here as well.

This function member should be called from any override of the CBaseInputPin::Receive or CBasePin::EndOfStream member function (or they should do some equivalent check).

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseInputPin::Disconnect

CBaseInputPin Class

Releases the stored allocator.

HRESULT Disconnect( );

Return Values

Returns an HRESULT value.

Remarks

This member function overrides the CBasePin::Disconnect member function. It calls **CBasePin::Disconnect** first, and then releases the allocator held by m_pAllocator.

Previous Home Topic Contents Index Next

CBaseInputPin::EndFlush

CBaseInputPin Class

Informs the pin to end a flush operation and notifies the pin that it can start accepting data again.

HRESULT EndFlush(void);

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IPin::EndFlush method. When this method is called, the pin is beginning to end a flush operation. Your derived class must override this member function, but should call this member function at the end of your implementation to clear m_bFlushing so that IMemInputPin::Receive calls will succeed.

Before calling this base class implementation, your overriding member function should perform the following steps.

1. Ensure that your filter will not push any additional data. (To do this, synchronize with a thread, stop it pushing, and discard any queued data.)
2. Pass the EndFlush method downstream by calling the method on the downstream filter's input pin.

IPin::EndFlush is not logically part of the media stream. It 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::EndFlush member function, which uses the CBaseOutputPin::DeliverEndFlush member function to perform the last step.

Previous Home Topic Contents Index Next

CBaseInputPin::GetAllocator

CBaseInputPin Class

Retrieves the allocator interface that this input pin identifies as the interface for the output pin to use.

HRESULT GetAllocator(
IMemAllocator ** *ppAllocator*
);

Parameters

ppAllocator
Pointer to an obtained IMemAllocator object.

Return Values

Default implementation returns either E_OUTOFMEMORY, if an allocator cannot be created, or NOERROR upon success.

Remarks

This member function implements the IMemInputPin::GetAllocator method, which is called by the connected output pin to retrieve an allocator to use for transporting media samples. By default, this member function creates a CMemAllocator object and obtains the IMemAllocator interface, to which it adds a reference count for the pin when assigning it to the m_pAllocator data member, and adds another reference count before passing it back to the output pin.

Override this member function if your filter has another allocator, such as one from a downstream pin, or a specialized allocator to offer the connected output pin.

Previous Home Topic Contents Index Next

CBaseInputPin::GetAllocatorRequirements

CBaseInputPin Class

Optional member function to use if the filter has specific alignment or prefix requirements but could use an upstream allocator.

HRESULT GetAllocatorRequirements(
ALLOCATOR_PROPERTIES * *pProps*
);

Parameters

pProps
ALLOCATOR_PROPERTIES structure containing the required size, count, and alignment of the allocator.

Return Values

Returns an HRESULT value. Returns E_NOTIMPL by default.

Remarks

Override this member function if you have specific alignment or prefix requirements but could use an upstream allocator.

Previous Home Topic Contents Index Next

CBaseInputPin::Inactive

CBaseInputPin Class

Releases the allocator's memory.

HRESULT Inactive(void);

Return Values

Returns an HRESULT value.

Remarks

This member function is called through IMediaFilter, which is responsible for locking the object first.

Previous Home Topic Contents Index Next

CBaseInputPin::IsFlushing

CBaseInputPin Class

Checks the m_bFlushing data member and returns its value.

BOOL IsFlushing(void);

Return Values

Returns TRUE if the input pin is flushing data; otherwise, returns FALSE.

Previous Home Topic Contents Index Next

CBaseInputPin::IsReadOnly

CBaseInputPin Class

Checks the m_bReadOnly data member and returns its value.

BOOL IsReadOnly(void);

Return Values

Returns TRUE if the allocator has read-only samples; otherwise, returns FALSE.

Previous Home Topic Contents Index Next

CBaseInputPin::NonDelegatingQueryInterface

CBaseInputPin 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 IMemInputPin and IUnknown interfaces. Override this class to return other interfaces on the object in the derived class.

Previous Home Topic Contents Index Next

CBaseInputPin::Notify

CBaseInputPin Class

Notifies the recipient that a quality change is requested.

HRESULT Notify(
IBaseFilter * *pSelf*,
Quality *q*
);

Parameters

pSelf
Pointer to the filter that is sending the quality notification.
q
Quality notification structure.

Return Values

Returns NOERROR by default.

Remarks

The IQualityControl::Notify method is usually implemented on the output pin, because quality-control messages are passed upstream, and not on the input pin.

Previous Home Topic Contents Index Next

CBaseInputPin::NotifyAllocator

CBaseInputPin Class

Notifies the input pin as to which allocator the output pin is actually going to use.

HRESULT NotifyAllocator(
IMemAllocator * *pAllocator*,
BOOL *bReadOnly*
);

Parameters

pAllocator
Pointer to the IMemAllocator object to use. This might or might not be the same **IMemAllocator** object that the input pin provided in the IMemInputPin::GetAllocator method (the output pin could provide its own allocator).
bReadOnly
Flag to indicate if the samples from this allocator are read-only.

Return Values

Default implementation returns NOERROR.

Remarks

This member function implements the IMemInputPin::NotifyAllocator method, which is called by the connected output pin to inform the input pin of the chosen allocator for the memory transport. Override this member function if your filter cares about this information. By default, this sets the m_pAllocator data member to the allocator interface passed in after adding a reference count to that interface.

Previous Home Topic Contents Index Next

CBaseInputPin::PassNotify

CBaseInputPin Class

Passes a quality-control notification to the appropriate sink.

HRESULT PassNotify(
Quality *q*
);

Parameters

q
Quality-control notification object.

Return Values

Returns VFW_E_NOT_FOUND if no quality sink is set and the upstream filter does not support the IQualityControl interface. Otherwise, returns the HRESULT value resulting from notifying the sink or the upstream filter.

Remarks

Output pins receive quality-control notifications and, if possible, filters act on them to degrade

appropriately. Often, filters cannot respond to the notifications; in this case the notification should be passed to the quality-control sink or, by default, upstream to the next filter. The **PassNotify** member function is called from the CTransformOutputPin::Notify member function when a notification requires passing. The Quality structure passed is the one that the output pin received.

CBaseInputPin::Receive

CBaseInputPin Class

Retrieves the next block of data from the stream.

HRESULT Receive(
IMediaSample * *pSample*
);

Parameters

pSample
Pointer to a media sample.

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IMemInputPin::Receive method. It first checks that it can process the sample by calling CBaseInputPin::CheckStreaming; if that member function does not return S_OK, **Receive** returns immediately with the value returned by **CBaseInputPin::CheckStreaming**.

This base class member function checks to see if the format has changed with this media sample; if so, it checks that the filter will accept it, generating a run-time error if not. If a run-time error is raised, the m_bRunTimeError data member is set so that no more samples will be accepted.

The overriding member function does something with the passed-in sample, such as calling a member function to transform it or pass it downstream.

This is a blocking synchronous call. Typically no blocking occurs, but if a filter cannot process the sample immediately, it can use the calling application's thread to wait until it can.

Call the IUnknown::AddRef method if you must hold the returned data block beyond the

completion of the IMemInputPin::Receive method. If you call AddRef, be sure to call IUnknown::Release when done with it.

Previous Home Topic Contents Index Next

CBaseInputPin::ReceiveCanBlock

CBaseInputPin Class

Determines if the implementation of the IMemInputPin::Receive method might block on the connected output pin.

HRESULT ReceiveCanBlock(void);

Return Values

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

Value	Meaning
S_FALSE	Input pin will not block on a Receive method.
S_OK	Input pin might block on a Receive method.

Remarks

This member function implements the IMemInputPin::ReceiveCanBlock method. The base class implementation calls the **IMemInputPin::ReceiveCanBlock** method on the input pin connected to each of the filter's output pins.

This member function is useful because an output pin from a filter might require notification if its thread might be blocked when it calls the Receive method on the connected input pin. For example, a source filter might prefer to keep reading and buffering data rather than be blocked, and might choose to start another thread to wait on the blocking **Receive** method. See the COutputQueue base class for queuing samples to input pins that potentially block.

Previous Home Topic Contents Index Next

CBaseInputPin::ReceiveMultiple

CBaseInputPin Class

Retrieves the next block of data from the stream. This method behaves much like the IMemInputPin::Receive method, but it works with multiple samples. Override this function if you can usefully process samples in batches.

HRESULT ReceiveMultiple(
IMediaSample ** *pSamples*,
long *nSamples*,
long * *nSamplesProcessed*
);

Parameters

pSamples
Pointer to an array of samples.
nSamples
Number of samples to process.
nSamplesProcessed
Number of samples processed.

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IMemInputPin::ReceiveMultiple method. It is implemented to call the CBaseInputPin::Receive member function in a loop for *nSamples* number of iterations.

CBaseList Class

The **CBaseList** class represents a list of pointers to objects. No storage management or copying is done on the objects that are pointed to.

The implementation allows for objects to be on multiple lists simultaneously and does not require support in the objects themselves; therefore, it is particularly useful for holding variable-length lists of interface pointers.

The implementation is not multithread safe. External locks are required to maintain the integrity of the list when it is accessed from more than one thread simultaneously.

The POSITION structure represents a position in a linked list that is actually a void pointer. A position represents a cursor on the list that can be set to identify any element. NULL is a valid value, and several operations regard NULL as the position that is "one step off the end of the list." (In an *n* element list there are *n*+1 places to insert, and NULL is that *n*+1 value.) The position of an element in the list is only invalidated if that element is deleted. Move operations might indicate that what was a valid position in one list is now a valid position in a different list.

Some operations, which at first sight seem illegal, are allowed as harmless null operations (no-ops). For example, the CBaseList::RemoveHeadI member function is legal on an empty list, and it returns NULL. This allows an atomic way to test if there is an element there and, if so, to retrieve it.

Single-element operations return positions, where a non-NULL value indicates that it worked. Entire list operations return a Boolean value, where TRUE indicates success.

