method. It calls the pure virtual CBaseControlVideo::GetVideoFormat to retrieve the VIDEOINFOHEADER structure from the derived class.

Previous Home Topic Contents Index Next

CBaseControlVideo::get_BitRate

CBaseControlVideo Class

Returns an approximate bit rate for the video.

HRESULT get_BitRate(
long *pBitRate*
);

Parameters

pBitRate
Bit rate in bits per second.

Return Values

Returns NOERROR if successful or E_OUTOFMEMORY not enough memory is available.

Remarks

This member function implements the IBasicVideo::get_BitRate method. It calls the pure virtual CBaseControlVideo::GetVideoFormat to retrieve the VIDEOINFOHEADER structure from the derived class.

Previous Home Topic Contents Index Next

CBaseControlVideo::GetCurrentImage

CBaseControlVideo Class

Returns a copy of the current image at the renderer.

HRESULT GetCurrentImage(
long *pBufferSize*,
long *pVideoImage*
);

Parameters

pBufferSize
Size of the output buffer.
pVideoImage
Pointer to the output buffer for the image.

Return Values

Returns an HRESULT value.

Remarks

This member function retrieves the image from the sample and copies it into the output buffer. The section of video copied into the output buffer reflects the source rectangle set through the IBasicVideo interface. It does not reflect the destination rectangle.

Previous Home Topic Contents Index Next

CBaseControlVideo::get_DestinationHeight

CBaseControlVideo Class

Retrieves the current destination rectangle height.

HRESULT get_DestinationHeight(
long *pDestinationHeight*
);

Parameters

pDestinationHeight
Holds the destination height.

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IBasicVideo::get_DestinationHeight method.

An application can change the source and destination rectangles for the video through the IBasicVideo interface. The source rectangle affects which section of the native video source will appear on the display; the destination rectangle affects where it will be played. The destination rectangle is relative to the client area of the window that it is playing in. The upper-left corner of the window is coordinate (0,0).

Previous Home Topic Contents Index Next

CBaseControlVideo::get_DestinationLeft

CBaseControlVideo Class

Retrieves the left coordinate of the current destination rectangle.

HRESULT get_DestinationLeft(
long *pDestinationLeft*
);

Parameters

pDestinationLeft
Contains the left coordinate of the destination rectangle.

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IBasicVideo::get_DestinationLeft method.

An application can change the source and destination rectangles for the video through the IBasicVideo interface. The source rectangle affects which section of the native video source will appear on the display; the destination rectangle affects where the video will appear when played. The destination rectangle is relative to the client area of the window in which it is playing. The upper-left corner of the window is coordinate (0,0).

Previous Home Topic Contents Index Next

CBaseControlVideo::GetDestinationPosition

CBaseControlVideo Class

Retrieves the destination rectangle.

HRESULT GetDestinationPosition(
long *pLeft*,
long *pTop*,
long *pWidth*,

long **pHeight*
);

Parameters

pLeft
Contains the left coordinate.
pTop
Contains the top coordinate.
pWidth
Contains the width.
pHeight
Contains the height.

Return Values

Returns an HRESULT value.

Remarks

This member function can be used in place of separate calls to the CBaseControlVideo::get_DestinationLeft, CBaseControlVideo::get_DestinationTop, CBaseControlVideo::get_DestinationWidth, and CBaseControlVideo::get_DestinationHeight member functions. An application can change the source and destination rectangles for the video through the IBasicVideo interface. The source rectangle affects which section of the native video source will appear on the display; the destination rectangle affects where the video will appear when played. The destination rectangle is relative to the client area of the window in which it is playing. The upper-left corner of the window is coordinate (0,0).

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseControlVideo::get_DestinationTop

CBaseControlVideo Class

Retrieves the top coordinate of the current destination rectangle.

HRESULT get_DestinationTop(
long **pDestinationTop*
);

Parameters

pDestinationTop

Contains the top coordinate of the destination rectangle.

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IBasicVideo::get_DestinationTop method.

An application can change the source and destination rectangles for the video through the IBasicVideo interface. The source rectangle affects which section of the native video source will appear on the display; the destination rectangle affects where the video will appear when played. The destination rectangle is relative to the client area of the window in which it is playing. The upper-left corner of the window is coordinate (0,0).

Previous Home Topic Contents Index Next

CBaseControlVideo::get_DestinationWidth

CBaseControlVideo Class

Retrieves the width of the current destination rectangle.

HRESULT get_DestinationWidth(
long *pDestinationWidth*
);

Parameters

pDestinationWidth
Contains the destination width.

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IBasicVideo::get_DestinationWidth method.

An application can change the source and destination rectangles for the video through the IBasicVideo interface. The source rectangle affects which section of the native video source will appear on the display; the destination rectangle affects where the video will appear when played. The destination rectangle is relative to the client area of the window in which it is playing. The upper-left corner of the window is coordinate (0,0).

Previous Home Topic Contents Index Next

CBaseControlVideo::GetImageSize

CBaseControlVideo Class

Retrieves video image size information.

```
HRESULT GetImageSize(
  VIDEOINFOHEADER *pVideoInfo,
  long *pBufferSize,
  RECT *pSourceRect
  );
```

Parameters

pVideoInfo
Contains a pointer to a VIDEOINFOHEADER structure to be filled in.
pBufferSize
Size of the video buffer.
pSourceRect
Rectangle dimensions of the source video.

Return Values

Returns an HRESULT value.

Remarks

This member function is a helper function used for creating memory image renderings of DIB images. It is called from the base class implementation of CBaseControlVideo::GetCurrentImage when a null *pVideoImage* parameter is passed to that member function. As a result, this member function constructs and returns a VIDEOINFOHEADER structure, using the information in *pBufferSize* and *pSourceRect*.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseControlVideo::get_SourceHeight

CBaseControlVideo Class

Retrieves the height of the current source rectangle.

HRESULT get_SourceHeight(
long **pSourceHeight*
);

Parameters

pSourceHeight
Contains the height of the source rectangle.

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IBasicVideo::get_SourceHeight method.

An application can change the source and destination rectangles for the video through the IBasicVideo interface. The source rectangle affects which section of the native video source will appear on the display; the destination rectangle affects where the video will appear when played. The destination rectangle is relative to the client area of the window in which it is playing. The upper-left corner of the window is coordinate (0,0).

Previous Home Topic Contents Index Next

CBaseControlVideo::get_SourceLeft

CBaseControlVideo Class

Retrieves the left coordinate of the current source rectangle.

HRESULT get_SourceLeft(
long **pSourceLeft*
);

Parameters

pSourceLeft

Holds the left coordinate of the current source rectangle.

Return Values

Returns an HRESULT value.

Remarks

An application can change the source and destination rectangles for the video through the IBasicVideo interface. The source rectangle affects which section of the native video source will appear on the display; the destination rectangle affects where the video will appear when played. The destination rectangle is relative to the client area of the window in which it is playing. The upper-left corner of the window is coordinate (0,0).

CBaseControlVideo::GetSourcePosition

CBaseControlVideo Class

Retrieves the source rectangle in one atomic operation.

```
HRESULT GetSourcePosition(
  long *pLeft,
  long *pTop,
  long *pWidth,
  long *pHeight
  );
```

Parameters

pLeft
Contains the left coordinate.
pTop
Contains the top coordinate.
pWidth
Contains the width.
pHeight
Contains the height.

Return Values

Returns an HRESULT value.

Remarks

An application can change the source and destination rectangles for the video through the IBasicVideo interface. The source rectangle affects which section of the native video source will appear on the display; the destination rectangle affects where the video will appear when played. The destination rectangle is relative to the client area of the window in which it is playing. The upper-left corner of the window is coordinate (0,0).

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseControlVideo::GetSourceRect

CBaseControlVideo Class

Returns the source rectangle. This is an internal method.

virtual HRESULT GetSourceRect(
RECT *pSourceRect*
) PURE;

Parameters

pSourceRect
Contains the retrieved source rectangle.

Return Values

Returns an HRESULT value.

Remarks

This member function must be overridden in the derived class to return the source rectangle held by the video renderer. It is called from the following CBaseControlVideo member functions.

- CBaseControlVideo::GetSourcePosition
- CBaseControlVideo::put_SourceLeft
- CBaseControlVideo::get_SourceLeft
- CBaseControlVideo::put_SourceWidth
- CBaseControlVideo::get_SourceWidth
- CBaseControlVideo::put_SourceTop
- CBaseControlVideo::get_SourceTop
- CBaseControlVideo::put_SourceHeight
- CBaseControlVideo::get_SourceHeight

The following example from the video renderer sample, SampVid, demonstrates an implementation of this function in a derived class.

```
// Return the current source rectangle

HRESULT CVideoText::GetSourceRect(RECT *pSourceRect)
{
    ASSERT(pSourceRect);
    m_pRenderer->m_DrawImage.GetSourceRect(pSourceRect);
    return NOERROR;
}
```

In this example, CVideoText is a class derived from CBaseControlVideo, m_pRenderer holds an object of a class derived from CBaseVideoRenderer, and the m_DrawImage data member, defined in the derived class, holds a CDrawImage object.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseControlVideo::get_SourceTop

CBaseControlVideo Class

Retrieves the top coordinate of the current source rectangle.

HRESULT get_SourceTop(
long **pSourceTop*
);

Parameters

pSourceTop
Contains the top coordinate of the source rectangle.

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IBasicVideo::get_SourceTop method.

An application can change the source and destination rectangles for the video through the IBasicVideo interface. The source rectangle affects which section of the native video source will

appear on the display; the destination rectangle affects where the video will appear when played. The destination rectangle is relative to the client area of the window in which it is playing. The upper-left corner of the window is coordinate (0,0).

Previous Home Topic Contents Index Next

CBaseControlVideo::get_SourceWidth

CBaseControlVideo Class

Retrieves the width of the current source rectangle.

HRESULT get_SourceWidth(
long **pSourceWidth*
);

Parameters

pSourceWidth
Contains the width of the current source rectangle.

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IBasicVideo::get_SourceWidth method.

An application can change the source and destination rectangles for the video through the IBasicVideo interface. The source rectangle affects which section of the native video source will appear on the display; the destination rectangle affects where the video will appear when played. The destination rectangle is relative to the client area of the window in which it is playing. The upper-left corner of the window is coordinate (0,0).

Previous Home Topic Contents Index Next

CBaseControlVideo::GetStaticImage

CBaseControlVideo Class

Pure virtual method that derived classes override.

virtual HRESULT GetStaticImage(
long **pBufferSize*,
long **pDIBImage*
) PURE;

Parameters

pBufferSize
Size of the output buffer.
pDIBImage
Pointer to output buffer.

Return Values

Returns an HRESULT value.

Remarks

Through the IBasicVideo interface, an application can request that it be given a copy of the current image in a memory buffer (some renderers can return E_NOTIMPL to this if they do not support it). The derived class determines how to retrieve the image. When the application calls **CBaseControlVideo::GetStaticImage**, it calls this pure virtual method that the derived class should override to implement it. This is also called by the CBaseControlVideo::GetCurrentImage member function.

The class provides a helper member function, CBaseControlVideo::CopyImage, that can be given a sample that contains an image, and the member function will copy the relevant section of it (based on the current source rectangle) into the output buffer supplied by the application.

The following example from the video renderer sample, SampVid, demonstrates an implementation of this member function in a derived class. In this example, m_pRenderer holds an object of a class derived from CBaseVideoRenderer.

```
// Return a copy of the current image in the video renderer

HRESULT CVideoText::GetStaticImage(long *pBufferSize,long *pDIBImage)
{
    // Get any sample the renderer may be holding

    IMediaSample *pMediaSample = m_pRenderer->GetCurrentSample();
    if (pMediaSample == NULL) {
        return E_UNEXPECTED;
    }

    // Call the base class helper method to do the work

    HRESULT hr = CopyImage(pMediaSample,          // Buffer containing image
                           &m_pRenderer->m_mtIn,                // Type representin
                           pBufferSize,                                  // Size
                           (BYTE*) pDIBImage);                   // Data buffer for

    pMediaSample->Release();
    return hr;
```

```
}
```


Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseControlVideo::GetTargetRect

CBaseControlVideo Class

Returns the destination rectangle. This is an internal helper member function.

virtual HRESULT GetTargetRect(
RECT *pTargetRect*
) PURE;

Parameters

pTargetRect
Contains the destination rectangle.

Return Values

Returns an HRESULT value.

Remarks

This member function must be overridden in the derived class to return the target rectangle held by the video renderer. It is called from the following CBaseControlVideo member functions.

- CBaseControlVideo::GetDestinationPosition
- CBaseControlVideo::put_DestinationLeft
- CBaseControlVideo::get_DestinationLeft
- CBaseControlVideo::put_DestinationWidth
- CBaseControlVideo::get_DestinationWidth
- CBaseControlVideo::put_DestinationTop
- CBaseControlVideo::get_DestinationTop
- CBaseControlVideo::put_DestinationHeight
- CBaseControlVideo::get_DestinationHeight

The following example from the video renderer sample, SampVid, demonstrates an implementation of this function in a derived class.

```
// Return the current destination rectangle

HRESULT CVideoText::GetTargetRect(RECT *pTargetRect)
{
    ASSERT(pTargetRect);
    m_pRenderer->m_DrawImage.GetTargetRect(pTargetRect);
    return NOERROR;
}
```

In this example, CVideoText is a class derived from CBaseControlVideo, m_pRenderer holds an object of a class derived from CBaseVideoRenderer, and the m_DrawImage data member, defined in the derived class, holds a CDrawImage object.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseControlVideo::GetVideoFormat

CBaseControlVideo Class

Retrieves a video sample that represents the current video format.

virtual VIDEOINFOHEADER * GetVideoFormat( ) PURE;

Return Values

Returns a pointer to a VIDEOINFOHEADER structure that contains the current video format.

Remarks

To return and check certain information through IBasicVideo, the object must know the current video format. It gets this information by calling this pure virtual method that derived classes must override. This member function is called by the following CBaseControlVideo member functions.

- CBaseControlVideo::OnVideoSizeChange
- CBaseControlVideo::get_AvgTimePerFrame
- CBaseControlVideo::get_BitRate
- CBaseControlVideo::get_BitErrorRate
- CBaseControlVideo::get_VideoWidth
- CBaseControlVideo::get_VideoHeight
- CBaseControlVideo::GetVideoPaletteEntries
- CBaseControlVideo::GetVideoSize

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseControlVideo::get_VideoHeight

CBaseControlVideo Class

Retrieves the height of the native video.

HRESULT get_VideoHeight(
long **pVideoHeight*
);

Parameters

pVideoHeight
Contains the height of the native video, in pixels.

Return Values

Returns NOERROR if successful or E_OUTOFMEMORY if there is not enough memory available.

Remarks

This member function implements the IBasicVideo::get_VideoHeight method. It calls the pure virtual CBaseControlVideo::GetVideoFormat to retrieve the VIDEOINFOHEADER structure from the derived class.

Previous Home Topic Contents Index Next

CBaseControlVideo::GetVideoPaletteEntries

CBaseControlVideo Class

Retrieves a range of palette entries for the video.

HRESULT GetVideoPaletteEntries(
long *StartIndex*,
long *Entries*,

long **pRetrieved*,
long **pPalette*
);

Parameters

StartIndex
Zero-based start palette entry.
Entries
Number of entries required.
pRetrieved
Number of colors obtained.
pPalette
Pointer to output buffer for colors.

Return Values

Returns NOERROR if successful, VFW_E_NO_PALETTE_AVAILABLE if the video samples has no color palette, E_OUTOFMEMORY if there is not enough memory available, E_INVALIDARG if *StartIndex* is invalid, or S_FALSE if there are no colors in the palette.

Remarks

This member function returns the current palette of the video as an array allocated by the user. To remain consistent, use the members in the Win32 PALETTEENTRY structure to return the colors, rather than the members in the RGBQUAD structure (although the parameter is a LONG). The memory is allocated by the caller, so simply copy each in turn. Determine that the number of entries requested and the start position offset are both valid. If the number of entries evaluates to zero, return an S_FALSE code.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseControlVideo::GetVideoSize

CBaseControlVideo Class

Retrieves the native video's width and height.

HRESULT GetVideoSize(
long **pWidth*,
long **pHeight*
);

Parameters

pWidth
Contains the video width.
pHeight
Contains the video height.

Return Values

Returns an HRESULT value.

Previous Home Topic Contents Index Next

CBaseControlVideo::get_VideoWidth

CBaseControlVideo Class

Retrieves the width of the native video.

```
HRESULT get_VideoWidth(
  long *pVideoWidth
  );
```

Parameters

pVideoWidth
Contains the width of the native video, in pixels.

Return Values

Returns NOERROR if successful or E_OUTOFMEMORY if there is not enough memory available.

Remarks

This member function implements the IBasicVideo::get_VideoWidth method. It calls the pure virtual CBaseControlVideo::GetVideoFormat to retrieve the VIDEOINFOHEADER structure from the derived class.

Previous Home Topic Contents Index Next

CBaseControlVideo::IsDefaultSourceRect

CBaseControlVideo Class

Determines if the renderer is using the default source rectangle (pure virtual).

virtual HRESULT IsDefaultSourceRect(void) PURE;

Return Values

Returns an HRESULT value.

Remarks

This member function must be implemented in the derived class. It is called by the CBaseControlVideo::IsUsingDefaultSource member function.

The following example from the video renderer sample, SampVid, demonstrates an implementation of this function in a derived class.

```
// Return S_OK if using the default source otherwise S_FALSE

HRESULT CVideoText::IsDefaultSourceRect()
{
    RECT SourceRect;

    VIDEOINFO *pVideoInfo = (VIDEOINFO *) m_pRenderer->m_mtIn.Format();
    BITMAPINFOHEADER *pHeader = HEADER(pVideoInfo);
    m_pRenderer->m_DrawImage.GetSourceRect(&SourceRect);

    // Check the coordinates that match the video dimensions

    if (SourceRect.left != 0 || SourceRect.top != 0 ||
            SourceRect.right != pHeader->biWidth ||
                SourceRect.bottom != pHeader->biHeight) {
                    return S_FALSE;
    }
    return S_OK;
}
```

In this example, CVideoText is a class derived from CBaseControlVideo, m_pRenderer holds an object of a class derived from CBaseVideoRenderer, and the m_DrawImage data member, defined in the derived class, holds a CDrawImage object. The m_mtIn data member, also defined in the derived class, holds a CMediaType object with the media type of the input pin.

Previous Home Topic Contents Index Next

CBaseControlVideo::IsDefaultTargetRect

CBaseControlVideo Class

Determines if the renderer is using the default target rectangle (pure virtual).

virtual HRESULT IsDefaultTargetRect(void) PURE;

Return Values

Returns an HRESULT value.

Remarks

This member function must be implemented in the derived class. It is called by the CBaseControlVideo::IsUsingDefaultDestination member function.

The following example from the video renderer sample, SampVid, demonstrates an implementation of this function in a derived class.

```
// Return S_OK if using the default target; otherwise S_FALSE

HRESULT CVideoText::IsDefaultTargetRect()
{
    RECT TargetRect;

    VIDEOINFO *pVideoInfo = (VIDEOINFO *) m_pRenderer->m_mtIn.Format();
    BITMAPINFOHEADER *pHeader = HEADER(pVideoInfo);
    m_pRenderer->m_DrawImage.GetTargetRect(&TargetRect);

    // Check the destination that matches the initial client area

    if (TargetRect.left != 0 || TargetRect.top != 0 ||
            TargetRect.right != m_Size.cx ||
                TargetRect.bottom != m_Size.cy) {
                    return S_FALSE;
    }
    return S_OK;
}
```

In this example, CVideoText is a class derived from CBaseControlVideo, m_pRenderer holds an object of a class derived from CBaseVideoRenderer, and the m_DrawImage data member, defined in the derived class, holds a CDrawImage object. The m_mtIn data member, also defined in the derived class, holds a CMediaType object with media type of the input pin.

Previous Home Topic Contents Index Next

CBaseControlVideo::IsUsingDefaultSource

CBaseControlVideo Class

Determines if the renderer is using the default source window.

virtual HRESULT IsUsingDefaultSource(void);

Return Values

Returns an HRESULT value.

Previous Home Topic Contents Index Next

CBaseControlVideo::IsUsingDefaultDestination

CBaseControlVideo Class

Determines if the renderer is using the default destination window.

virtual HRESULT IsUsingDefaultDestination(void);

Return Values

Returns an HRESULT value. Returns S_OK if using the default destination; otherwise, returns S_FALSE.

Previous Home Topic Contents Index Next

CBaseControlVideo::OnUpdateRectangles

CBaseControlVideo Class

Called when either the source or destination rectangle changes.

virtual HRESULT OnUpdateRectangles( );

Return Values

Returns an HRESULT value.

Previous Home Topic Contents Index Next

CBaseControlVideo::OnVideoSizeChange

CBaseControlVideo Class

Passes an EC_VIDEO_SIZE_CHANGED message to the filter graph manager.

virtual HRESULT OnVideoSizeChange( );

Return Values

Returns an HRESULT value.

Remarks

A video renderer should call this member function each time the video size is changed; this will typically be called once after initial connection. If the renderer can support dynamic format changes (from 320x240 to 160x120), it should also call it after each change.

Previous Home Topic Contents Index Next

CBaseControlVideo::put_DestinationHeight

CBaseControlVideo Class

Sets the destination rectangle height.

HRESULT put_DestinationHeight(
long *DestinationHeight*
);

Parameters

DestinationHeight
New destination height.

Return Values

Returns an HRESULT value.

Remarks

An application can change the source and destination rectangles for the video through the IBasicVideo interface. The source rectangle affects which section of the native video source will appear on the display; the destination rectangle affects where the video will appear when played. The destination rectangle is relative to the client area of the window in which it is playing. The upper-left corner of the window is coordinate (0,0).

Previous Home Topic Contents Index Next

CBaseControlVideo::put_DestinationLeft

CBaseControlVideo Class

Sets the left coordinate of the destination rectangle.

HRESULT put_DestinationLeft(
long *DestinationLeft*
);

Parameters

DestinationLeft
New left coordinate of destination rectangle.

Return Values

Returns an HRESULT value.

Remarks

An application can change the source and destination rectangles for the video through the IBasicVideo interface. The source rectangle affects which section of the native video source will appear on the display; the destination rectangle affects where the video will appear when played. The destination rectangle is relative to the client area of the window in which it is playing. The upper-left corner of the window is coordinate (0,0).

Previous Home Topic Contents Index Next

CBaseControlVideo::put_DestinationTop

CBaseControlVideo Class

Sets the top coordinate of the destination rectangle.

HRESULT put_DestinationTop(
long *DestinationTop*
);

Parameters

DestinationTop
New top coordinate of the destination rectangle.

Return Values

Returns an HRESULT value.

Remarks

An application can change the source and destination rectangles for the video through the IBasicVideo interface. The source rectangle affects which section of the native video source will appear on the display; the destination rectangle affects where the video will appear when played. The destination rectangle is relative to the client area of the window in which it is playing. The upper-left corner of the window is coordinate (0,0).

Previous Home Topic Contents Index Next

CBaseControlVideo::put_DestinationWidth

CBaseControlVideo Class

Sets the width of the destination rectangle.

HRESULT put_DestinationWidth(
long *DestinationWidth*
);

Parameters

DestinationWidth
New destination width.

Return Values

Returns an HRESULT value.

Remarks

An application can change the source and destination rectangles for the video through the IBasicVideo interface. The source rectangle affects which section of the native video source will appear on the display; the destination rectangle affects where the video will appear when played. The destination rectangle is relative to the client area of the window in which it is playing. The upper-left corner of the window is coordinate (0,0).

Previous Home Topic Contents Index Next

CBaseControlVideo::put_SourceHeight

CBaseControlVideo Class

Sets the source rectangle height.

HRESULT put_SourceHeight(
long *SourceHeight*
);

Parameters

SourceHeight
Contains the source height.

Return Values

Returns an HRESULT value.

Remarks

An application can change the source and destination rectangles for the video through the IBasicVideo interface. The source rectangle affects which section of the native video source will appear on the display; the destination rectangle affects where the video will appear when played. The destination rectangle is relative to the client area of the window in which it is playing. The upper-left corner of the window is coordinate (0,0).

Previous Home Topic Contents Index Next

CBaseControlVideo::put_SourceLeft

CBaseControlVideo Class

Sets the source rectangle left coordinate.