Protected Data Members

Name	Description
m_Count	Number of nodes in the list.
m_pFirst	Pointer to the first node in the list.
m_pLast	Pointer to the last node in the list.

Member Functions

Name	Description
AddAfter	Inserts a list of nodes after the specified node.
AddAfterI	Inserts a node after the specified node.
AddBefore	Inserts a list of nodes before the specified node.
AddBeforeI	Inserts a node before the specified node.
AddHead	Inserts a list of nodes at the front of the list.
AddHeadI	Inserts a node at the front of the list.
AddTail	Appends a list of nodes to the end of the list.
AddTailI	Appends a node to the end of the list.
CBaseList	Constructs a CBaseList object.
FindI	Returns the first position that holds the specified object.
GetCountI	Returns the number of objects in the list.
GetHeadPositionI	Returns a cursor identifying the first element of the list.
GetI	Returns the object at the specified position.
GetNextI	Returns the specified object and updates the position.
GetTailPositionI	Returns a cursor identifying the last element of the list.
MoveToHead	Moves the node or list of nodes to the beginning of a second list.
MoveToTail	Moves the node or list of nodes to the end of a second list.
Next	Returns the next position in the list.
Prev	Returns the previous position in the list.
RemoveAll	Removes all nodes from the list.
RemoveHeadI	Removes the first node in the list.
RemoveI	Removes the specified node from the list.
RemoveTailI	Removes the last node in the list.
Reverse	Reverses the order of the pointers to the objects in the list.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseList::AddAfter

CBaseList Class

Inserts a list of nodes after the specified node.

```
BOOL AddAfter(
  POSITION pos,
  CBaseList *pList
  );
```

Parameters

pos
Position after which to add the list of nodes.
pList
Pointer to the list of objects to add.

Return Values

Returns TRUE if successful; otherwise, returns FALSE.

CBaseList::AddAfterI

CBaseList Class

Inserts a node after the specified node.

```
POSITION AddAfterI(
  POSITION pos,
  void * pObj
  );
```

Parameters

pos
Position after which to add the node.
pObj
Pointer to the object to add.

Return Values

Returns the position of the inserted object.

Remarks

The following member function adds *x* to the start, which is equivalent to calling the CBaseList::AddHeadI member function:

```
AddAfterI(NULL,x)
```

If the list insertion fails, some of the elements might have been added. Existing positions in the list, including the position specified in the *pos* parameter, remain valid. The following two member functions are equivalent even in cases where *pos* is NULL or the *Next(p)* parameter is

NULL. (This is similar for the mirror case.)

```
AddAfterI (p,x)
AddBeforeI(Next(p),x)
```


Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseList::AddBefore

CBaseList Class

Inserts a list of nodes before the specified node.

BOOL AddBefore(
POSITION *pos*,
CBaseList **pList*
);

Parameters

pos
Position before which to add the list of nodes.
pList
Pointer to the list of objects to add.

Return Values

Returns TRUE if successful; otherwise, returns FALSE.

Remarks

If the list insertion fails, some of the elements might have been added. Existing positions in the list, including the position specified in the *pos* parameter, remain valid.

Previous Home Topic Contents Index Next

CBaseList::AddBeforeI

CBaseList Class

Inserts a node before the specified node.

POSITION AddBeforeI(
POSITION *pos*,
void * *pObj*
);

Parameters

pos
Position before which to add the node or list of nodes.
pObj
Pointer to the object to add.

Return Values

Returns the position of the inserted object.

Remarks

The following member function adds the value specified in the *x* parameter to the end, which is equivalent to calling the CBaseList::AddTailI member function:

```
AddBeforeI(NULL,x)
```

The following two member functions are equivalent even in cases where *pos* is NULL or the *Next(p)* parameter is NULL. (This is similar for the mirror case.)

```
AddAfterI(p,x)
AddBeforeI(Next(p),x)
```


Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseList::AddHead

CBaseList Class

Inserts a list of nodes at the front of the list.

BOOL AddHead(

CBaseList **pList*
);

Parameters

pList
Pointer to the list of objects to add.

Return Values

No return value.

Remarks

If you are adding Component Object Model (COM) objects, you might want to add references to them (using the IUnknown::AddRef method) first. Other existing positions in the list remain valid.

This member function duplicates all the nodes in the *pList* parameter (that is, duplicates all its pointers to objects). It does not duplicate the objects.

Previous Home Topic Contents Index Next

CBaseList::AddHeadI

CBaseList Class

Inserts a node at the front of the list.

POSITION AddHeadI(
void * *pObj*
);

Parameters

pObj
Pointer to the object to add.

Return Values

Returns the new head position, or NULL if it fails. For list insertions, returns TRUE if successful; otherwise, returns FALSE.

Remarks

If you are adding Component Object Model (COM) objects, you might want to add references to

them (using the IUnknown::AddRef method) first. Other existing positions in the list remain valid.

Previous Home Topic Contents Index Next

CBaseList::AddTail

CBaseList Class

Appends a list of nodes to the end of the list.

BOOL AddTail(
CBaseList **pList*
);

Parameters

pList
Pointer to the list of objects to add.

Return Values

No return value.

Remarks

This member function duplicates all the nodes in *pList* (that is, duplicates all its pointers to objects). It does not duplicate the objects. Existing positions in the list remain valid.

Previous Home Topic Contents Index Next

CBaseList::AddTailI

CBaseList Class

Appends a single node to the end of the list.

POSITION AddTailI(
void * *pObj*

);

Parameters

pObj
Pointer to the object to add.

Return Values

Returns the new tail position, if successful; otherwise, returns NULL.

Previous Home Topic Contents Index Next

CBaseList::CBaseList

CBaseList Class

Constructs a CBaseList object.

CBaseList(
TCHAR **pName*,
INT *iItems*
);

CBaseList(
TCHAR **pName*
);

Parameters

pName
Name of the list.
iItems
Number of items in the list.

Return Values

No return value.

Previous Home Topic Contents Index Next

CBaseList::FindI

CBaseList Class

Retrieves the first position that holds the specified object.

POSITION FindI(
void * *pObj*
);

Parameters

pObj
Pointer to the object to find.

Return Values

Returns a position cursor.

Remarks

A position cursor identifies an element on the list. Use the CBaseList::GetI member function to return the object at this position.

Previous Home Topic Contents Index Next

CBaseList::GetCountI

CBaseList Class

Retrieves the number of objects (object count) in the list.

int GetCountI( );

Return Values

Returns the number of objects in the list.

Previous Home Topic Contents Index Next

CBaseList::GetHeadPositionI

CBaseList Class

Retrieves a cursor identifying the first element of the list.

POSITION GetHeadPositionI( );

Return Values

Returns a position cursor.

Remarks

A position cursor represents an element on the list. It is defined as a pointer to a void.

Previous Home Topic Contents Index Next

CBaseList::GetI

CBaseList Class

Retrieves the object at the specified position.

void *GetI(
POSITION *pos*
);

Parameters

pos
Position in the list from which to retrieve the object.

Return Values

Returns a pointer to the object as position *pos*.

Remarks

Use the CBaseList::Next, CBaseList::Prev, or CBaseList::FindI member function to obtain the position. Asking for the object at a NULL position returns NULL without generating an error.

Previous Home Topic Contents Index Next

CBaseList::GetNextI

CBaseList Class

Retrieves the specified object and updates the position.

void *GetNextI(
POSITION& *rp*
);

Parameters

rp
Returned pointer to the next object.

Return Values

Returns a pointer to an object at the next position.

Remarks

This member function updates the *rp* parameter to the next node in the list, but makes it NULL if it was at the end of the list.

This member function is retained only for backward compatibility. (`GetPrev` is not implemented.)

Use the CBaseList::Next and CBaseList::Prev member functions to access the next or previous object in the list.

Previous Home Topic Contents Index Next

CBaseList::GetTailPositionI

CBaseList Class

Retrieves a cursor identifying the last element of the list.

POSITION GetTailPositionI( );

Return Values

Returns a position cursor.

Remarks

A position cursor represents an element on the list. A position is defined as a pointer to a void.

Previous Home Topic Contents Index Next

CBaseList::MoveToHead

CBaseList Class

Moves the node or list of nodes to the beginning of a second list.

BOOL MoveToHead(
POSITION *pos*,
CBaseList **pList*
);

Parameters

pos
Position that marks the split in the list.

pList
List in which to add the section of the list preceding the position passed in the *pos* parameter.

Return Values

Returns TRUE if successful; otherwise, returns FALSE.

Remarks

This member function splits the current list after the position specified in the *pos* parameter in the list and retains the head portion of the original list. It then adds the tail portion to the head of the second list, identified by the *pList* parameter.

Previous Home Topic Contents Index Next

CBaseList::MoveToTail

CBaseList Class

Moves the node or list of nodes to the end of a second list.

```
BOOL MoveToTail(
  POSITION pos,
  CBaseList *pList
  );
```

Parameters

pos
Position that marks the split in the list.

pList
List in which to add the section of the list specified in the *pos* parameter.

Return Values

Returns TRUE if successful; otherwise, returns FALSE.

Remarks

This member function splits the current list after the position specified in the *pos* parameter in the list and retains the tail portion of the original list. It then adds the head portion to the tail end of the second list, using the *pList* parameter.

‹Previous Home Topic Contents Index Next›

CBaseList::Next

CBaseList Class

Retrieves the next position in the list.

```
POSITION Next(
  POSITION pos
  );
```

Parameters

pos
Current position in the list.

Return Values

Returns a position cursor.

Remarks

This member function returns NULL when going past the beginning of the list. Calling the **CBaseList::Next** member function with a null value is similar to calling the CBaseList::GetHeadPositionI member function.

Use the CBaseList::GetI member function to return the object at the returned position.

Previous Home Topic Contents Index Next

CBaseList::Prev

CBaseList Class

Retrieves the previous position in the list.

```
POSITION Prev(
  POSITION pos
  );
```

Parameters

pos
Current position in the list.

Return Values

Returns a position cursor.