HRESULT put_SourceLeft(
long *SourceLeft*
);

Parameters

SourceLeft
New left coordinate of the source rectangle.

Return Values

Returns an HRESULT value.

Remarks

An application can change the source and destination rectangles for the video through the IBasicVideo interface. The source rectangle affects which section of the native video source will appear on the display; the destination rectangle affects where the video will appear when played. The destination rectangle is relative to the client area of the window in which it is playing. The upper-left corner of the window is coordinate (0,0).

Previous Home Topic Contents Index Next

CBaseControlVideo::put_SourceTop

CBaseControlVideo Class

Sets the top coordinate of the source rectangle.

HRESULT put_SourceTop(
long *SourceTop*
);

Parameters

SourceTop
New top coordinate of the source rectangle.

Return Values

Returns an HRESULT value.

Remarks

An application can change the source and destination rectangles for the video through the IBasicVideo interface. The source rectangle affects which section of the native video source will appear on the display; the destination rectangle affects where the video will appear when played. The destination rectangle is relative to the client area of the window in which it is playing. The upper-left corner of the window is coordinate (0,0).

Previous Home Topic Contents Index Next

CBaseControlVideo::put_SourceWidth

CBaseControlVideo Class

Sets the width of the source rectangle.

```
HRESULT put_SourceWidth(
  long SourceWidth
  );
```

Parameters

SourceWidth
New width of the source rectangle.

Return Values

Returns an HRESULT value.

Remarks

An application can change the source and destination rectangles for the video through the IBasicVideo interface. The source rectangle affects which section of the native video source will appear on the display; the destination rectangle affects where the video will appear when played. The destination rectangle is relative to the client area of the window in which it is playing. The upper-left corner of the window is coordinate (0,0).

CBaseControlVideo::SetControlVideoPin

CBaseControlVideo Class

Sets the pin used by the filter.

void SetControlVideoPin(
CBasePin *pPin*
);

Parameters

pPin
Pin with which the interface is synchronized.

Return Values

No return value.

Remarks

The interface can be called only when the filter has been connected successfully. The object is passed through this method to the pin with which it is synchronized; in most cases it will determine if the pin is connected when it has an interface method called and will return VFW_E_NOT_CONNECTED if it fails.

CBaseControlVideo::SetDefaultDestinationPositio

CBaseControlVideo Class

Sets the renderer back to using the default destination position (typically the entire window client area).

HRESULT SetDefaultDestinationPosition( );

Return Values

Returns an HRESULT value.

Remarks

An application can change the source and destination rectangles for the video through the IBasicVideo interface. The source rectangle affects which section of the native video source will appear on the display; the destination rectangle affects where the video will appear when played. The destination rectangle is relative to the client area of the window in which it is playing. The upper-left corner of the window is coordinate (0,0).

Previous Home Topic Contents Index Next

CBaseControlVideo::SetDefaultSourcePosition

CBaseControlVideo Class

Sets the renderer back to using the default source position (typically all the native video).

HRESULT SetDefaultSourcePosition( );

Return Values

Returns an HRESULT value.

Remarks

An application can change the source and destination rectangles for the video through the IBasicVideo interface. The source rectangle affects which section of the native video source will appear on the display; the destination rectangle affects where the video will appear when played. The destination rectangle is relative to the client area of the window in which it is playing. The upper-left corner of the window is coordinate (0,0).

Previous Home Topic Contents Index Next

CBaseControlVideo::SetDefaultSourceRect

CBaseControlVideo Class

Sets the default source video rectangle (pure virtual). This in an internal member function that gets called when the source rectangle is reset.

virtual HRESULT SetDefaultSourceRect( ) PURE;

Return Values

Returns an HRESULT value.

Remarks

Derived classes should override this to reset the source rectangle. It is called from CBaseControlVideo::SetDefaultSourcePosition.

The following example from the video renderer sample, SampVid, demonstrates an implementation of this function in a derived class.

```
// This is called when we reset the default source rectangle

HRESULT CVideoText::SetDefaultSourceRect()
{
    VIDEOINFO *pVideoInfo = (VIDEOINFO *) m_pRenderer->m_mtIn.Format();
    BITMAPINFOHEADER *pHeader = HEADER(pVideoInfo);
    RECT SourceRect = {0,0,pHeader->biWidth,pHeader->biHeight};
    m_pRenderer->m_DrawImage.SetSourceRect(&SourceRect);
    return NOERROR;
}
```

In this example, CVideoText is a class derived from CBaseControlVideo, m_pRenderer holds an object of a class derived from CBaseVideoRenderer, and the m_DrawImage data member, defined in the derived class, holds a CDrawImage object. The m_mtIn data member, also defined in the derived class, holds a CMediaType object with media type of the input pin.

CBaseControlVideo::SetDefaultTargetRect

CBaseControlVideo Class

Sets the default target video rectangle (pure virtual). This is an internal member function that gets called when the source rectangle is reset.

virtual HRESULT SetDefaultTargetRect( ) PURE;

Return Values

Returns an HRESULT value.

Remarks

Derived classes should override this to reset the destination video rectangle. It is called from the CBaseControlVideo::SetDefaultDestinationPosition member function.

The following example from the video renderer sample, SampVid, demonstrates an implementation of this function in a derived class.

```
// This is called when we reset the default target rectangle

HRESULT CVideoText::SetDefaultTargetRect()
{
    VIDEOINFO *pVideoInfo = (VIDEOINFO *) m_pRenderer->m_mtIn.Format();
    BITMAPINFOHEADER *pHeader = HEADER(pVideoInfo);
    RECT TargetRect = {0,0,m_Size.cx,m_Size.cy};
    m_pRenderer->m_DrawImage.SetTargetRect(&TargetRect);
    return NOERROR;
}
```

In this example, CVideoText is a class derived from CBaseControlVideo, m_pRenderer holds an object of a class derived from CBaseVideoRenderer, and the m_DrawImage data member, defined in the derived class, holds a CDrawImage object. The m_mtIn data member, also defined in the derived class, holds a CMediaType object with the media type of the input pin.

Previous Home Topic Contents Index Next

CBaseControlVideo::SetDestinationPosition

CBaseControlVideo Class

Sets the destination rectangle for the video.

HRESULT SetDestinationPosition(
long *Left*,
long *Top*,
long *Width*,
long *Height*
);

Parameters

Left
New left coordinate.
Top

New top coordinate.

Width

New width.

Height

New height.

Return Values

Returns an HRESULT value.

Remarks

An application can change the source and destination rectangles for the video through the IBasicVideo interface. The source rectangle affects which section of the native video source will appear on the display; the destination rectangle affects where the video will appear when played. The destination rectangle is relative to the client area of the window in which it is playing. The upper-left corner of the window is coordinate (0,0).

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseControlVideo::SetSourcePosition

CBaseControlVideo Class

Sets a new source position for the video.

```
HRESULT SetSourcePosition(
  long Left,
  long Top,
  long Width,
  long Height
  );
```

Parameters

Left

New left coordinate.

Top

New top coordinate.

Width

New width.

Height

New height.

Return Values

Returns an HRESULT value.

Remarks

An application can change the source and destination rectangles for the video through the IBasicVideo interface. The source rectangle affects which section of the native video source will appear on the display; the destination rectangle affects where the video will appear when played. The destination rectangle is relative to the client area of the window in which it is playing. The upper-left corner of the window is coordinate (0,0).

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseControlVideo::SetSourceRect

CBaseControlVideo Class

Sets the current source video rectangle (pure virtual). This is an internal member function that gets called when the source rectangle changes.

```
virtual HRESULT SetSourceRect(
  RECT *pSourceRect
  ) PURE;
```

Parameters

pSourceRect
Contains the source rectangle.

Return Values

Returns an HRESULT value.

Remarks

Derived classes should override this member function to know when the source rectangle changes. It is called from the following member functions.

- CBaseControlVideo::SetSourcePosition
- CBaseControlVideo::put_SourceLeft
- CBaseControlVideo::put_SourceWidth

- CBaseControlVideo::put_SourceTop
- CBaseControlVideo::put_SourceHeight

The following example from the video renderer sample, SampVid, demonstrates an implementation of this function in a derived class.

```
HRESULT CVideoText::SetSourceRect(RECT *pSourceRect)
{
    m_pRenderer->m_DrawImage.SetSourceRect(pSourceRect);
    return NOERROR;
}
```

In this example, CVideoText is a class derived from CBaseControlVideo, m_pRenderer holds an object of a class derived from CBaseVideoRenderer, and the m_DrawImage data member, defined in the derived class, holds a CDrawImage object.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseControlVideo::SetTargetRect

CBaseControlVideo Class

Sets the current target rectangle (pure virtual). This is an internal member function that gets called when the destination rectangle changes.

virtual HRESULT SetTargetRect(
RECT **pTargetRect*
) PURE;

Parameters

pTargetRect
Contains the destination rectangle.

Return Values

Returns an HRESULT value.

Remarks

Derived classes should override this to know when the destination rectangle changes. It is called from the following member functions.

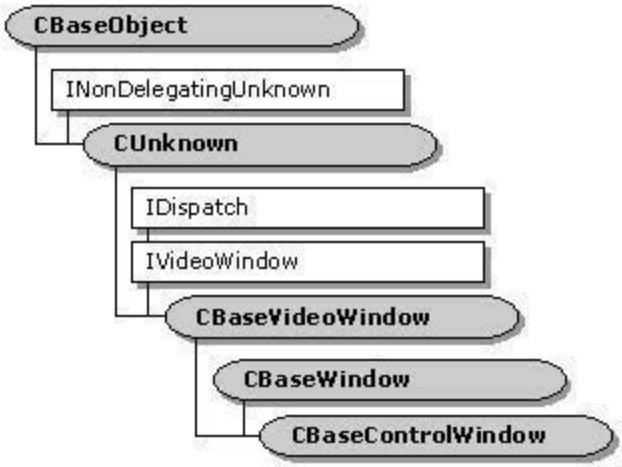
- CBaseControlVideo::SetDestinationPosition
- CBaseControlVideo::put_DestinationLeft
- CBaseControlVideo::put_DestinationWidth
- CBaseControlVideo::put_DestinationTop
- CBaseControlVideo::put_DestinationHeight

The following example from the video renderer sample, SampVid, demonstrates an implementation of this function in a derived class.

```
HRESULT CVideoText::SetTargetRect(RECT *pTargetRect)
{
    m_pRenderer->m_DrawImage.SetTargetRect(pTargetRect);
    return NOERROR;
}
```

In this example, CVideoText is a class derived from CBaseControlVideo, m_pRenderer holds an object of a class derived from CBaseVideoRenderer, and the m_DrawImage data member, defined in the derived class, holds a CDrawImage object.

CBaseControlWindow Class



The **CBaseControlWindow** class implements the IVideoWindow interface and controls external access to its associated filter. You must synchronize the **CBaseControlWindow** object with the filter by passing it a pointer to a critical section synchronization object. For more information about critical section synchronization objects, see the Platform SDK. The **CBaseControlWindow** class provides a number of methods that return property settings without dealing with this critical section. For example, calling CBaseControlWindow::get_AutoShow to retrieve the value of the m_bAutoShow data member locks the critical section. The filter might already have a locked internal critical section, however, which could violate the filter's lock hierarchy. Instead, calling the CBaseControlWindow::IsAutoShowEnabled member function returns the required value without affecting the critical section.

All **CBaseControlWindow** implemented IVideoWindow methods require that the filter be connected correctly with its upstream filter. For this reason, class objects require a synchronization pin, which you set by calling the CBaseControlWindow::SetControlWindowPin method. Whenever you call an **IVideoWindow** method, the **CBaseControlWindow** object checks that the pin is still connected.

Protected Data Members

Name	Description
m_bAutoShow	Result when the state changes.
m_bCursorHidden	Determination of whether the cursor is displayed or hidden.
m_BorderColour	Color of the current window border.
m_hwndDrain	Window handle to which messages received are posted.
m_hwndOwner	Owning window.
m_pFilter	Pointer to the owning media filter.
m_pInterfaceLock	Externally defined critical section.
m_pPin	Control of the media types for connection.