Remarks

This member function returns NULL when going past the end of the list. Calling the **CBaseList::Prev** member function with a null value is similar to calling the CBaseList::GetTailPositionI member function.

Use the CBaseList::GetI member function to return the object at the returned position.

Previous Home Topic Contents Index Next

CBaseList::RemoveAll

CBaseList Class

Removes all nodes from the list.

void RemoveAll( );

Return Values

No return value.

Previous Home Topic Contents Index Next

CBaseList::RemoveHeadI

CBaseList Class

Removes the first node in the list.

void *RemoveHeadI( );

Return Values

Returns the pointer to the object that was removed.

Remarks

This member function deletes the pointer to its object from the list, but does not free the object itself.

If the list was already empty, this member function harmlessly returns NULL.

Previous Home Topic Contents Index Next

CBaseList::RemoveI

CBaseList Class

Removes the specified node from the list.

void *RemoveI(
POSITION *pos*
);

Parameters

pos
Position in the list of the node to remove.

Return Values

Returns the pointer to the object that was removed.

Remarks

This member function deletes the pointer to its object from the list, but does not free the object itself.

If the list was already empty, this member function harmlessly returns NULL.

Previous Home Topic Contents Index Next

CBaseList::RemoveTailI

CBaseList Class

Removes the last node in the list.

void *RemoveTailI( );

Return Values

Returns the pointer to the object that was removed.

Remarks

This member function deletes the pointer to its object from the list, but does not free the object.

If the list was already empty, this member function harmlessly returns NULL.

Previous Home Topic Contents Index Next

CBaseList::Reverse

CBaseList Class

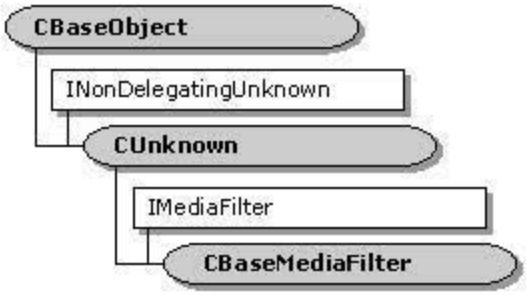
Reverses the order of the pointers to the objects in the list.

void Reverse( );

Return Values

No return value.

CBaseMediaFilter Class



This is an abstract base class that provides support for the IMediaFilter interface. The **CBaseMediaFilter** class handles State_Stopped, State_Paused, and State_Running state transitions. Typically, this class is used for plug-in distributors rather than filters with pins. Derive your filter classes from the CBaseFilter class (or base classes derived from this) instead of 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_clsid	Class identifier (CLSID) used for serialization using IPersist.
m_pClock	Pointer to a reference clock used for synchronization. The reference count of the clock object must be incremented using AddRef. Pass NULL if no reference clock is available.
m_State	Current state of the filter, which can be State_Stopped, State_Paused, or State_Running.
m_tStart	Offset from the stream time to the reference time.

Member Functions

Name	Description
CBaseMediaFilter	Constructs a CBaseMediaFilter object.
IsActive	Determines if the filter is currently active (running or paused) or stopped.

Overridable Member Functions

Name	Description
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 the new (paused) state.
Run	Instructs the filter to transition to the new (running) state.
SetSyncSource	Informs the filter of the reference clock with which it should synchronize activity.
Stop	Instructs the filter to transition to the new (stopped) state.

Implemented INonDelegatingUnknown Methods

Name	Description
NonDelegatingQueryInterface	Passes out references to interfaces supported by CBaseFilter. Override this to pass out pointers to interfaces supported in a derived filter class.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseMediaFilter::CBaseMediaFilter

CBaseMediaFilter Class

Constructs a CBaseMediaFilter object.

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

Parameters

pName
Name of the CBaseMediaFilter class.

pUnk
IUnknown interface of the delegating object.

pLock
Pointer to the object that maintains the lock.

clsid
Class identifier used to serialize this filter.

Return Values

No return value.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseMediaFilter::GetClassID

CBaseMediaFilter 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

Returns an HRESULT value.

Previous Home Topic Contents Index Next

CBaseMediaFilter::GetState

CBaseMediaFilter 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
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.

Filters should derive their filters from CBaseFilter and not from CBaseMediaFilter, so filters will not likely use this member function. Use CBaseFilter::GetState instead.

Previous Home Topic Contents Index Next

CBaseMediaFilter::GetSyncSource

CBaseMediaFilter 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.

Filters should derive their filters from CBaseFilter and not from CBaseMediaFilter, so filters will not likely use this member function. Use CBaseFilter::GetSyncSource instead.

Previous Home Topic Contents Index Next

CBaseMediaFilter::IsActive

CBaseMediaFilter 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 it is stopped.

Previous Home Topic Contents Index Next

CBaseMediaFilter::NonDelegatingQueryInterface

CBaseMediaFilter 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 IMediaFilter, IPersist, and IUnknown interfaces. Override this class to return other interfaces on the object in the derived class.

Previous Home Topic Contents Index Next

CBaseMediaFilter::Pause

CBaseMediaFilter Class

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

HRESULT Pause (void);

Return Values

Returns an HRESULT return value (S_OK by default).

Remarks

This member function implements the IMediaFilter::Pause method. It sets the value of m_State to State_Paused.

Note that filters should derive their filters from CBaseFilter and not from CBaseMediaFilter, so this member function will not likely be used by filters. Use CBaseFilter::Pause instead.

Previous Home Topic Contents Index Next

CBaseMediaFilter::Run

CBaseMediaFilter Class

Transitions the filter to State_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.

Remarks

If the filter is in State_Stopped state, the CBaseMediaFilter::Pause member function 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 CBasePin::Active member function, 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.

Filters should derive their filters from CBaseFilter and not from CBaseMediaFilter, so filters will not likely use this member function. Use CBaseFilter::Run instead.

Previous Home Topic Contents Index Next

CBaseMediaFilter::SetSyncSource

CBaseMediaFilter 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.

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.

Filters should derive their filters from CBaseFilter and not from CBaseMediaFilter, so filters will not likely use this member function. Use CBaseFilter::SetSyncSource instead.

Previous Home Topic Contents Index Next

CBaseMediaFilter::Stop

CBaseMediaFilter 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 sets the m_State member variable to State_Stopped.

Note that filters should derive their filters from CBaseFilter and not from CBaseMediaFilter, so this member function will not likely be used by filters. Use CBaseFilter::Stop instead.

Previous Home Topic Contents Index Next

CBaseMediaFilter::StreamTime

CBaseMediaFilter 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.

CBaseObject Class

CBaseObject

The **CBaseObject** class is an abstract base class that is the basis for all component objects. It maintains a process-wide count of active objects that can be queried from the DllCanUnloadNow entry point.

All Component Object Model (COM) objects are derived from the CUnknown class, which is derived from the **CBaseObject** class. Other objects can be derived from **CBaseObject** to assist in the detection of memory leaks, because **CBaseObject** maintains the count of created objects.

The constructor requires a character-string name that describes the object being created. This string can be displayed on the debugging screen to trace the creation of objects; the string will also be displayed upon deletion of the object. The string should be created in static storage rather than in local-function storage. The string can be enclosed by the NAME macro, which compiles to NULL in retail builds so that the static strings are optimized out during compilation.

```
/* Typical object creation method */
HRESULT CSomeClass::CreateMyObject(void)
{
   HRESULT hr = NOERROR;

   CMyObject *pObject = new CMyObject(NAME("My filter object"),NULL,&hr);
      if (FAILED(hr)) {
         return hr;
   }

   if (pObject == NULL) {
      return E_OUTOFMEMORY;
   }
   m_pObject = pObject;
   return NOERROR;
}

/* Incorrect object creation method */

HRESULT CSomeClass::ThisMayAccessViolate(void)
{
   HRESULT hr = NOERROR;

   TCHAR MyObjectName[] = TEXT("My GP faulting object");
   CMyObject *pObject = new CMyObject(MyObjectName,NULL,&hr);
}
```

Member Functions

Name	Description
CBaseObject	Constructs a CBaseObject object.
ObjectsActive	Retrieves the count of active objects.

Previous Home Topic Contents Index Next

CBaseObject::CBaseObject

CBaseObject Class

Constructs a CBaseObject object.

CBaseObject(
const TCHAR *pName*
);

Parameters

pName
Name assigned to the object for debugging purposes.

Return Values

No return value.

Remarks

The *pName* parameter should be allocated in static memory. This name appears on the debugging screen when the object is created and deleted.

Previous Home Topic Contents Index Next

CBaseObject::ObjectsActive

CBaseObject Class

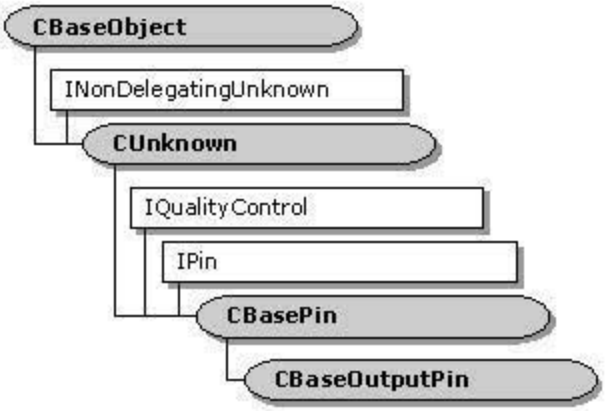
Retrieves the count of active objects.

static LONG ObjectsActive( );

Return Values

Returns the current number of active objects.

CBaseOutputPin Class



CBaseOutputPin is an abstract base class derived from the CBasePin class that provides support for the common memory transport. **CBaseOutputPin** connects only to an input pin that supplies an IMemInputPin interface (such as a pin class derived from the CBaseInputPin class), and provides methods for the filter to access that interface. Derive your output pins from this class for the easiest implementation.

An output pin must provide one or more media types when connected to an input pin. If the media type that returns an index size, for example, is not currently available, the output pin should return S_FALSE in the CBasePin::GetMediaType member function, and the base class will skip it.

Your output pin class methods (represented here with the class name CYourPin) should call **CBaseOutputPin**. For example, CYourPin::Active should call CBaseOutputPin::Active first, to see if it should proceed. CYourPin::Inactive should call CBaseOutputPin::Inactive first, to decommit the sample allocator and avoid deadlock problems with CBaseOutputPin::GetDeliveryBuffer.

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

All IQualityControl method implementations are inherited from the CBasePin class and are not overridden by this class.

Protected Data Members

Name	Description
m_pAllocator	Pointer to the IMemAllocator interface for this pin.
m_pInputPin	Pointer to the input pin to which this pin is connected.

Member Functions

Name	Description
CBaseOutputPin	Constructs a CBaseOutputPin object.

Overridable Member Functions

Name	Description
Active	Switches the pin to the active (running) mode.
BreakConnect	Releases the allocator and the IMemInputPin interface.
CheckConnect	Calls QueryInterface to retrieve an IMemInputPin interface.
CompleteConnect	Completes the connection.
DecideAllocator	Negotiates the allocator.
DecideBufferSize	Retrieves the number and size of buffers required for the transfer.
Deliver	Delivers an IMediaSample buffer to the connecting pin.
DeliverBeginFlush	Calls the IPin::BeginFlush method on the connected pin.
DeliverEndFlush	Calls IPin::EndFlush on the connected input pin to pass an end-flushing notification.
DeliverEndOfStream	Calls IPin::EndOfStream on the connected input pin to pass an end-of-stream notification.
DeliverNewSegment	Calls IPin::NewSegment on the connected input pin to pass a segment.
GetDeliveryBuffer	Returns an IMediaSample buffer suitable for passing across the connection.
Inactive	Switches the pin to the inactive (stopped) mode.
InitAllocator	Creates a default memory allocator. Override this to provide your own allocator or to provide no allocator.

Implemented IPin Methods

Name	Description
BeginFlush	Informs the pin to begin a flush operation. Implemented to return E_UNEXPECTED because it is an error to call this on an output pin.
EndFlush	Informs the pin to end a flush operation. Implemented to return E_UNEXPECTED because it is an error to call this on an output pin.
EndOfStream	Informs the pin that no additional data is expected until a new run command is issued. Implemented to return E_UNEXPECTED because it is an error to call this on an output pin.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseOutputPin::Active

CBaseOutputPin Class

Called by the CBaseFilter implementation when the state changes from stopped to either paused or running.

HRESULT Active(void);

Return Values

Returns VFW_E_NO_ALLOCATOR if there is no allocator.

Remarks

This member function calls CMemAllocator::Commit to commit memory required before becoming active.

Previous Home Topic Contents Index Next

CBaseOutputPin::BeginFlush

CBaseOutputPin Class

Informs the pin to begin a flush operation.

HRESULT BeginFlush(void);

Return Values

Returns E_UNEXPECTED.

Remarks

This member function implements the IPin::BeginFlush method. It returns E_UNEXPECTED because this should be called only on input pins.

Previous Home Topic Contents Index Next

CBaseOutputPin::BreakConnect

CBaseOutputPin Class

Releases IMemAllocator and IMemInputPin objects acquired by the pin.

HRESULT BreakConnect(void);

Return Values

Returns NOERROR by the default base class implementation.

Remarks

This member function releases the IMemAllocator and IPin interfaces used during the connection.

If you override this method, always call the base class **BreakConnect** or unexpected behavior will result, including reference count leaks.

Previous Home Topic Contents Index Next

CBaseOutputPin::CBaseOutputPin

CBaseOutputPin Class

Constructs a CBaseOutputPin object.

CBaseOutputPin(
TCHAR **pObjectName*,
CBaseFilter **pFilter*,
CCritSec **pLock*,
HRESULT * *phr*,
LPCWSTR *pName*
);

Parameters

pObjectName
Name of the object used in the CBaseOutputPin constructor for debugging purposes.
pFilter
Filter to which the pin will be attached.
pLock
Pointer to a CBaseOutputPin object for locking.
phr
Pointer to the general COM return value. This value is changed only if this function fails.
pName
Pin name.

Return Values

No return value.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseOutputPin::CheckConnect

CBaseOutputPin Class

Calls QueryInterface on the connected pin to retrieve an IMemInputPin interface.

HRESULT CheckConnect(
IPin **pPin*
);

Parameters

pPin
Pointer to the IPin interface on the connecting pin.

Return Values

Returns NOERROR if successful; otherwise, returns an HRESULT error value.

Previous Home Topic Contents Index Next

CBaseOutputPin::CompleteConnect

CBaseOutputPin Class

Completes a connection to another filter.

virtual HRESULT CompleteConnect(
IPin **pReceivePin*
);

Parameters

pReceivePin
Pointer to the connected (receiving) pin.

Return Values

Returns an HRESULT value. The default implementation returns NOERROR.

Remarks

This member function overrides the CBasePin::CompleteConnect member function and calls the CBaseOutputPin::DecideAllocator member function to finish completing the connection.

Previous Home Topic Contents Index Next

CBaseOutputPin::DecideAllocator

CBaseOutputPin Class

Negotiates the allocator to use.

```
virtual HRESULT DecideAllocator(
  IMemInputPin * pPin,
  IMemAllocator ** pAlloc
  );
```

Parameters

pPin
Pointer to the IPin interface of the connecting pin.
pAlloc
Pointer to the negotiated IMemAllocator interface.

Return Values

Returns NOERROR if successful; otherwise, returns an HRESULT value.

Remarks

This member function calls the CBaseOutputPin::DecideBufferSize member function, which is not implemented by this base class. Override DecideBufferSize to call IMemAllocator::SetProperties.

If the connected input pin fails a call to IMemInputPin::GetAllocator, this member function

constructs a CMemAllocator object and calls CBaseOutputPin::DecideBufferSize on that object. If the call to DecideBufferSize is successful, this member function notifies the input pin of the selected allocator. This function is called by the base class implementation of the IPin::Connect method, which is responsible for locking the object's critical section.

Override this member function if you want to use your own allocator. The input pin gets the first choice for the allocator, and the output pin agrees or forces it to use another allocator.

Previous Home Topic Contents Index Next

CBaseOutputPin::DecideBufferSize

CBaseOutputPin Class

Retrieves the number and size of buffers required for the transfer.

virtual HRESULT DecideBufferSize(
IMemAllocator * *pAlloc*,
ALLOCATOR_PROPERTIES * *ppropInputRequest*
) PURE;

Parameters

pAlloc
Allocator assigned to the transfer.
ppropInputRequest
Requested allocator properties for count, size, and alignment, as specified by the ALLOCATOR_PROPERTIES structure.

Return Values

Returns an HRESULT value.

Remarks

The CBaseOutputPin::DecideAllocator member function calls this member function. You must override this member function in your derived class and call IMemAllocator::SetProperties.

Previous Home Topic Contents Index Next

CBaseOutputPin::Deliver

CBaseOutputPin Class

Delivers the IMediaSample buffer to the connected pin.

virtual HRESULT Deliver(
IMediaSample *pSample*
);

Parameters

pSample
Buffer to deliver.

Return Values

Returns VFW_E_NOT_CONNECTED if no input pin is found; otherwise, returns an HRESULT value.

Remarks

This member function delivers this buffer to the connected input pin by calling its IMemInputPin::Receive method.

Previous Home Topic Contents Index Next

CBaseOutputPin::DeliverBeginFlush

CBaseOutputPin Class

Calls the IPin::BeginFlush method on the connected input pin.

virtual HRESULT DeliverBeginFlush(void);

Return Values

Returns VFW_E_NOT_CONNECTED if no input pin is found; otherwise, returns the value that is returned by the IPin::BeginFlush method.

Remarks

This member function delivers the BeginFlush notification downstream.

Previous Home Topic Contents Index Next

CBaseOutputPin::DeliverEndFlush

CBaseOutputPin Class

Calls the IPin::EndFlush method on the connected input pin.

virtual HRESULT DeliverEndFlush(void);

Return Values

Returns VFW_E_NOT_CONNECTED if no input pin is found; otherwise, returns the value that is returned by IPin::EndFlush.

Remarks

This member function delivers the EndFlush notification downstream.

Previous Home Topic Contents Index Next

CBaseOutputPin::DeliverEndOfStream

CBaseOutputPin Class

Calls the IPin::EndOfStream method on the connected input pin.

virtual HRESULT DeliverEndOfStream(void);

Return Values

Returns VFW_E_NOT_CONNECTED if no input pin is found; otherwise, returns the value returned by the IPin::EndOfStream call to the connected pin.

Remarks

This member function delivers the end-of-stream notification downstream by calling the IPin::EndOfStream method on the connected pin.

Previous Home Topic Contents Index Next

CBaseOutputPin::DeliverNewSegment

CBaseOutputPin Class

Calls the IPin::NewSegment method on the connected input pin.

```
virtual HRESULT DeliverNewSegment(
  REFERENCE_TIME tStart,
  REFERENCE_TIME tStop,
  double dRate
  );
```

Parameters

tStart
Start time of the segment.
tStop
Stop time of the segment.
dRate
Rate of the segment.

Return Values

Returns an HRESULT value.

Remarks

You will need to override this member function in your derived output pin class if your filter queues any data in the output pin.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseOutputPin::EndFlush

IPin Interface

Informs the pin to end a flush operation.

HRESULT EndFlush(void);

Return Values

Returns E_UNEXPECTED.

Remarks

This member function implements the IPin::EndFlush method. It returns E_UNEXPECTED because this should be called only on input pins.

Previous Home Topic Contents Index Next

CBaseOutputPin::EndOfStream

IPin Interface

Informs the input pin that no additional data is expected until a new run command is issued.

HRESULT EndOfStream(void);

Return Values

Returns E_UNEXPECTED.

Remarks

This member function implements the IPin::EndOfStream method but isn't expected to be called on an output pin.