Member Functions

Name	Description
CBaseControlWindow	Constructs a CBaseControlWindow object.
DoGetWindowStyle	Retrieves either the typical or extended window styles.
DoSetWindowStyle	Sets the typical or extended window styles.
GetBorderColour	Retrieves the current border color. This is a helper member function.
GetOwnerWindow	Retrieves the owning window. This is a helper member function.
IsAutoShowEnabled	Retrieves information about whether the video window automatically appears when the rendering filter pauses or runs.
IsCursorHidden	Retrieves the current state of the m_bCursorHidden data member without locking the critical section. This is a helper member function.
PossiblyEatMessage	Distributes messages to the parent window.
SetControlWindowPin	Notifies the object of the pin to which it applies.

Implemented IVideoWindow Methods

Name	Description
get_AutoShow	Retrieves the current AutoShow flag setting.
get_BackgroundPalette	Retrieves the realized palette in the background flag.
get_BorderColor	Retrieves the current border color.
get_Caption	Retrieves the current window caption.
get_FullScreenMode	Retrieves the current full-screen mode.
get_Height	Retrieves the current window height.
get_Left	Retrieves the current left window coordinate.
GetMaxIdealImageSize	Retrieves the maximum size of the ideal image.
get_MessageDrain	Returns the current message drain.
GetMinIdealImageSize	Retrieves the minimum size of the ideal image.
get_Owner	Retrieves the Microsoft® Win32® parent window handle.
GetRestorePosition	Retrieves the position to which the window will be restored when maximized or minimized.
get_Top	Retrieves the y-coordinate for the top of the window.
get_Visible	Retrieves the current visibility setting of the window.
get_Width	Retrieves the width of the window.
GetWindowPosition	Retrieves the current window coordinates.
get_WindowState	Retrieves the current state of the window.
get_WindowStyle	Retrieves the standard window styles.
get_WindowStyleEx	Retrieves the extended window styles.
HideCursor	Hides or displays the cursor.
IsCursorHidden	Retrieves the current state of the m_bCursorHidden data member.
NotifyOwnerMessage	Passes on messages that are sent to owning windows.
put_AutoShow	Sets the AutoShow property.
put_BackgroundPalette	Sets a flag to realize the palette in the background.
put_BorderColor	Sets the current border color.
put_Caption	Sets the current window caption.
put_FullScreenMode	Sets the full-screen mode.
put_Height	Sets the current window height.

put_Left	Sets the left coordinate for the window.
put_MessageDrain	Sets the message drain window.
put_Owner	Sets the Microsoft Win32 parent window handle.
put_Top	Sets the position for the top of the window.
put_Visible	Hides or shows the window.
put_Width	Sets the width of the window.
put_WindowState	Sets the state of the window.
put_WindowStyle	Sets the standard window styles.
put_WindowStyleEx	Sets the extended window styles.
SetWindowForeground	Sets the window in the foreground.
SetWindowPosition	Sets the window position.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseControlWindow::CBaseControlWindow

CBaseControlWindow Class

Constructs a CBaseControlWindow object.

```
CBaseControlWindow(
  CBaseMediaFilter *pFilter,
  CCritSec *pInterfaceLock,
  TCHAR *pName,
  LPUNKNOWN pUnk,
  HRESULT *phr
  );
```

Parameters

pFilter
Owning media filter object.
pInterfaceLock
Critical section to use for locking.
pName
Object description.
pUnk
Typical Component Object Model (COM) ownership.
phr
COM return value.

Return Values

No return value.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseControlWindow::DoGetWindowStyle

CBaseControlWindow Class

Retrieves the current normal or extended window styles.

HRESULT DoGetWindowStyle(
long **pStyle*,
long *WindowLong*
);

Parameters

pStyle
Contains the appropriate styles.
WindowLong
Either GWL_STYLE or GWL_EXSTYLE.

Return Values

Returns an HRESULT value.

Remarks

This member function calls the Win32 GetWindowLong function to retrieve the window style. It is called by the CBaseControlWindow::get_WindowStyle and CBaseControlWindow::get_WindowStyleEx member functions.

Previous Home Topic Contents Index Next

CBaseControlWindow::DoSetWindowStyle

CBaseControlWindow Class

Changes the typical or extended window styles.

HRESULT DoSetWindowStyle(
long *Style*,
long *WindowLong*
);

Parameters

Style
Contains the appropriate window styles.
WindowLong
Either GWL_STYLE or GWL_EXSTYLE.

Return Values

Returns an HRESULT value.

Remarks

This member function calls the Win32 SetWindowLong function to set the window style, and then redisplays the window in the current position. This member function is called by the CBaseControlWindow::put_WindowStyle and CBaseControlWindow::put_WindowStyleEx member functions.

Previous Home Topic Contents Index Next

CBaseControlWindow::get_AutoShow

CBaseControlWindow Class

Retrieves the current AutoShow state flag.

HRESULT get_AutoShow(
long **AutoShow*
);

Parameters

AutoShow
Automation Boolean flag (0 is off, –1 is on).

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IVideoWindow::get_AutoShow method. This property simplifies window display access for applications. If this is set to –1 (on), the window, which is typically hidden after connection of the filter, will be displayed automatically when the filter pauses or runs. The window should not be hidden when the filter stops, however. If this parameter is set to 0 (off), the window is made visible only when the application calls CBaseControlWindow::put_Visible or CBaseControlWindow::put_WindowState with the appropriate parameters.

This member function is meant to be called by external objects through the IVideoWindow interface, and therefore locks the critical section to synchronize with the associated filter. Call the CBaseControlWindow::IsAutoShowEnabled member function to retrieve this property if you are not calling from an external object.

Previous Home Topic Contents Index Next

CBaseControlWindow::get_BackgroundPalette

CBaseControlWindow Class

Retrieves the realized palette in the background flag.

HRESULT get_BackgroundPalette(
long *pBackgroundPalette*
);

Parameters

pBackgroundPalette
Automation Boolean flag (0 is off, –1 is on).

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IVideoWindow::get_BackgroundPalette method. If a video will be played within another application or document, the application might want to use its own palette. It can ask that the video use the current foreground palette rather than its own by setting this flag to –1. If this is set to 0, the window will install and realize its own preferred palette. Note that asking the window to use a different palette will cause severe

performance penalties.

Previous Home Topic Contents Index Next

CBaseControlWindow::get_BorderColor

CBaseControlWindow Class

Retrieves the current border color.

HRESULT get_BorderColor(
long **Color*
);

Parameters

Color
Current border color.

Return Values

Returns an HRESULT value.

Remarks

An application can set a destination rectangle in which the video should be displayed. This rectangle is relative to the client area for the window. If this is done (the default is to always paint the entire window), there is a border surrounding the video. This property affects the color used by the border. Although the parameter is specified as a LONG type, it is actually a COLORREF value.

This member function is meant to be called by external objects through the IVideoWindow interface, and therefore locks the critical section to synchronize with the associated filter. Call the CBaseControlWindow::GetBorderColour member function to retrieve this property if not calling from an external object.

Previous Home Topic Contents Index Next

CBaseControlWindow::GetBorderColour

CBaseControlWindow Class

Returns the current window border color, m_BorderColour.

COLORREF GetBorderColour( );

Return Values

Returns the color of the border.

Remarks

An application can set a destination rectangle to display the video. This rectangle should be relative to the client area for the window. If this is done (the default is to always paint the entire window), there is an area that surrounds the video; that is, the border. The border color can be set through the CBaseControlWindow::put_BorderColor member function. This property affects the color of the border. Use this member function instead of CBaseControlWindow::get_BorderColor, unless you are calling this externally through the IVideoWindow::get_BorderColor method.

Previous Home Topic Contents Index Next

CBaseControlWindow::get_Caption

CBaseControlWindow Class

Retrieves the current window caption.

HRESULT get_Caption(
BSTR *pstrCaption*
);

Parameters

pstrCaption
Current window caption.

Return Values

Returns an HRESULT value.

Remarks

Most top-level windows on a Windows-based desktop have a title (caption) associated with them. This property can be queried and set through the IVideoWindow interface. Any caption set will be visible only if the window has the WS_CAPTION style applied. If it does not, the

caption can still be set (and retrieved), although it will not be visible to the user.

Previous Home Topic Contents Index Next

CBaseControlWindow::get_FullScreenMode

CBaseControlWindow Class

Retrieves the current full-screen mode.

HRESULT get_FullScreenMode(
long **FullScreenMode*
);

Parameters

FullScreenMode
Current full-screen mode.

Return Values

Returns an HRESULT value.

Remarks

This member function returns E_NOTIMPL by default. This informs the IVideoWindow plug-in distributor that this renderer does not implement a full-screen renderer.

Previous Home Topic Contents Index Next

CBaseControlWindow::get_Height

CBaseControlWindow Class

Retrieves the current window height.

HRESULT get_Height(
long **pHeight*
);

Parameters

pHeight
Current window height, in pixels.

Return Values

Returns an HRESULT value.

Remarks

The window has a position on the desktop. This is expressed in pixels by four coordinates (left, top, right, and bottom). Interfaces that are automated by OLE typically express this position through left, top, width, and height; this is the convention used in DirectShow™. All coordinates are expressed in pixels, and changing any coordinate will update the window immediately.

Setting the left or top coordinates moves the window left or up, respectively; these coordinates have no effect on the width and height of the window. Likewise, setting the width and height does not affect the left and top coordinates.

Previous Home Topic Contents Index Next

CBaseControlWindow::get_Left

CBaseControlWindow Class

Retrieves the current left window coordinate.

```
HRESULT get_Left(
  long *pLeft
  );
```

Parameters

pLeft
Contains the left coordinate, in pixels.

Return Values

Returns an HRESULT value.

Remarks

The window has a position on the desktop. This position is expressed in pixels by four

coordinates (left, top, right, and bottom). Interfaces that are automated by OLE typically express this position through left, top, width, and height; this is the convention used in DirectShow. All coordinates are expressed in pixels, and changing any coordinate will update the window immediately.

Setting the left or top coordinates moves the window left and up, respectively; these coordinates have no effect on the width and height of the window. Likewise, setting the width and height have no effect on the left and top coordinates.

Previous Home Topic Contents Index Next

CBaseControlWindow::GetMaxIdealImageSize

CBaseControlWindow Class

Retrieves the maximum ideal image size.

HRESULT GetMaxIdealImageSize(
long **pWidth*,
long **pHeight*
);

Parameters

pWidth
Maximum ideal width, in pixels.
pHeight
Maximum ideal height, in pixels.

Return Values

Returns an HRESULT value.

Remarks

Various renderers have performance restrictions on the size of images they can display. Although they should still function properly when requested to display images larger than the specified maximum, renderers can nominate the minimum and maximum ideal sizes through the IVideoWindow interface. This interface can be called only when the filter graph is paused or running, because it is not until then that resources are allocated and the renderer can recognize its restrictions. If no restrictions exist, the renderer fills in the *pWidth* and *pHeight* parameters with the native video dimensions and returns S_FALSE. If restrictions do exist, the restricted width and height are entered, and the member function returns S_OK.

The dimensions apply to the size of the destination video and not to the overall window size. So, when calculating the size of the window to set, account for the current window styles (for example, WS_CAPTION and WS_BORDER).

Previous Home Topic Contents Index Next

CBaseControlWindow::get_MessageDrain

CBaseControlWindow Class

Returns the current message drain.

HRESULT get_MessageDrain(
OAHWND **Drain*
);

Parameters

Drain
Current window receiving window messages.

Return Values

Returns an HRESULT value.

Remarks

Messages sent to the video renderer filter can be posted to another window. The window registered to receive these messages (using the **CBaseControlWindow::get_MessageDrain** member function) is the current message drain.

Previous Home Topic Contents Index Next

CBaseControlWindow::GetMinIdealImageSize

CBaseControlWindow Class

Retrieves the minimum ideal image size.

HRESULT GetMinIdealImageSize(
long **pWidth*,
long **pHeight*

);

Parameters

pWidth
Minimum ideal width, in pixels.
pHeight
Minimum ideal height, in pixels.

Return Values

Returns an HRESULT value.

Remarks

Various renderers have performance restrictions on the size of images they can display. Although they should still function properly when requested to display images larger than the specified maximum, renderers can nominate the minimum and maximum ideal sizes through the IVideoWindow interface. This interface can be called only when the filter graph is paused or running, because it is not until then that resources are allocated and the renderer can recognize its restrictions. If no restrictions exist, the renderer fills in the *pWidth* and *pHeight* parameters with the native video dimensions and returns S_FALSE. If restrictions do exist, the restricted width and height are entered, and the member function returns S_OK.

The dimensions apply to the size of the destination video and not to the overall window size. So, when calculating the size of the window to set, account for the current window styles (for example, WS_CAPTION and WS_BORDER).

Previous Home Topic Contents Index Next

CBaseControlWindow::get_Owner

CBaseControlWindow Class

Retrieves the current window owner.

HRESULT get_Owner(
OAHWND **Owner*
);

Parameters

Owner
Contains the window owner.

Return Values

Returns an HRESULT value.

Remarks

The video window can play back within a document environment. To do this, the window must be made a child of another window (so that it is clipped and moved appropriately). This property allows the owner of the window to be set and retrieved. When the window is owned by another window, it simply calls the Microsoft Win32 SetParent function. An application calling this function will change the window styles to set the WS_CHILD bit on.

When the window is owned by another window, it will automatically forward certain sets of messages (in particular, mouse and keyboard messages). This allows an application to do simple hot-spot editing and other interactions.

This member function is meant to be called by external objects through the IVideoWindow interface, and therefore locks the critical section to synchronize with the associated filter. Call the CBaseControlWindow::GetOwnerWindow member function to retrieve this property if not calling from an external object.

Previous Home Topic Contents Index Next

CBaseControlWindow::GetOwnerWindow

CBaseControlWindow Class

Returns the owning window handle, m_hwndOwner.

HWND GetOwnerWindow( );

Return Values

Returns an internal method to return the owner window.

Remarks

Retrieves the owning window without calling the interface method. Use this member function instead of CBaseControlWindow::get_Owner, unless you are calling this externally through the IVideoWindow::get_Owner method.

Previous Home Topic Contents Index Next

CBaseControlWindow::GetRestorePosition

CBaseControlWindow Class

Retrieves the position to which the window will be restored when it is not maximized or minimized.

HRESULT GetRestorePosition(
long **pLeft*,
long **pTop*,
long **pWidth*,
long **pHeight*
);

Parameters

pLeft
Value for leftmost coordinate.
pTop
Value for top of the window.
pWidth
Value for width of the window.
pHeight
Value for height of window.

Return Values

Returns an HRESULT value.

Remarks

This is the same as the values returned by the CBaseControlWindow::GetWindowPosition function when the window is neither maximized nor minimized.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseControlWindow::get_Top

CBaseControlWindow Class

Retrieves the top window coordinate.

HRESULT get_Top(
long *pTop*
);

Parameters

pTop
Contains the top coordinate, in pixels.

Return Values

Returns an HRESULT value.

Remarks

The window has a position on the desktop. This is expressed in pixels by four coordinates (left, top, right, and bottom). Interfaces that are automated by OLE typically express this position through left, top, width, and height; this is the convention used in DirectShow. All coordinates are expressed in pixels, and changing any coordinate will update the window immediately.

Setting the left or top coordinates moves the window left or up, respectively; these coordinates have no effect on the width and height of the window. Likewise, setting the width and height does not affect the left and top coordinates.

Previous Home Topic Contents Index Next

CBaseControlWindow::get_Visible

CBaseControlWindow Class

Retrieves the current window visibility.

HRESULT get_Visible(
long *pVisible*
);

Parameters

pVisible
Automation Boolean flag (0 is off, –1 is on).

Return Values

Returns an HRESULT value.

Remarks

This member function returns –1 if the window has the WS_VISIBLE style; 0 otherwise.

Previous Home Topic Contents Index Next

CBaseControlWindow::get_Width

CBaseControlWindow Class

Retrieves the current window width.

```
HRESULT get_Width(
  long *pWidth
  );
```

Parameters

pWidth
Contains the window width, in pixels.

Return Values

Returns an HRESULT value.

Remarks

The window has a position on the desktop. This is expressed in pixels by four coordinates (left, top, right, and bottom). Interfaces that are automated by OLE typically express this position through left, top, width, and height; this is the convention used in DirectShow. All coordinates are expressed in pixels, and changing any coordinate will update the window immediately.

Setting the left or top coordinates moves the window left or up, respectively; these coordinates have no effect on the width and height of the window. Likewise, setting the width and height does not affect the left and top coordinates.

Previous Home Topic Contents Index Next

CBaseControlWindow::GetWindowPosition

CBaseControlWindow Class

Retrieves the current coordinates for the window.

HRESULT GetWindowPosition(
long **pLeft*,
long **pTop*,
long **pWidth*,
long **pHeight*
);

Parameters

pLeft
Contains the left coordinate, in screen coordinates.
pTop
Contains the top coordinate, in screen coordinates.
pWidth
Contains the window width, in screen coordinates.
pHeight
Contains the window height, in screen coordinates.

Return Values

Returns an HRESULT value.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseControlWindow::get_WindowState

CBaseControlWindow Class

Retrieves the current window state.

HRESULT get_WindowState(
long **pWindowState*
);

Parameters

pWindowState
Contains the window state.

Return Values

Returns an HRESULT value.

Remarks

This member function returns a subset of the parameters of the Microsoft Win32 ShowWindow function. In particular, it returns SW_SHOW and SW_HIDE, depending on the current visibility of the window. It also returns SW_MINIMIZE and SW_MAXIMIZE, depending on whether the window is an icon or is expanded.

Previous Home Topic Contents Index Next

CBaseControlWindow::get_WindowStyle

CBaseControlWindow Class

Retrieves the standard window styles.

HRESULT get_WindowStyle(
long *pWindowStyle*
);

Parameters

pWindowStyle
Contains the window styles.

Return Values

Returns an HRESULT value.

Remarks

This member function returns the standard window styles, such as WS_CHILD and WS_VISIBLE. It calls the CBaseControlWindow::DoGetWindowStyle member function.

Previous Home Topic Contents Index Next

CBaseControlWindow::get_WindowStyleEx

CBaseControlWindow Class

Retrieves the extended window styles.

HRESULT get_WindowStyleEx(
long **pWindowStyleEx*
);

Parameters

pWindowStyleEx
Contains the extended window styles.

Return Values

Returns an HRESULT value.

Remarks

This member function retrieves the extended window styles. It calls the CBaseControlWindow::DoGetWindowStyle member function.

Previous Home Topic Contents Index Next

CBaseControlWindow::HideCursor

CBaseControlWindow Class

Hides or displays the cursor.

HRESULT HideCursor(
long *HideCursor*
);

Parameters

HideCursor
Set to OATRUE to hide the cursor, or OAFALSE to display the cursor.

Return Values

Returns an HRESULT value.

CBaseControlWindow::IsCursorHidden

CBaseControlWindow Class

Retrieves the current state of the m_bCursorHidden data member.

HRESULT IsCursorHidden(
long **CursorHidden*
);

BOOL IsCursorHidden( );

Parameters

CursorHidden
Value of m_bCursorHidden.

Return Values

When called without a parameter, returns OATRUE if the cursor is hidden, or OAFALSE if the cursor is visible.

When called with a parameter, returns an HRESULT value.

Remarks

Internal objects should call this member function without the *CursorHidden* parameter to avoid locking the critical section. External objects access this member function with the *CursorHidden* parameter through the IVideoWindow::IsCursorHidden method.

CBaseControlWindow::IsAutoShowEnabled

CBaseControlWindow Class

Retrieves information about whether the video window automatically appears when the rendering filter pauses or runs.

BOOL IsAutoShowEnabled( );

Return Values

Returns TRUE if the m_bAutoShow member is set to –1 or FALSE if it is set to 0.

Remarks

If the m_bAutoShow member is set to –1 on a video window that is hidden, the window becomes visible when the filter pauses or runs. If this member is set to 0, the window will appear only if you use the CBaseControlWindow::put_Visible or CBaseControlWindow::put_WindowState member function with the appropriate parameters.

This member function retrieves the m_bAutoShow member setting and has the same result as using the IVideoWindow::get_AutoShow method.

Previous Home Topic Contents Index Next

CBaseControlWindow::NotifyOwnerMessage

CBaseControlWindow Class

Passes along specific messages to the video window.

HRESULT NotifyOwnerMessage(
long *hwnd*,
long *uMsg*,
long *wParam*,
long *lParam*
);

Parameters

hwnd
Handle to the video window.
uMsg
Message details.
wParam
Standard WPARAM parameter.
lParam
Standard LPARAM parameter.

Return Values

Returns NO_ERROR.

Remarks

When the video window is a child of another window, it does not receive certain top-level window messages. These messages can be valuable to a renderer, because they could affect its behavior. **NotifyOwnerMessage** passes any of the following messages to the video window.

WM_ACTIVATEAPP
WM_DEVMODECHANGE
WM_DISPLAYCHANGE
WM_PALETTECHANGED
WM_PALETTEISCHANGING
WM_QUERYNEWPALETTE
WM_SYSCOLORCHANGE

You can request that the IVideoWindow plug-in distributor (PID) make a window become a child of another window. When this occurs, the PID will look for certain messages that might be sent to the owning window. The PID will then forward those messages to the owned window. The default processing for the messages is to send them to the owned window procedure synchronously by calling the Win32 SendMessage function.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseControlWindow::PossiblyEatMessage

CBaseControlWindow Class

Forwards keyboard and mouse messages to a specified window.

BOOL WINAPI PossiblyEatMessage(
HWND *hwndDrain*,
UINT *uMsg*,
WPARAM *wParam*,
LPARAM *lParam*
)

Parameters

hwndDrain

Handle of the window to which messages will be forwarded.

uMsg
Message that was forwarded.

wParam
First message parameter.

lParam
Second message parameter.

Return Values

Returns TRUE if the message was posted or FALSE if it wasn't.

Remarks

When the window is owned, it will pass certain classes of messages to the owning window (such as keyboard and mouse events). In this case, the Win32 PostMessage function is used to post messages to any window specified by *hwndDrain* which is set in CBaseControlWindow::put_MessageDrain. If a certain message cannot be posted, this message will return FALSE.

The following is a list of messages that will get passed on untranslated and return TRUE.

WM_CHAR	WM_DEADCHAR
WM_KEYDOWN	WM_KEYUP
WM_LBUTTONDBLCLK	WM_LBUTTONDOWN
WM_LBUTTONUP	WM_MBUTTONDBLCLK
WM_MBUTTONDOWN	WM_MBUTTONUP
WM_MOUSEACTIVATE	WM_MOUSEMOVE
WM_NCHITTEST	WM_NCLBUTTONDBLCLK
WM_NCLBUTTONDOWN	WM_NCLBUTTONUP
WM_NCMBUTTONDBLCLK	WM_NCMBUTTONDOWN
WM_NCMBUTTONUP	WM_NCMOUSEMOVE
WM_NCRBUTTONDBLCLK	WM_NCRBUTTONDOWN
WM_NCRBUTTONUP	WM_RBUTTONDBLCLK
WM_RBUTTONDOWN	WM_RBUTTONUP
WM_SYSCHAR	WM_SYSDEADCHAR
WM_SYSKEYDOWN	WM_SYSKEYUP

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseControlWindow::put_AutoShow

CBaseControlWindow Class

Sets the AutoShow state flag.

HRESULT put_AutoShow(
long *AutoShow*
);

Parameters

AutoShow
Automation Boolean flag (0 is off, –1 is on).

Return Values

Returns an HRESULT value.

Remarks

This property simplifies window display access for applications. If this is set to –1 (on), the window, which is typically hidden after the filter is connected, will be displayed automatically when the filter pauses or runs. The window should not be hidden when the filter stops, however. If this is set to 0 (off), the window is made visible only when the application calls CBaseControlWindow::put_Visible or CBaseControlWindow::put_WindowState with the appropriate parameters.

Previous Home Topic Contents Index Next

CBaseControlWindow::put_BackgroundPalette

CBaseControlWindow Class

Sets a flag to realize the palette in the background.

HRESULT put_BackgroundPalette(
long *BackgroundPalette*
);

Parameters

BackgroundPalette
Automation Boolean flag (0 is off, –1 is on).

Return Values

Returns an HRESULT value.

Remarks

To play a video within another application or document, the application might want to use its own palette. It can ask that the video use the current foreground palette rather than its own as the background palette by setting this flag to –1. If this is set to 0, the window will install and realize its own preferred palette. Asking the window to use a different palette will cause severe performance penalties.