Previous Home Topic Contents Index Next

CBaseOutputPin::GetDeliveryBuffer

CBaseOutputPin Class

Retrieves an IMediaSample buffer suitable for passing across the connection.

virtual HRESULT GetDeliveryBuffer(
IMediaSample ** *ppSample*,
REFERENCE_TIME * *pStartTime*,
REFERENCE_TIME * *pEndTime*,
DWORD *dwFlags*
);

Parameters

ppSample
IMediaSample buffer to be provided.
pStartTime
Start time of the media sample (optional and can be NULL).
pEndTime
Stop time of the media sample (optional and can be NULL).
dwFlags
The following flags are supported.

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

Return Values

Returns E_NOINTERFACE if an allocator is not found; otherwise, returns the value returned from calling the IMemAllocator::GetBuffer method.

Remarks

The pin object must lock itself before calling this member function; otherwise, the filter graph could disconnect this pin from the input pin midway through the process. If the filter has no worker threads, the lock is best applied on the IMemInputPin::Receive call; otherwise, it should be done when the worker thread is ready to deliver the sample.

This call can block; therefore, to avoid deadlocking with an IMediaFilter::Stop command, a two-tier locking scheme (such as that implemented in CTransformFilter) is required. Only the second-level lock is acquired here. The IBaseFilter base class implementation of **IMediaFilter::Stop** first gets the first-level lock and then calls IMemAllocator::Decommit on the allocator. This has the effect of making **GetDeliveryBuffer** return with a failure code. The Stop member function then gets the second-level lock and completes the command by calling Inactive for this pin.

No lock is needed when calling **CBaseOutputPin::GetDeliveryBuffer** when passing on samples using a worker thread. In this case, the CBaseFilter::Stop base class implementation acquires its filter-level lock and just calls IMemAllocator::Decommit on the allocator, at which point the worker thread is freed up to listen for a command to stop.

You must release the sample yourself after this function. If the connected input pin needs to hold on to the sample beyond the function, it will add the reference for the sample itself through IUnknown::AddRef. You must release this one and call

CBaseOutputPin::GetDeliveryBuffer for the next. (You cannot reuse it directly.)

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseOutputPin::Inactive

CBaseOutputPin Class

Called by the CBaseFilter implementation when the state changes from either paused or running to stopped.

HRESULT Inactive(void);

Return Values

Returns VFW_E_NO_ALLOCATOR if there is no allocator; otherwise, returns the value from calling the IMemAllocator::Decommit method.

Remarks

This member function calls IMemAllocator::Decommit to decommit memory before becoming inactive.

Previous Home Topic Contents Index Next

CBaseOutputPin::InitAllocator

CBaseOutputPin Class

Creates a default memory allocator. Override this to provide your own allocator or to provide no allocator.

virtual HRESULT InitAllocator(
IMemAllocator ***ppAlloc*
);

Parameters

ppAlloc
Returned memory allocator.

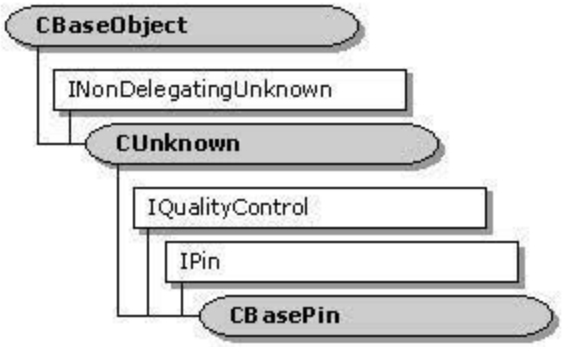
Return Values

Returns an HRESULT value.

Remarks

The allocator should be released after use. This is typically handled in the CBaseOutputPin::BreakConnect member function.

CBasePin Class



CBasePin is an abstract base class from which all pins are derived. **CBasePin** supports the IPin interface. You can derive from this if your pin does not supply or use the IMemInputPin interface; otherwise, derive from the CBaseInputPin class or the CBaseOutputPin class.

The connection process is crucial to the success of creating filter graphs. The filter graph finds two filters (and subsequently two pins) to connect. It calls the IPin::Connect method on the output pin (it can also call Connect on the input pin at the same time). The output pin then calls the virtual pin member function CBasePin::CheckConnect. Derived classes should override this member function to use QueryInterface to return any interfaces required. The base class implementation of CheckConnect queries the IMemInputPin interface to establish the default transport protocol.

After calling CheckConnect, the output pin calls CBasePin::AgreeMediaType; this is a worker member function not intended for overriding in derived classes. This gets the input pin's enumerator and calls CBasePin::TryMediaTypes with it. TryMediaTypes is another base pin worker member function that is not intended for derivation. It cycles through each media type provided by an enumerator to determine if a connection can be made with that type.

If that process fails, AgreeMediaType retrieves the output pin's media type enumerator and calls CBasePin::GetMediaType, which cycles through the media types to agree on a connection type. If there is agreement, a media type with the input and output pins becomes the type used in the connection.

If no media type can be agreed on, the connection between the pins cannot be made. The base pin calls CBasePin::SetMediaType to broadcast the format. The m_mt base pin variable is set during this process.

The IPin interface provides a method called QueryAccept. This method allows a connected filter to query whether the pin will accept a specified media type. The method is asynchronous so that a filter can call it at any time—even when another filter is calling it. For this reason, its implementation in any override of the base class should not lock the filter. The base class implementation of IPin::QueryAccept calls the overridden CBasePin::CheckMediaType member function on the derived pin 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_bRunTimeError	Run-time error generated.
m_Connected	Pin to which this pin is connected.
m_dir	Direction of this pin.
m_dRate	Rate from the CBasePin::NewSegment call.
m_mt	Media type that this pin is using. This is established during the connection process.
m_pFilter	Filter that created the pin.
m_pLock	Object used for locking.
m_pQSink	Target for quality messages.
m_pName	Name of the pin.
m_tStart	Start time from the CBasePin::NewSegment call.
m_tStop	Stop time from the CBasePin::NewSegment call.
m_TypeVersion	Current media type version (see CBasePin::GetMediaTypeVersion).

Member Functions

Name	Description
AttemptConnection	Attempts to make a connection to another pin using a specified media type.
CBasePin	Constructs a CBasePin object.
CurrentRate	Returns the segment rate set by the CBasePin::NewSegment member function.
CurrentStartTime	Returns the segment start time set by the CBasePin::NewSegment member function.
CurrentStopTime	Returns the segment stop time set by the CBasePin::NewSegment member function.
DisplayPinInfo	Displays pin information on the debugging monitor.
DisplayTypeInfo	Displays media type information on the debugging monitor.
GetConnected	Returns the pin that is connected to this pin.
IncrementTypeVersion	Adds 1 to the current media type version.
IsConnected	Determines whether the pin is connected.
IsStopped	Determines whether the filter owning this pin is in the State_Stopped state.
Name	Returns the m_pName name of the pin.

Overridable Member Functions

Name	Description
Active	Switches the pin to the active (running) mode.
AgreeMediaType	Agrees on the media type to be used by the pin.
BreakConnect	Adds custom code when the connection quits. This is also called when a stage in the connection process fails, so this member function should also clean up partial connection states.
CheckConnect	Adds custom code when the connection is being made. This is called at the start of the connection process.
CheckMediaType	Checks if the pin can support a specific media type.
CompleteConnect	Completes the connection.
GetMediaType	Returns the media type used by the pin.
GetMediaTypeVersion	Returns the version of the pins that were created dynamically.
Inactive	Switches the pin to the inactive (stopped) mode.
SetMediaType	Sets the m_mt data member to the established media type.
TryMediaTypes	Tries to find an acceptable media type for a connection from the list returned by a media type enumerator.

Implemented IPin Methods

Name	Description
Connect	Initiates a connection to another pin.
ConnectedTo	Returns a pointer to the connecting pin.
ConnectionMediaType	Returns the media type of this pin's connection.
Disconnect	Breaks a connection.
EndOfStream	Informs the input pin that no additional data is expected until a new run command is issued. (returns S_FALSE by default).
EnumMediaTypes	Returns an enumerator for this pin's preferred media types.
NewSegment	Specifies that samples following this call are grouped as a segment with a given start time, stop time, and rate.
QueryAccept	Determines whether this pin accepts the media type.
QueryDirection	Retrieves the pin direction of the pin.
QueryId	Retrieves an identifier for the pin.
QueryInternalConnections	Returns an array of the pins to which this pin connects internally.
QueryPinInfo	Retrieves information about the pin itself (the name, owning filter, or direction).
ReceiveConnection	Called by a connecting pin to make a connection to this pin. Usually this does not need to be overridden, because the default implementation calls CBasePin::CheckConnect, CBasePin::CheckMediaType, and CBasePin::BreakConnect.
Run	Notifies the pin that the filter has changed state from paused to running.

Implemented IQualityControl Methods

Name	Description
Notify	Notifies the recipient that a quality change is requested.
SetSink	Sets the IQualityControl object that will receive quality messages.

Implemented INonDelegatingUnknown Methods

Name	Description
NonDelegatingAddRef	Increments the owning filter's reference count.
NonDelegatingQueryInterface	Retrieves CBasePin interfaces. Override this member function to pass out pointers to any interfaces added by the derived pin class.
NonDelegatingRelease	Decrements the owning filter's reference count.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBasePin::Active

CBasePin Class

Called by the CBaseFilter implementation when the state changes from stopped to either paused or running.

virtual HRESULT Active(void);

Return Values

Returns an HRESULT value. The default implementation returns NOERROR.

Remarks

Any class that requires notification of a change of state should override this member function. This is called when the filter owning the pin exits the State_Stopped state.

Note that the filter graph manager's internal state variable is not updated until after this member function returns, so testing the filter graph manager's state (directly or indirectly) from within this member function should be avoided.

Previous Home Topic Contents Index Next

CBasePin::AgreeMediaType

CBasePin Class

Searches for a media type for the pin connection.

virtual HRESULT AgreeMediaType(
IPin **pReceivePin*,
const CMediaType **pmt*
);

Parameters

pReceivePin
Pointer to the receiving pin.
pmt
Pointer to a media type object to be returned.