Previous Home Topic Contents Index Next

CBaseControlWindow::put_BorderColor

CBaseControlWindow Class

Changes the border color.

HRESULT put_BorderColor(
long *Color*
);

Parameters

Color
Contains the new border color.

Return Values

Returns an HRESULT value.

Remarks

An application can establish a destination rectangle in which the video should be displayed. This rectangle is relative to the client area for the window. If this is done (the default is to always paint the entire window), there is a border surrounding the video. This property affects the color used by the border. Although the parameter is specified as a LONG type, it is actually a COLORREF value.

Previous Home Topic Contents Index Next

CBaseControlWindow::put_Caption

CBaseControlWindow Class

Sets the window title or caption.

HRESULT put_Caption(
BSTR *strCaption*
);

Parameters

strCaption
Contains the new window caption.

Return Values

Returns an HRESULT value.

Remarks

Most top-level windows on a Windows-based desktop have a title (caption) associated with them. This property can be queried and set through the IVideoWindow interface. Any caption set will be visible only if the window has the WS_CAPTION style applied. If it does not, the caption can still be set (and retrieved), although it will not be visible to the user.

Previous Home Topic Contents Index Next

CBaseControlWindow::put_FullScreenMode

CBaseControlWindow Class

Sets the full-screen mode of the renderer.

HRESULT put_FullScreenMode(
long *FullScreenMode*
);

Parameters

FullScreenMode
Full-screen mode to apply.

Return Values

Returns an HRESULT value.

Remarks

The current implementation returns E_NOTIMPL. A video renderer that implements a full-screen mode should override this member function and implement whatever modes it supports.

Previous Home Topic Contents Index Next

CBaseControlWindow::put_Height

CBaseControlWindow Class

Sets the window height.

HRESULT put_Height(
long *Height*
);

Parameters

Height
New window height, in pixels.

Return Values

Returns an HRESULT value.

Remarks

The window has a position on the desktop. This is expressed in pixels by four coordinates (left, top, right, and bottom). Interfaces that are automated by OLE typically express this position through left, top, width, and height; this is the convention used in DirectShow. All coordinates are expressed in pixels, and changing any coordinate will update the window immediately.

Setting the left or top coordinates moves the window left or up, respectively; these coordinates have no effect on the width and height of the window. Likewise, setting the width and height does not affect the left and top coordinates.

Previous Home Topic Contents Index Next

CBaseControlWindow::put_Left

CBaseControlWindow Class

Sets the left coordinate for the window.

HRESULT put_Left(
long *Left*
);

Parameters

Left
New left coordinate, in pixels.

Return Values

Returns an HRESULT value.

Remarks

The window has a position on the desktop. This is expressed in pixels by four coordinates (left, top, right, and bottom). Interfaces that are automated by OLE typically express this position through left, top, width, and height; this is the convention used in DirectShow. All coordinates are expressed in pixels, and changing any coordinate will update the window immediately.

Setting the left or top coordinates moves the window left or up, respectively; these coordinates have no effect on the width and height of the window. Likewise, setting the width and height does not affect the left and top coordinates.

Previous Home Topic Contents Index Next

CBaseControlWindow::put_MessageDrain

CBaseControlWindow Class

Sets the window to receive window messages sent to the video renderer.

HRESULT put_MessageDrain(
OAHWND *Drain*
);

Parameters

Drain
Window to post messages to.

Return Values

Returns an HRESULT value.

Remarks

Messages sent to the video renderer filter can be posted to another window. This member function registers the window to receive these messages. Unlike the CBaseControlWindow::put_Owner member function, this member function does not make the video window a child of another window. It is particularly useful for full-screen video renderers, which cannot be child windows.

Previous Home Topic Contents Index Next

CBaseControlWindow::put_Owner

CBaseControlWindow Class

Sets the video window's parent window; the parent window then forwards certain messages to the video window.

HRESULT put_Owner(
OAHWND *Owner*
);

Parameters

Owner
Handle to the parent window.

Return Values

Returns NOERROR.

Remarks

Internally, this method calls the Microsoft Win32 SetParent function to set the new owner and sets the parent window's style to WS_CHILD. The parent window will then forward certain sets of messages (in particular, mouse and keyboard messages) to the video window.

After you set the video window's owner, you must set the owner to NULL and the owner's window style to WS_OVERLAPPED and WS_CLIPCHILDREN before releasing the filter graph.

When you set the owner to NULL, this method turns off the parent window's WS_CHILD bit. If you don't set the owner to NULL, the parent window will continue to pass messages to the video window and errors will likely occur when the application closes.

Previous Home Topic Contents Index Next

CBaseControlWindow::put_Top

CBaseControlWindow Class

Sets the top window coordinate.

HRESULT put_Top(
long *Top*
);

Parameters

Top
New top coordinate, in pixels.

Return Values

Returns an HRESULT value.

Remarks

The window has a position on the desktop. This is expressed in pixels by four coordinates (left, top, right, and bottom). Interfaces that are automated by OLE typically express this position through left, top, width, and height; this is the convention used in DirectShow. All coordinates are expressed in pixels, and changing any coordinate will update the window immediately.

Setting the left or top coordinates moves the window left or up, respectively; these coordinates have no effect on the width and height of the window. Likewise, setting the width and height does not affect the left and top coordinates.

Previous Home Topic Contents Index Next

CBaseControlWindow::put_Visible

CBaseControlWindow Class

Makes the window either visible or hidden.

HRESULT put_Visible(
long *Visible*
);

Parameters

Visible
Automation Boolean flag (0 means window is hidden, –1 means window is shown).

Return Values

Returns an HRESULT value.

Previous Home Topic Contents Index Next

CBaseControlWindow::put_Width

CBaseControlWindow Class

Sets the window width.

HRESULT put_Width(
long *Width*
);

Parameters

Width
New window width, in pixels.

Return Values

Returns an HRESULT value.

Remarks

The window has a position on the desktop. This is expressed in pixels by four coordinates (left, top, right, and bottom). Interfaces that are automated by OLE typically express this position through left, top, width, and height; this is the convention used in DirectShow. All coordinates are expressed in pixels and changing any coordinate will update the window immediately.

Setting the left or top coordinates moves the window left or up respectively; these coordinates

have no effect on the width and height of the window. Likewise, setting the width and height does not affect the left and top coordinates.

Previous Home Topic Contents Index Next

CBaseControlWindow::put_WindowState

CBaseControlWindow Class

Sets the window state.

HRESULT put_WindowState(
long *WindowState*
);

Parameters

WindowState
New window state.

Return Values

Returns an HRESULT value.

Remarks

This member function takes the same parameters as the Microsoft Win32 ShowWindow function (for example, WS_SHOWNORMAL, WS_SHOWMINNOACTIVATE, and WS_SHOWMAXIMIZED).

Previous Home Topic Contents Index Next

CBaseControlWindow::put_WindowStyle

CBaseControlWindow Class

Sets the standard Windows-based styles.

HRESULT put_WindowStyle(

long *WindowStyle*
);

Parameters

WindowStyle
New window styles.

Return Values

Returns an HRESULT value.

Remarks

Take care when changing the window styles. In most cases, an application should retrieve the current style and then add or remove the inappropriate bits. This procedure allows various internal window styles used by Windows® to remain intact.

Previous Home Topic Contents Index Next

CBaseControlWindow::put_WindowStyleEx

CBaseControlWindow Class

Sets the style of the control window.

HRESULT put_WindowStyleEx(
long *WindowStyleEx*
);

Parameters

WindowStyleEx
[in] Value that specifies the style of the control window.

Return Values

Returns NOERROR.

Remarks

This method uses EX (extended) window styles. For a complete list of extended window styles, see the Microsoft Win32 CreateWindowEx function. To change the window style, retrieve the current window style, and then add or remove the necessary bit fields.

Note: Do not use the following window styles because they are not validated.

WS_DISABLED
WS_HSCROLL
WS_ICONIC
WS_MAXIMIZE
WS_MINIMIZE
WS_VSCROLL

With some exceptions (noted here), the acceptable flags are the same as those allowed by the Win32 CreateWindow function.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseControlWindow::SetControlWindowPin

CBaseControlWindow Class

Sets the pin with which to synchronize.

void SetControlWindowPin(
CBasePin *pPin*
);

Parameters

pPin
Pin with which the interface is synchronized.

Return Values

No return value.

Remarks

This member function sets the m_pPin variable equal to the *pPin* parameter. As described in the constructor, the interface can be called only when the filter has been connected successfully. The object is passed in through this member function to the pin with which it should synchronize; in most cases, it will determine if the pin is connected whenever it has an interface method called and will return VFW_E_NOT_CONNECTED if it fails.

Previous Home Topic Contents Index Next

CBaseControlWindow::SetWindowForeground

CBaseControlWindow Class

Moves the video window to the foreground and optionally gives it focus.

HRESULT SetWindowForeground(
long *Focus*
);

Parameters

Focus
Long value that specifies whether the video window will get focus. A value of –1 gives the window focus and 0 does not.

Return Values

Returns one of the following values.

Value	Meaning
NOERROR	The method succeeded.
E_INVALIDARG	*Focus* doesn't equal –1 or 0.
VFW_E_NOT_CONNECTED	The current filter isn't connected to a complete filter graph.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseControlWindow::SetWindowPosition

CBaseControlWindow Class

Sets the window position on the desktop.

HRESULT SetWindowPosition(
long *Left*,
long *Top*,
long *Width*,
long *Height*
);

Parameters

Left
New left coordinate.
Top
New top coordinate.
Width
Width of the window.
Height
Height of the window.

Return Values

Returns an HRESULT value.

‹Previous Home Topic Contents Index Next›

CBaseDispatch Class

The **CBaseDispatch** class is a base class that implements the IDispatch interface for use in a dual interface. A *dual interface* provides Automation and custom interface access to an interface.

CMediaControl and CMediaPosition (and other dual-interface support classes) are derived from this class or have members that are instances of this class.

For more information about the IDispatch methods, see the COM documentation included with the Microsoft® Platform Software Development Kit (SDK).

Member Functions

Name	Description
CBaseDispatch	Constructs a CBaseDispatch object.

Implemented IDispatch Methods

Name	Description
GetIDsOfNames	Maps a single member function and an optional set of parameters to a corresponding set of integer dispatch identifiers, which can be used upon subsequent calls to the IDispatch::Invoke method.
GetTypeInfo	Retrieves a type-information object, which can retrieve the type information for an interface.
GetTypeInfoCount	Retrieves the number of type-information interfaces provided by an object.

‹Previous Home Topic Contents Index Next›

CBaseDispatch::CBaseDispatch

CBaseDispatch Class

Constructs a CBaseDispatch object.

CBaseDispatch( );

Return Values

No return value.

CBaseDispatch::GetIDsOfNames

CBaseDispatch Class

Maps a single member function and an optional set of parameters to a corresponding set of integer dispatch identifiers, which can be used upon subsequent calls to the IDispatch::Invoke method.

HRESULT GetIDsOfNames(
REFIID *riid*,
OLECHAR ** *rgszNames*,
UINT *cNames*,
LCID *lcid*,
DISPID * *rgdispid*
);

Parameters

riid
Reference identifier. Reserved for future use. Must be NULL.
rgszNames
Passed-in array of names to be mapped.
cNames
Count of the names to be mapped.
lcid
Local context in which to interpret the names.
rgdispid
Caller-allocated array, each element of which contains an identifier that corresponds to one of the names passed in the *rgszNames* parameter. The first element represents the member name; the subsequent elements represent each of the member's parameters.

Return Values

Returns one of the following values.

Value	Meaning
S_OK	Success.
E_OUTOFMEMORY	Out of memory.
DISP_E_UNKNOWNNAME	One or more of the names were not known. The returned DISPIDs contain DISPID_UNKNOWN for each entry that corresponds to an unknown name.
DISP_E_UNKNOWN_CLSID	The class identifier was not recognized.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseDispatch::GetTypeInfo

CBaseDispatch Class

Retrieves a type-information object, which can retrieve the type information about an interface.

HRESULT GetTypeInfo(
UINT *itinfo*,
LCID *lcid*,
ITypeInfo ** *pptinfo*
);

Parameters

itinfo
Type information to return. Pass zero to retrieve type information for the IDispatch implementation.

lcid
Local identifier for the type information. An object can return different type information for different languages. This capability is important for classes that support localized member names. For classes that do not support localized member names, ignore this parameter.

pptinfo
Pointer to the type-information object requested.