Return Values

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

Value	Meaning
NOERROR	A media type was found.
VFW_E_NO_ACCEPTABLE_TYPES	No agreement on a media type was reached.

Remarks

This member function is called during the connection process. It calls CBasePin::TryMediaTypes on both the owning pin and the pin connected to the owning pin; it enumerates the preferred data types on the pin. If one is found, TryMediaTypes tries the media type with the pin in a call to the CBasePin::ReceiveConnection member function. If this pin proposes a media type, its support is still verified by calling CBasePin::CheckMediaType. The enumerator can list all the media types, even if some of them are not currently available.

This member function is protected.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBasePin::AttemptConnection

CBasePin Class

Attempts to make a connection to another pin using a specified media type.

virtual HRESULT AttemptConnection(
IPin * *pReceivePin*,

const CMediaType **pmt*
);

Parameters

pReceivePin
Pointer to the receiving pin.
pmt
Pointer to a media type object containing the preferred media type for the connection.

Return Values

Returns an HRESULT value.

Remarks

This member function is used to attempt to connect with a given media type. Its main purpose is to call the IPin::ReceiveConnection method of the pin passed in the *pReceivePin* parameter. This member function is protected.

Previous Home Topic Contents Index Next

CBasePin::BreakConnect

CBasePin Class

Called when a connection is broken to allow for customization (intended for overriding).

virtual HRESULT BreakConnect( );

Return Values

Returns an HRESULT value. The default implementation returns NOERROR.

Remarks

This member function is called when a connection to the pin cannot be made or when CBasePin::Disconnect is called. In this case, it is necessary to undo anything performed during the connection process. You can override this member function to release any references to interfaces that were made during the connection.

Previous Home Topic Contents Index Next

CBasePin::CBasePin

CBasePin Class

Constructs a CBasePin object.

CBasePin(
TCHAR **pObjectName*,
CBaseFilter **pFilter*,
CCritSec **pLock*,
HRESULT **phr*,
LPCWSTR *pName*,
PIN_DIRECTION *dir*
);

Parameters

pObjectName
Description of the object.
pFilter
Owning filter that knows about pins.
pLock
Object that implements the lock.
phr
Pointer to a general COM return value. This value is changed only if this function fails.
pName
Pin name.
dir
Either PINDIR_INPUT or PINDIR_OUTPUT.

Return Values

No return value.

Remarks

This is a standard class constructor.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBasePin::CheckConnect

CBasePin Class

Allows for customization when the connection is first made (intended for overriding, if required).

virtual HRESULT CheckConnect(
IPin * *pPin*
);

Parameters

pPin
Pointer to the connecting pin.

Return Values

Returns one of the following arguments by default; if overridden, should return standard HRESULT values, including the following values.

Value	Meaning
E_INVALIDARG	Pin directions do not match between pins.
NOERROR	Connection verified successfully.

Remarks

This member function is called during a call to the IPin::Connect method to provide a virtual method that can do any specific check required for a connection, such as calling CBasePin::NonDelegatingQueryInterface. This base class method determines if the pin directions match.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBasePin::CheckMediaType

CBasePin Class

Determines if the pin can support a specific media type.

virtual HRESULT CheckMediaType(
const CMediaType * *pmt*
)
PURE;

Parameters

pmt
Pointer to a media type object containing the proposed media type.

Return Values

The overriding member function should return S_OK if the proposed media type is accepted; otherwise, it should return an HRESULT failure value, such as S_FALSE.

Remarks

This member function is typically called before calling the CBasePin::SetMediaType member function. It is also called from several other member functions, including CBasePin::ReceiveConnection and CBasePin::QueryAccept.

Previous Home Topic Contents Index Next

CBasePin::CompleteConnect

CBasePin Class

Completes a connection to another filter (intended for overriding).

```
virtual HRESULT CompleteConnect(
  IPin *pReceivePin
  );
```

Parameters

pReceivePin
Pointer to the connected (receiving) pin.

Return Values

Returns an HRESULT value. The default implementation returns NOERROR.

Remarks

Override this member function to check for required connection interfaces on the *pReceivePin* parameter or its filter. Failing this member function fails the connection and disconnects the other pin. The CBaseOutputPin class overrides this member function to establish a local memory transport.

CBasePin::Connect

CBasePin Class

Initiates a connection from this pin to the other pin.

HRESULT Connect(
IPin * *pReceivePin*,
const AM_MEDIA_TYPE *pmt
);

Parameters

pReceivePin
Input pin to connect to.
pmt
Optional media type parameter.

Return Values

Returns one of the following arguments by default; if overridden, should return standard HRESULT values.

Value	Meaning
VFW_E_ALREADY_CONNECTED	This output pin is already connected to another pin.
VFW_E_NOT_STOPPED	The filter graph is not in a stopped state and connection can't be performed.
Other error value	Returned from CBasePin::AgreeMediaType or CBasePin::CheckConnect or overridden versions of these member functions.

Remarks

This member function implements the IPin::Connect method. **IPin::Connect** is implemented on the output pin and calls the IPin::ReceiveConnection method for the connected input pin (implemented in the base classes as CBasePin::ReceiveConnection). This member function calls the virtual CBasePin::CheckConnect member function, which can be overridden to verify that the connection is possible. **CBasePin::CheckConnect** then calls CBasePin::AgreeMediaType to negotiate a common media type with the connected pin.

CBasePin::AgreeMediaType calls CBasePin::TryMediaTypes twice; once for this pin's media type enumerator and once for the receiving pin's media type enumerator. For each media type found, CBasePin::AttemptConnection is called, which in turn calls the receiving pin's IPin::ReceiveConnection method, and finally CBasePin::CompleteConnect if successful.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBasePin::ConnectedTo

CBasePin Class

Retrieves a pointer to the connected pin, if there is one.

HRESULT ConnectedTo(
IPin ** *ppPin*
);

Parameters

ppPin
IPin interface of the other pin (if any) to which this pin is connected.

Return Values

The base class returns S_OK if connected; otherwise, returns VFW_E_NOT_CONNECTED.

Remarks

This member function implements the IPin::ConnectedTo method. It adds a reference to the connected IPin interface by calling the IUnknown::AddRef method, because each copy of an interface pointer has its reference incremented. The calling application is responsible for calling IUnknown::Release on this interface when done with it.

Previous Home Topic Contents Index Next

CBasePin::ConnectionMediaType

CBasePin Class

Retrieves the media type associated with the current connection of the pin.

HRESULT ConnectionMediaType(
AM_MEDIA_TYPE *pmt*

);

Parameters

pmt

Pointer to an AM_MEDIA_TYPE structure. If the pin isn't connected, this structure is initialized to zero. Otherwise, the media type is returned in this parameter.

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IPin::ConnectionMediaType method. It returns a copy of the AM_MEDIA_TYPE structure that was negotiated for the pin connection when the pin was connected.

This method fails if the pin is unconnected. The task allocator allocates the media type's format block. Use the task allocator to free the format block, for example by calling the Microsoft Win32 CoTaskMemFree function.

Previous Home Topic Contents Index Next

CBasePin::CurrentRate

CBasePin Class

Retrieves the segment rate set by the CBasePin::NewSegment member function.

double CurrentRate( );

Return Values

Returns the value of m_dRate.

Previous Home Topic Contents Index Next

CBasePin::CurrentStartTime

CBasePin Class

Retrieves the segment start time set by the CBasePin::NewSegment member function.

REFERENCE_TIME CurrentStartTime( );

Return Values

Returns the value of m_tStart.

Previous Home Topic Contents Index Next

CBasePin::CurrentStopTime

CBasePin Class

Retrieves the segment stop time set by the CBasePin::NewSegment member function.

REFERENCE_TIME CurrentStopTime( );

Return Values

Returns the value of m_tStop.

Previous Home Topic Contents Index Next

CBasePin::Disconnect

CBasePin Class

Breaks a connection.

HRESULT Disconnect(void);

Return Values

Returns NOERROR if there is no connection.

Remarks

This member function implements the IPin::Disconnect method. It calls the CBasePin::BreakConnect member function and releases the IPin interface of the connected pin (held by m_Connected). There are no parameters because there is only one possible connection on this pin.

Previous Home Topic Contents Index Next

CBasePin::DisplayPinInfo

CBasePin Class

Displays pin information during debugging.

void DisplayPinInfo(
IPin *pReceivePin*
);

Parameters

pReceivePin
Pointer to the receiving pin.

Return Values

Returns an HRESULT value.

Previous Home Topic Contents Index Next

CBasePin::DisplayTypeInfo

CBasePin Class

Displays media type information during debugging.

void DisplayTypeInfo(
IPin *pPin*,

const CMediaType **pmt*
);

Parameters

pPin
Pointer to the pin's IPin interface.
pmt
Pointer to the media type object.

Return Values

No return value.

Remarks

This member function displays the major and minor media types of the specified media type object.

Previous Home Topic Contents Index Next

CBasePin::EndOfStream

IPin Interface

Informs the input pin that no additional data is expected until a new run command is issued.

HRESULT EndOfStream(void);

Return Values

Returns S_FALSE.

Remarks

This member function implements the IPin::EndOfStream method. This is intended for input pins only.

Previous Home Topic Contents Index Next

CBasePin::EnumMediaTypes

CBasePin Class

Provides an enumerator for this pin's preferred media types.

HRESULT EnumMediaTypes(
IEnumMediaTypes ** *ppEnum*
);

Parameters

ppEnum
Pointer to an enumerator for the media types.

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IPin::EnumMediaTypes method. It returns an enumerator object implemented by the CEnumMediaTypes class and obtains the IEnumMediaTypes interface, which adds a reference count to this enumerator. If an application receives an enumerator, the application must release this when done with it by calling IUnknown::Release on the enumerator.

‹Previous Home Topic Contents Index Next›

CBasePin::GetConnected

CBasePin Class

Retrieves the pin that is connected to this pin.

IPin * GetConnected( );

Return Values

Returns a pointer to an IPin interface.