Return Values

Returns an E_POINTER if *pptinfo* is invalid. Returns TYPE_E_ELEMENTNOTFOUND if *itinfo* is not zero. Returns S_OK if is successful. Otherwise, returns an HRESULT from one of the calls to retrieve the type. The **HRESULT** indicates the error and can be one of the following standard constants, or other values not listed:

Value	Meaning
E_FAIL	Failure.
E_POINTER	Null pointer argument.
E_INVALIDARG	Invalid argument.
S_OK or NOERROR	Success.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseDispatch::GetTypeInfoCount

CBaseDispatch Class

Retrieves the number of type-information interfaces provided by an object.

HRESULT GetTypeInfoCount(
UINT * *pctinfo*
);

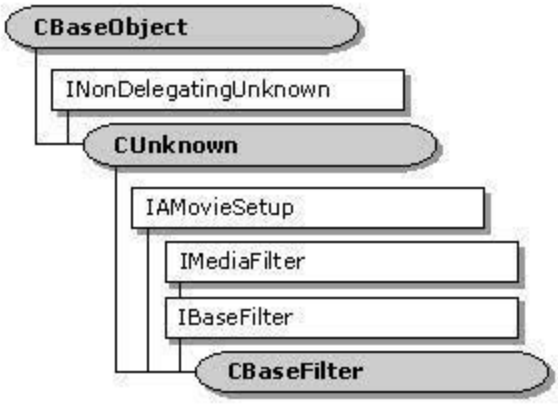
Parameters

pctinfo
Pointer to the location that receives the number of type-information interfaces that the object provides. If the object provides type information, this number is 1; otherwise, the number is 0.

Return Values

Returns E_POINTER if *pctinfo* is invalid; otherwise, returns S_OK.

CBaseFilter Class



CBaseFilter is an abstract base class from which all filters are derived. It supports the Component Object Model (COM) IBaseFilter interface and is derived from the CUnknown class. This class supports the enumeration of pins by calling the pure virtual member functions CBaseFilter::GetPin and GetPinCount. These member functions must be overridden by any derived class.

The **CBaseFilter** class assumes that all the filter's pins are derived from the CBasePin class. CBaseFilter::GetPin must return a pointer to **CBasePin**.

All member functions in this class that return HRESULT and accept a pointer as a parameter return E_POINTER when passed a null pointer.

Protected Data Members

Name	Description
m_clsid	Class identifier (CLSID) used for serialization using IPersist.
m_pClock	Filter graph's reference clock.
m_pGraph	Pointer to a graph to which this filter belongs.
m_PinVersion	Current version of the pins used on the filter.
m_pLock	Pointer to the critical section used for locking.
m_pName	Filter name.
m_pSink	Pointer to the IMediaEventSink interface on the filter graph manager.
m_State	Current state: running or paused.
m_tStart	Offset from the stream time to the reference time.

Member Functions

Name	Description
CBaseFilter	Constructs a CBaseFilter object.
GetFilterGraph	Returns the filter graph associated with the filter. This is used in the implementation of CEnumPins.
IncrementPinVersion	Adds 1 to the pin version stored in m_PinVersion.
IsActive	Determines if the filter is currently active (running or paused) or stopped.
NotifyEvent	Sends an event notification to the filter graph.
ReconnectPin	Requests pin for a reconnect.

Overridable Member Functions

Name	Description
GetPin	Returns a pointer to the requested pin.
GetPinCount	Returns the number of pins currently available on this object.
GetPinVersion	Returns the current version of the base filter for comparison with the version with which the pin was initialized. This member function can be overridden if pins are being created dynamically.
GetSetupData	Retrieves the registration data associated with the filter.
StreamTime	Returns the current stream time.

Implemented IPersist Methods

Name	Description
GetClassID	Returns the class identifier of this filter.

Implemented IMediaFilter Methods

Name	Description
GetState	Retrieves the current state of the filter.
GetSyncSource	Retrieves the current reference clock in use by this filter.
Pause	Instructs the filter to transition to State_Paused state.
Run	Instructs the filter to transition to State_Running state. Passes a time value to synchronize independent streams.
SetSyncSource	Informs the filter of the reference clock with which it should synchronize activity.
Stop	Instructs the filter to transition to the State_Stopped state.

Implemented IBaseFilter Methods

Name	Description
EnumPins	Provides an enumerator for this pin's preferred media types (implemented by this class).
FindPin	Retrieves the pin with the specified identifier.
JoinFilterGraph	Notifies a filter that it has joined a filter graph (implemented by this class).
QueryFilterInfo	Gets information about the specified filter (implemented by this class).
QueryVendorInfo	Retrieves optional information supplied by a vendor for the specified filter.

Implemented IAMovieSetup Methods

Name	Description
Register	Adds the filter to the registry.
Unregister	Removes the filter from the registry.

Implemented INonDelegatingUnknown Methods

Name	Description
NonDelegatingQueryInterface	Passes out pointers to any interfaces added to the derived filter class.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseFilter::CBaseFilter

CBaseFilter Class

Constructs a CBaseFilter object.

```
CBaseFilter(
  TCHAR *pName,
  LPUNKNOWN pUnk,
  CCritSec *pLock,
  REFCLSID clsid
  );
```

Parameters

pName
Pointer to an object description.

pUnk
IUnknown interface of the delegating object.

pLock
Pointer to an object that maintains the lock.

clsid
Class identifier to be used to serialize this filter.

Return Values

No return value.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseFilter::EnumPins

CBaseFilter Class

Retrieves an IEnumPins pointer that can be used to enumerate all the pins available on this filter.

HRESULT EnumPins(
IEnumPins ** *ppEnum*
);

Parameters

ppEnum
Pointer to the IEnumPins interface to retrieve.

Return Values

Returns E_OUTOFMEMORY if a new enumerate could not be created or NOERROR if successful.

Remarks

This member function implements the IBaseFilter::EnumPins method. It uses the CEnumPins object to construct an enumerator and retrieves the IEnumPins interface from the **CEnumPins** object. The implementation of CEnumPins::Next calls the CBaseFilter::GetPin member function, which the derived class must provide. The **IEnumPins** interface is used by the filter graph manager when adding the filter to the filter graph.

Previous Home Topic Contents Index Next

CBaseFilter::FindPin

CBaseFilter Class

Retrieves the pin with the specified identifier.

HRESULT FindPin(
LPCWSTR *Id*,

IPin ***ppPin*
);

Parameters

Id
Identifier of the pin.
ppPin
Pointer to the IPin interface for this pin after the filter has been restored.

Return Values

The default implementation by this member function returns S_OK if the pin name was found or VFW_E_NOT_FOUND otherwise.

Remarks

This member function provides a base class definition of the IBaseFilter::FindPin method that, along with the IPin::QueryId method, is used to implement persistent filter graphs. A filter must be able to translate the IPin interface pointers to its pins into identifiers that can be saved along with the configuration of the filter graph. It does this by using the **IPin::QueryId** method. It must then be able to convert those identifiers back into **IPin** interface pointers when the filter and its connections are restored as part of a persistent filter graph. This is accomplished using the **IBaseFilter::FindPin** method.

By default, the base classes use the pin name in the CBasePin::m_pName data member, so implementing this member function in your derived filter class is not normally required.

The *ppPin* parameter is set to NULL if the identifier cannot be matched.

Previous Home Topic Contents Index Next

CBaseFilter::GetClassID

CBaseFilter Class

Fills the *pClsID* parameter with the class identifier of this filter (from m_clsid).

HRESULT GetClassID(
CLSID **pClsID*
);

Parameters

pClsID
Pointer to the class identifier to be filled out.

Return Values

Return NOERROR.

Previous Home Topic Contents Index Next

CBaseFilter::GetFilterGraph

CBaseFilter Class

Retrieves the filter graph associated with the filter.

IFilterGraph *GetFilterGraph( );

Return Values

Returns the value of m_pGraph.

Previous Home Topic Contents Index Next

CBaseFilter::GetPin

CBaseFilter Class

Retrieves a CBasePin object on the filter.

virtual CBasePin *GetPin(
int *n*
) PURE;

Parameters

n
Number of the specified pin.

Return Values

Returns a pointer to the pin specified by the *n* parameter.

Remarks

Override this member function to return a pointer to the *n*th pin on this filter. CBaseFilter adds a reference to it, when necessary, before passing it to any other object. This member function is called by the base class CEnumPins::Next member function to retrieve pins for the IEnumPins interface, which is used by the filter graph manager.

Previous Home Topic Contents Index Next

CBaseFilter::GetPinCount

CBaseFilter Class

Retrieves the number of supported pins.

virtual int GetPinCount( ) PURE;

Return Values

Returns the pin count.

Remarks

Override this member function to return the count of pins currently available on this object.

Previous Home Topic Contents Index Next

CBaseFilter::GetPinVersion

CBaseFilter Class

Retrieves the version number of the pin.

virtual long GetPinVersion( );

Return Values

By default, returns the value of m_PinVersion. If overridden, this member function should

return the pin version number.

Remarks

Returns the current version of the filter that matches the version used to initialize the pin. The enumerator calling this member function performs the matching.

A filter provides an enumerator to gain access to the input and output pins it keeps. Each time a pin enumerator's method is called, the pin enumerator calls the **CBaseFilter::GetPinVersion** member function to ensure that the base filter's version matches the version with which the pin enumerator was initialized.

A filter class can override **CBaseFilter::GetPinVersion** if there is a need to increment the version by changing the available pins dynamically. Or, it can more easily call IncrementPinVersion.

GetPinVersion does not lock the filter because the enumerators are designed to be separate objects. The derived class's **GetPinVersion** will likely have to do some specialized locking with the part of the object responsible for creating and deleting pins.

Previous Home Topic Contents Index Next

CBaseFilter::GetSetupData

CBaseFilter Class

Retrieves the registration data associated with the filter.

virtual LPAMOVIESETUP_FILTER GetSetupData( );

Return Values

Returns a pointer to an AMOVIESETUP_FILTER structure containing registration information for the filter.

Remarks

You must override this member function and implement it to return an AMOVIESETUP_FILTER structure containing its associated AMOVIESETUP_PIN and AMOVIESETUP_MEDIATYPE structures for pin and media type information. This member function is called from the CBaseFilter::Register member function.

Previous Home Topic Contents Index Next

CBaseFilter::GetState

CBaseFilter Class

Retrieves the current state of the filter.

HRESULT GetState(
DWORD *dwMilliSecsTimeout*,
FILTER_STATE * *State*
);

Parameters

dwMilliSecsTimeout
Duration of the time-out, in milliseconds.
State
Holds the returned state of the filter.

Return Values

Returns S_OK.

Remarks

This member function implements the IMediaFilter::GetState method. It returns the value of the m_State data member. Override this member function if the state changes in your filter are not synchronous.

‹Previous Home Topic Contents Index Next›

CBaseFilter::GetSyncSource

CBaseFilter Class

Retrieves the current reference clock in use by this filter.

HRESULT GetSyncSource(
IReferenceClock ** *pClock*
);

Parameters

pClock
Pointer to a reference clock; will be set to the IReferenceClock interface.

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IMediaFilter::GetSyncSource method. It returns the value of m_pClock after adding a reference to it. Be sure to release the interface by calling the IUnknown::Release method when finished with the pointer.

Previous Home Topic Contents Index Next

CBaseFilter::IncrementPinVersion

CBaseFilter Class

Adds 1 to the version number of the pin.

void IncrementPinVersion( );

Return Values

No return value.

Remarks

By default, increments the value of m_PinVersion.

Previous Home Topic Contents Index Next

CBaseFilter::IsActive

CBaseFilter Class

Determines if the filter is currently active (running or paused) or stopped.

BOOL IsActive(void);

Return Values

Returns TRUE if the filter is paused or running, or FALSE if the filter is stopped.

Previous Home Topic Contents Index Next

CBaseFilter::JoinFilterGraph

CBaseFilter Class

Notifies a filter that it has joined a filter graph.

HRESULT JoinFilterGraph(
IFilterGraph * *pGraph*,
LPCWSTR *pName*
);

Parameters

pGraph
Pointer to the filter graph to join.
pName
[in, string] Name of the filter being added.

Return Values

No return value.