Remarks

The caller should call the CBasePin::IsConnected member function before calling

CBasePin::GetConnected.

Previous Home Topic Contents Index Next

CBasePin::GetMediaType

CBasePin Class

Retrieves the current type version, which is used by enumerators.

virtual HRESULT GetMediaType(
int *iPosition*,
const CMediaType **pMediaType*
);

Parameters

iPosition
Position in the media type list.
pMediaType
Returned pointer to the media type at this position.

Return Values

Returns E_UNEXPECTED by default implementation; the overriding member function should return one of the following values, or an HRESULT error value if the value could not be set.

Value	Meaning
S_FALSE	Media type exists but is not currently usable.
S_OK	Media type was set.
VFW_S_NO_MORE_ITEMS	End of the list of media types has been reached.

Remarks

This is a virtual member function that returns a media type corresponding to the position in the list specified by the *iPosition* parameter. This base class simply returns an error because no media types are supported by default. Derived classes should override this member function.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBasePin::GetMediaTypeVersion

CBasePin Class

Retrieves the current type version, which is used by enumerators.

virtual LONG GetMediaTypeVersion( );

Return Values

Returns the value of m_TypeVersion by default. To return new media types, override this member function.

Remarks

This is a virtual member function that returns the current media type version. The base class initializes the media type enumerators to 1. A derived class can change the list of available media types. Each time it does, it should increment the version in the overriding member function. The media type enumerators call this member function when they are called to determine if they are out of date.

Previous Home Topic Contents Index Next

CBasePin::Inactive

CBasePin Class

Switches the pin to an inactive state.

virtual HRESULT Inactive(void);

Return Values

Returns NOERROR for a base class implementation. The overriding member function returns a standard HRESULT value and should not fail if the pin is already set as inactive.

Remarks

This member function is called by the IMediaFilter implementation when the state changes to inactive. This member function should be overridden to decommit allocators and free any hardware resources that were obtained in the CBasePin::Active call. The default implementation of the base class member function does nothing.

Note that the filter graph manager's internal state variable is not updated until after this member function returns, so testing the filter graph manager's state (directly or indirectly) from within this member function should be avoided.

Previous Home Topic Contents Index Next

CBasePin::IncrementTypeVersion

CBasePin Class

Adds 1 to the current media type version.

void IncrementTypeVersion(void);

Return Values

No return value.

Remarks

The media type version is used to ensure that the filter has not changed the media type. If it changes the media type, the filter should call this member function.

Previous Home Topic Contents Index Next

CBasePin::IsConnected

CBasePin Class

Determines if the pin is connected to another pin.

BOOL IsConnected(void);

Return Values

Returns TRUE if the pin is connected; otherwise, returns FALSE.

Remarks

This member function checks the value of the m_Connected protected data member.

Previous Home Topic Contents Index Next

CBasePin::IsStopped

CBasePin Class

Determines if the filter is stopped.

BOOL IsStopped( );

Return Values

Returns TRUE if the filter is stopped; otherwise, returns FALSE.

Remarks

Note that this member function must not be used in the constructor of the pin, because the filter that is passed is often not initialized properly at that time (due to the convention of using a **this** pointer during the construction of data members).

Previous Home Topic Contents Index Next

CBasePin::Name

CBasePin Class

Retrieves the name of the pin.

LPWSTR Name( );

Return Values

Returns the value of the m_pName data member.

Previous Home Topic Contents Index Next

CBasePin::NewSegment

CBasePin Class

Specifies that samples following this call are grouped as a segment with a given start time, stop time, and rate.

HRESULT NewSegment(
REFERENCE_TIME *tStart*,
REFERENCE_TIME *tStop*,
double *dRate*
);

Parameters

tStart
Start time of the segment.
tStop
Stop time of the segment.
dRate
Rate of the segment.

Return Values

Returns an HRESULT value (S_OK by default).

Remarks

This member function implements the IPin::NewSegment method. The default implementation sets the m_tStart, m_tStop, and m_dRate data members to the values passed in as parameters. Overriding member functions should pass this notification downstream.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBasePin::NonDelegatingAddRef

CBasePin Class

Increments the reference count for an interface.

ULONG NonDelegatingAddRef( );

Return Values

Returns the reference count of the object.

Remarks

This member function implements the INonDelegatingUnknown::NonDelegatingAddRef method. It increments the reference count of the owning filter.

Previous Home Topic Contents Index Next

CBasePin::NonDelegatingQueryInterface

CBasePin 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 IPin, IQualityControl, and IUnknown interfaces. Override this class to return other interfaces on the object in the derived class.

CBasePin::NonDelegatingRelease

CBasePin Class

Decrements the reference count for an interface.

ULONG NonDelegatingRelease( );

Return Values

Returns the reference count.

Remarks

This member function implements the INonDelegatingUnknown::NonDelegatingRelease method. It releases a reference to the owning filter.

CBasePin::Notify

CBasePin Class

Notifies the recipient that a quality change is requested.

HRESULT Notify(
IBaseFilter * *pSelf*,
Quality *q*
);

Parameters

pSelf
Pointer to the filter that is sending the quality notification.

q
Quality notification structure.

Return Values

The default base class implementation returns E_FAIL.

Remarks

This member function must be overridden to accept notifications. It is typically overridden to implement this method on the output pin because quality-control messages are passed upstream. The CTransformOutputPin::Notify member function is one example of how this member function is overridden to pass quality-control messages to the next filter upstream.

Previous Home Topic Contents Index Next

CBasePin::QueryAccept

CBasePin Class

Determines whether the pin accepts the format type.

HRESULT QueryAccept(
const AM_MEDIA_TYPE* *pmt*
);

Parameters

pmt
Pointer to a proposed media type.

Return Values

Returns S_TRUE if the format is accepted; otherwise, returns S_FALSE.

Remarks

This member function implements the IPin::QueryAccept method. It simply calls the pure virtual CBasePin::CheckMediaType member function, which the derived class must implement, and maps any returned codes from CheckMediaType other than S_OK to S_FALSE.

Previous Home Topic Contents Index Next

CBasePin::QueryDirection

CBasePin Class

Retrieves the direction of the pin.

HRESULT QueryDirection(
PIN_DIRECTION* *pPinDir*
);

Parameters

pPinDir
Pointer to a PIN_DIRECTION structure to be filled in with the direction.

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IPin::QueryDirection method. *pPinDir* will contain PINDIR_INPUT or PINDIR_OUTPUT. The same information is available through the CBasePin::QueryPinInfo member function, but this member function is more efficient.

Previous Home Topic Contents Index Next

CBasePin::QueryId

CBasePin Class

Retrieves an identifier for the pin.

HRESULT QueryId(
LPWSTR * *Id*
);

Parameters

Id
Pin identifier.

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IPin::QueryId method. By default, this member function uses the pin name in the CBasePin::m_pName data member, so implementing this member function in your derived filter class is not normally required.

Previous Home Topic Contents Index Next

CBasePin::QueryInternalConnections

CBasePin Class

Provides an array of pointers to IPin objects. These are the pins to which this pin internally connects.

HRESULT QueryInternalConnections(
IPin ** *apPin*,
ULONG * *nPin*
);

Parameters

apPin
Array of IPin pointers.
nPin
Upon input, indicates the number of channels; upon output, indicates the number of pins.

Return Values

Returns one of the following HRESULT values.

Value	Meaning
E_FAIL	Undetermined failure.
E_NOTIMPL	The filter graph manager interprets E_NOTIMPL as meaning all input pins connect to all output pins.
S_FALSE	Insufficient number of channels; returns no pins in *apPin*.

Remarks

This member function implements the IPin::QueryInternalConnections method but only to return E_NOTIMPL. Override this if you want to provide mapping between specific input and output pins.

The default implementation to return E_NOTIMPL implies that the caller can assume that all input pins feed all output pins. Overriding this member function allows a filter to specify when it is a renderer for some of its input pins and not for others.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBasePin::QueryPinInfo

CBasePin Class

Retrieves information about the pin.

HRESULT QueryPinInfo(
PIN_INFO * *pInfo*
);

Parameters

pInfo
Pointer to a PIN_INFO structure.

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IPin::QueryPinInfo method. By default, the member function fills in the PIN_INFO structure with the IBaseFilter interface of its owning filter, the pin name from m_pName, and the pin direction from m_dir.

The IBaseFilter 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

CBasePin::ReceiveConnection

CBasePin Class

Makes a connection to the calling output pin.

HRESULT ReceiveConnection(
IPin * *pConnector*,
AM_MEDIA_TYPE **pmt*
);

Parameters

pConnector
Connecting pin.
pmt
Media type of the samples to be streamed.

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IPin::ReceiveConnection method. It calls CheckConnect and, if successful, then calls CheckMediaType to verify if the media type is acceptable. If either of these calls fails, it calls BreakConnect and exits. To finish the connection process, it calls CompleteConnect, which is implemented in CBasePin to return NOERROR, but can be overridden in the derived class.

Previous Home Topic Contents Index Next

CBasePin::Run

CBasePin Class

Notifies the pin that the filter has changed state from paused to running.

HRESULT Run(
REFERENCE_TIME *tStart*
);

Parameters

tStart
Start time as passed to the filter's Run method.

Return Values

Returns an HRESULT value (NOERROR_OK by default).

Remarks

This member function can be overridden in the derived class to perform activities such as committing memory or obtaining resources. For an overriding implementation of this member function, see the CRenderedInputPin::Run member function.

Note that the filter graph manager's internal state variable is not updated until after this member function returns, so testing the filter graph manager's state (directly or indirectly) from within this member function should be avoided.

Previous Home Topic Contents Index Next

CBasePin::SetMediaType

CBasePin Class

Sets the m_mt data member to the established media type.

```
virtual HRESULT SetMediaType(
  const CMediaType * pmt
  );
```

Parameters

pmt
Pointer to a media type object that was previously agreed on.

Return Values

Returns NOERROR by default implementation. The overriding member functions return an HRESULT value.