Remarks

This member function implements the IBaseFilter::JoinFilterGraph method. It assigns the *pGraph* filter graph pointer to the m_pGraph data member and obtains the IMediaEventSink interface from the filter graph manager to allow the filter to post event notifications to the filter graph manager.

The filter should store the IMediaEventSink interface for later use, because it might need to notify the interface about events, but it should not increase the reference count on the filter graph manager object. A null pointer indicates that the filter is no longer part of a graph.

Previous Home Topic Contents Index Next

CBaseFilter::NonDelegatingQueryInterface

CBaseFilter Class

Retrieves an interface and increments the reference count.

HRESULT NonDelegatingQueryInterface(
REFIID *riid*,
void ** *ppv*
);

Parameters

riid
Reference identifier.
ppv
Pointer to the interface.

Return Values

Returns E_POINTER if *ppv* is invalid. Returns NOERROR if the query is successful or E_NOINTERFACE if it is not.

Remarks

This member function implements the INonDelegatingUnknown::NonDelegatingQueryInterface method and passes out references to the IBaseFilter, IMediaFilter, IPersist, IAMovieSetup, and IUnknown interfaces. Override this class to return other interfaces on the object in the derived class.

Previous Home Topic Contents Index Next

CBaseFilter::NotifyEvent

CBaseFilter Class

Sends an event notification to the filter graph.

HRESULT NotifyEvent(
long *EventCode*,
long *EventParam1*,

long *EventParam2*
);

Parameters

EventCode
Identifier of the event.
EventParam1
First parameter of the event.
EventParam2
Second parameter of the event.

Return Values

Returns S_OK if delivered, S_FALSE if the filter graph does not sink events, or an error otherwise.

Remarks

For a list of notification codes and event parameter values, see Event Notification Codes.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseFilter::Pause

CBaseFilter Class

Transitions the filter to State_Paused state if it is not in this state already.

HRESULT Pause (void);

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IMediaFilter::Pause method. If the filter is in State_Stopped state, the CBasePin::Active member function is called for each of the filter's pins to which it is connected. If this member function succeeds, the filter's m_State member variable is set to State_Paused. If any pin returns a failure return value from its Active method, the function fails and the state is not changed.

This member function holds the filter's lock.

Previous Home Topic Contents Index Next

CBaseFilter::QueryFilterInfo

CBaseFilter Class

Retrieves information about the filter.

```
HRESULT QueryFilterInfo(
  FILTER_INFO * pInfo
  );
```

Parameters

pInfo
Pointer to a FILTER_INFO structure to fill in.

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IBaseFilter::QueryFilterInfo method. It copies the filter's name from m_pName, and copies the pointer to the filter graph interface from m_pGraph into the FILTER_INFO structure before returning.

Note that the IFilterGraph interface passed out by this member function is reference counted, and so must be released when the caller has finished with it.

Previous Home Topic Contents Index Next

CBaseFilter::QueryVendorInfo

CBaseFilter Class

Retrieves a vendor information string.

HRESULT QueryVendorInfo(
LPWSTR * *pVendorInfo*
);

Parameters

pVendorInfo
Pointer to a string containing vendor information.

Return Values

Returns an HRESULT value (E_NOTIMPL by default).

Remarks

This member function implements the IBaseFilter::QueryVendorInfo method, but only to return E_NOTIMPL. Filters that want to expose vendor information must override this member function. If implemented in a derived class, callers should free memory when they are done using it by calling the Microsoft® Win32® CoTaskMemFree function.

Previous Home Topic Contents Index Next

CBaseFilter::ReconnectPin

CBaseFilter Class

Requests pin for a reconnect.

HRESULT ReconnectPin(
IPin **pPin*,
AM_MEDIA_TYPE const **pmt*
);

Parameters

pPin
Pointer to the pin to reconnect.
pmt
AM_MEDIA_TYPE media type to reconnect with. This can be NULL.

Return Values

Returns an HRESULT value.

Remarks

This function calls the IFilterGraph2::ReconnectEx method on the filter graph.

Previous Home Topic Contents Index Next

CBaseFilter::Register

CBaseFilter Class

Adds the filter to the registry.

HRESULT Register( );

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IAMovieSetup::Register method and registers the filter, its pins, and the media type associated with the pins. It does this by first calling GetSetupData to retrieve the setup data, and then calling the IFilterMapper::RegisterFilter, IFilterMapper::RegisterPin, and IFilterMapper::RegisterPinType methods.

Previous Home Topic Contents Index Next

CBaseFilter::Run

CBaseFilter Class

Transitions the filter from paused to running state if it is not in this state already.

HRESULT Run (
REFERENCE_TIME *tStart*
);

Parameters

tStart
Reference time value corresponding to stream time 0.

Return Values

Returns an HRESULT value. The default implementation returns NOERROR.

Remarks

If the filter is in State_Stopped state, the CBaseFilter::Pause method is called first to transition the filter to State_Paused state, which has the effect of activating any of the filter's connected pins. If any pin returns a failure return code from its Active method, the function fails and the state is not changed. If this member function succeeds, the filter's m_State member variable is set to State_Running.

This member function holds the filter's lock.

Previous Home Topic Contents Index Next

CBaseFilter::SetSyncSource

CBaseFilter Class

Identifies the reference clock to which the filter should synchronize activity.

```
HRESULT SetSyncSource (
  IReferenceClock * pClock
  );
```

Parameters

pClock
Pointer to the IReferenceClock interface.

Return Values

Returns an HRESULT value. The default implementation returns NOERROR.

Remarks

This member function implements the IMediaFilter::SetSyncSource method. It sets the m_pClock data member to the *pClock* parameter and increments the reference count on the IReferenceClock interface passed in.

This member function is most important to rendering filters and might not apply to other

filters.

CBaseFilter::Stop

CBaseFilter Class

Transitions the filter to State_Stopped state if it is not in this state already.

HRESULT Stop(void);

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IMediaFilter::Stop method. It first calls the CBasePin::Inactive member function on all its pins that have a connection, and then sets the filter's m_State member variable to State_Stopped.

This member function holds the filter's lock.

CBaseFilter::StreamTime

CBaseFilter Class

Retrieves the current stream time.

virtual HRESULT StreamTime(
CRefTime& *rtStream*
);

Parameters

rtStream

Current stream time.

Return Values

Returns an HRESULT value, which can include the following values.

Value	Meaning
E_FAIL	Unable to get time from clock.
S_OK	Stream time returned in the *rtStream* parameter.
VFW_E_NO_CLOCK	No reference clock is available.

Remarks

Current stream time is the reference clock time minus the stream time offset. All samples with time stamps less than or equal to this time should have been presented.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseFilter::Unregister

CBaseFilter Class

Removes the filter from the registry.

HRESULT Unregister( );

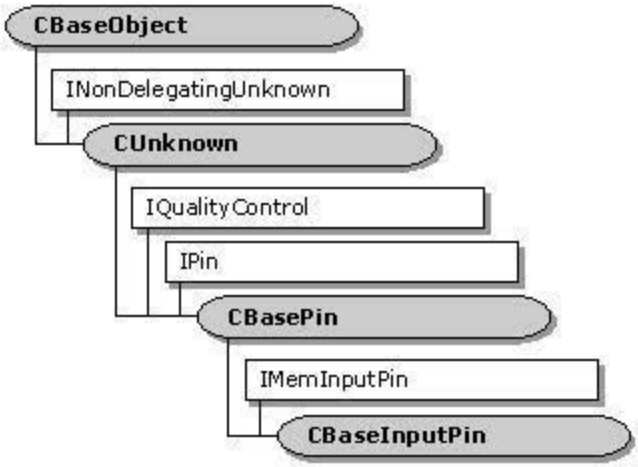
Return Values

Returns an HRESULT value.

Remarks

This member function implements the IAMovieSetup::Unregister method and calls the IFilterMapper::UnregisterFilter method to remove the filter from the registry. This effectively removes the pins and media types as well.

CBaseInputPin Class



CBaseInputPin is an abstract base class derived from CBasePin that adds support for IMemInputPin in addition to the IPin interface support provided by **CBasePin**. Its IMemInputPin::GetAllocator method returns a CMemAllocator object. Derive your input pin from this class.

All member functions in this class that return HRESULT and accept a pointer as a parameter return E_POINTER when passed a null pointer.

Protected Data Members

Name	Description
m_bFlushing	In the state of flushing; if TRUE, all IMemInputPin::Receive methods are returned with S_FALSE.
m_bReadOnly	If TRUE, indicates that the allocator being used contains samples that are read-only.
m_pAllocator	Pointer to the default memory allocator.

Member Functions

Name	Description
CBaseInputPin	Constructs a CBaseInputPin object.
IsReadOnly	Checks the m_bReadOnly data member and returns its value.
IsFlushing	Checks the m_bFlushing data member and returns its value.
PassNotify	Passes a quality-control notification to the appropriate sink.

Overridable Member Functions

Name	Description
CheckStreaming	Verifies conditions for continuing with a streaming operation.
Inactive	Switches the pin to an inactive state.

Implemented IPin Methods

Name	Description
BeginFlush	Informs the pin to begin a flush operation.
Disconnect	Releases the stored allocator.
EndFlush	Informs the pin to end a flush operation.

Implemented IMemInputPin Methods

Name	Description
GetAllocator	Returns the allocator interface that this input pin would like the output pin to use.
GetAllocatorRequirements	Indicates an optional method to use if the filter has specific alignment or prefix requirements but could use an upstream allocator.
NotifyAllocator	Tells the input pin which allocator the output pin is actually going to use.
Receive	Returns the next block of data from the stream. (Override this method to process a sample being passed in.)
ReceiveCanBlock	Determines if sending an IMemInputPin::Receive method might block.
ReceiveMultiple	Returns the next block of data from the stream. (Override this method to process samples being passed in.)

Implemented INonDelegatingUnknown Methods

Name	Description
NonDelegatingQueryInterface	Retrieves an interface from the subobject, not the aggregated object.

Implemented IQualityControl Methods

Name	Description
Notify	Notifies the recipient that a quality-control change is requested. (Override on the output pin only. This implementation returns NOERROR.)

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseInputPin::BeginFlush

CBaseInputPin Class

Informs the pin to begin a flush operation.

HRESULT BeginFlush(void);

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IPin::BeginFlush method. When this method is called, the pin is entering flush state. You must override this method in your derived class, but you should call this base class first in your implementation, because it sets m_bFlushing so that no more IMemInputPin::Receive calls will succeed.

The overriding member function should then carry out the following steps.

1. Discard any queued data.
2. Free any pin blocked by the Receive method.
3. Pass the IPin::BeginFlush method to any downstream pins.

IPin::BeginFlush is not logically part of the media stream and can be optimized in the sense that if a pin has passed no data downstream before this method is called, there is no need to pass this notification on.

An example of an overriding implementation of this member function can be found in the CTransformInputPin::BeginFlush member function, which uses the CBaseOutputPin::DeliverBeginFlush member function to perform the last step.

Previous Home Topic Contents Index Next

CBaseInputPin::CBaseInputPin

CBaseInputPin Class

Constructs a CBaseInputPin object.

CBaseInputPin::CBaseInputPin(
TCHAR *pObjectName*,
CBaseFilter *pFilter*,
CCritSec *pLock*,
HRESULT *phr*,
LPCWSTR *pPinName*
);

Parameters

pObjectName
Name of the class object.
pFilter
Pointer to the filter that owns this pin.
pLock
Pointer to the CCritSec critical section object used to lock the pin.
phr
Pointer to the general COM return value. This value is changed only if this function fails.
pPinName
Name of the pin.

Return Values

No return value.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseInputPin::CheckStreaming

CBaseInputPin Class

Verifies conditions for continuing with a streaming operation.

virtual HRESULT CheckStreaming( );

Return Values

Returns one of the following HRESULT values, depending on the state.

Value	Meaning
S_FALSE	Currently in flushing state.
S_OK	Receive or EndOfStream operations can safely proceed.
VFW_E_RUNTIME_ERROR	Run-time error occurred while processing a previous sample.
VFW_E_WRONG_STATE	Filter is in the State_Stopped state.

Remarks

Conditions checked in this member function include whether the filter is connected, if it is in an active state, if it is not currently flushing data, and if it has not just issued a run-time error. If all these conditions pass, it returns S_OK.

You can override this member function to add restrictions defined by your derived class. The overriding member function should call this base class implementation to check for conditions