Remarks

This member function is called to establish the format for a pin connection. The CBasePin::CheckMediaType member function will have been called to check the connection format and, if it did not return an error value, this virtual member function will be called. The default implementation sets the m_mt protected data member to the value passed to this member function. Override to inform the derived class when the media type is set.

Previous Home Topic Contents Index Next

CBasePin::SetSink

CBasePin Class

Sets the object containing the IQualityControl interface that will receive quality-control messages.

HRESULT SetSink(
IQualityControl **piqc*
);

Parameters

piqc
Pointer to the IQualityControl interface to which the notifications should be sent.

Return Values

Base class returns NOERROR by default. The overriding member function should return an HRESULT value.

Remarks

This member function implements the IQualityControl::SetSink method. The default implementation sets the m_pQSink data member to the *piqc* parameter passed in.

The IQualityControl::SetSink method tells a filter where to send quality-control messages it receives. When no sink has been explicitly set or if the last call to **CBasePin::SetSink** set the sink to NULL, the message should go upstream. The derived output pin class typically overrides CBasePin::Notify to enable this.

Previous Home Topic Contents Index Next

CBasePin::TryMediaTypes

CBasePin Class

Determines a media type for a pin connection.

virtual HRESULT TryMediaTypes(
IPin **pReceivePin*,
const CMediaType **pmt*,
IEnumMediaTypes **pEnum*
);

Parameters

pReceivePin
Pointer to the IPin interface of the receiving pin.
pmt
Pointer to a returned media type.
pEnum
Pointer to an IEnumMediaTypes enumerator interface.

Return Values

Returns an HRESULT value, which can include the following.

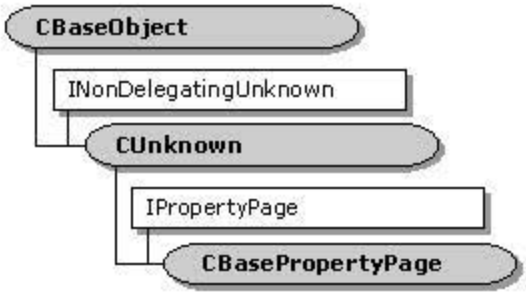
Value	Meaning
FAILED	Resetting of the enumerator failed.
NOERROR	Media type found.
VFW_E_NO_ACCEPTABLE_TYPES	No acceptable media types were found.

Remarks

Given an enumerator, this member function cycles through all the media types proposed by the enumerator. Each type is suggested to the derived pin class and, if acceptable, is tried with the connected pin in a call to the IPin::ReceiveConnection method. This means that if the owning pin proposes a media type, it is still checked to determine whether it is supported. This is deliberate so that, in simple cases, the enumerator can hold all the media types, even if some of them are not currently available.

This member function is protected.

CBasePropertyPage Class



Property pages can be implemented on filters to provide access to custom properties on the filter. This can be useful for developing and debugging the filter, because the Filter Graph Editor displays these property pages. Also, the Microsoft® ActiveMovie Control queries filters in its underlying filter graph for the property pages they support and exposes them to the user. A good example is the video renderer, which exposes quality management information (such as frame rate) through a property page.

This class provides the framework for a property page associated with a filter and implements the IPropertyPage interface. A property page is a Component Object Model (COM) object, which should be created with a resource ID for a dialog box that will be loaded when required. It should also be given a resource ID for a title string when created.

In addition to implementing the IPropertyPage interface methods, this class provides several virtual member functions that can be overridden and specialized by the derived class (they return NOERROR by default). These virtual member functions are called at specific events, such as when the property page is activated or deactivated, connected or disconnected, when the changes to properties are to be applied, or when messages to the dialog box are received.

A filter exposing custom property pages should also expose the same functionality to an application through a custom interface. Otherwise, an application has no way to control the filter without displaying the property page. For example, the video renderer supports the IQualProp interface to access the same quality management information. In fact, the renderer property page uses that interface to get the information for its property page. To make it easier for applications to access their custom interfaces, filters should also implement their custom interfaces in a plug-in distributor (PID), which is an object that is aggregated with the filter graph manager. Typically, the PID implements its associated filter's interface by simply passing calls through from the application to the filter interface.

Protected Data Members

Name	Description
m_bDirty	Flag to determine whether anything has changed.
m_DialogId	Resource identifier for the dialog box.
m_Dlg	Dialog box window handle for the property page.
m_hwnd	Window handle for the property page.
m_pPageSite	IPropertyPage interface pointer used to access the filter's property information.
m_TitleId	Resource identifier for the property page title.

Member Functions

Name	Description
CBasePropertyPage	Constructs a CBasePropertyPage object.

Overridable Member Functions

Name	Description
OnActivate	Called when the property page is activated.
OnApplyChanges	Called when the user applies changes to the property page.
OnConnect	Called when the property page is connected to the filter.
OnDeactivate	Called when the property page is dismissed.
OnDisconnect	Called when the property page is disconnected from the owning filter.
OnReceiveMessage	Called when a message is sent to the property page dialog box window.

Implemented INonDelegatingUnknown Methods

Name	Description
NonDelegatingAddRef	Default implementation increments the owning filter's reference count.
NonDelegatingQueryInterface	Called to retrieve CBasePropertyPage interfaces. Override this member function to pass out pointers to any interfaces added by the derived class.
NonDelegatingRelease	Default implementation decrements the owning filter's reference count.

Implemented IPropertyPage Methods

Name	Description
Activate	Creates a dialog box window for the property page.
Apply	Applies current property page values to the underlying object.
Deactivate	Destroys the window created with CBasePropertyPage::Activate.
GetPageInfo	Returns information about the property page.
Help	Invokes Help in response to user request.
IsPageDirty	Indicates whether the property page has changed since activated or since the most recent call to CBasePropertyPage::Apply.
Move	Positions and resizes the property page dialog box within the frame.
SetObjects	Provides the property page with an array of IUnknown pointers for objects associated with this property page.
SetPageSite	Initializes a property page and provides the page with a pointer to the IPropertyPageSite interface through which the property page communicates with the property frame.
Show	Makes the property page dialog box visible or invisible.

TranslateAccelerator Provides a pointer to an MSG structure that specifies a keystroke to process.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBasePropertyPage::Activate

CBasePropertyPage Class

Creates the property page dialog box.

HRESULT Activate(
HWND *hwndParent*,
LPCRECT *prect*,
BOOL *fModal*
);

Parameters

hwndParent
Handle to the parent window of the dialog box.
prect
Pointer to the RECT structure that contains the dialog box's screen position.
fModal
Value that specifies a modal dialog box if TRUE, or a modeless dialog box if FALSE.

Return Values

Returns E_OUTOFMEMORY if the dialog box creation fails, or E_UNEXPECTED if a property page already exists.

Remarks

This member function implements the COM IPropertyPage::Active method, which creates a dialog box for the property page (without a frame, caption, system menu, or controls) using *hwndParent* as the parent window and *prect* as the positioning rectangle.

The property page maintains the window handle created in this process, which it uses to destroy the dialog box within CBasePropertyPage::Deactivate.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBasePropertyPage::Apply

CBasePropertyPage Class

Applies current property page values to the underlying object.

HRESULT Apply(void);

Return Values

Returns E_UNEXPECTED if CBasePropertyPage::SetObjects has not been called or if the m_pPageSite data member has not been initialized with a pointer to the filter's property page.

Remarks

This member function implements the COM IPropertyPage::Apply method. The object to be changed is provided through a previous call to CBasePropertyPage::SetObjects. This should be the filter's IUnknown interface. Therefore, this member function should not fail because of nonexistent interfaces.

This member function sets the m_bDirty data member to FALSE and calls the virtual CBasePropertyPage::OnApplyChanges member function, which should be overridden in the derived class to apply the changes to the properties.

Previous Home Topic Contents Index Next

CBasePropertyPage::CBasePropertyPage

CBasePropertyPage Class

Constructs a CBasePropertyPage object.

CBasePropertyPage(
TCHAR *pName*,
LPUNKNOWN *pUnk*,
int *DialogId*,
int *TitleId*
);

Parameters

pName
Name of the property page object for debugging purposes.
pUnk
Pointer to the COM delegating object.
DialogId
Resource ID for the dialog box.
TitleId
Resource ID for the dialog box title.

Remarks

This constructor sets the CBasePropertyPage data members as follows:

- m_DialogId is set to *DialogId*.
- m_TitleId is set to *TitleId*.
- m_hwnd is set to NULL.
- m_Dlg is set to NULL.
- m_pPageSite is set to NULL.
- m_bDirty is set to FALSE.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBasePropertyPage::Deactivate

CBasePropertyPage Class

Destroys the window created with CBasePropertyPage::Activate.

HRESULT Deactivate(void);

Return Values

Returns E_UNEXPECTED if the data member m_hwnd does not contain a Window handle for the property page.

Remarks

This member function implements the COM IPropertyPage::Deactivate method. It calls the virtual CBasePropertyPage::OnDeactivate member function and then destroys the property page dialog box.

CBasePropertyPage::GetPageInfo

CBasePropertyPage Class

Returns information about the property page.

HRESULT GetPageInfo(
LPPROPPAGEINFO *pPageInfo*
);

Parameters

pPageInfo
Pointer to the caller-allocated PROPPAGEINFO structure in which the property page stores its page information. All allocations stored in this structure become the caller's responsibility.

Return Values

Returns E_OUTOFMEMORY if the function cannot allocate memory for the property page title.

Remarks

This member function implements the COM IPropertyPage::GetPageInfo method. It calls the GetDialogSize function to obtain the dialog box size and sets it to a default value in case this call fails.

CBasePropertyPage::Help

CBasePropertyPage Class

Invokes Help in response to user request.

HRESULT Help(

LPCWSTR *lpszHelpDir*
);

Parameters

lpszHelpDir
Pointer to the string under the HelpDir key in the property page's CLSID information in the registry. If HelpDir does not exist, this will be the path found in the InProcServer32 entry minus the server file name.

Return Values

Returns E_NOTIMPL by default.

Remarks

This member function implements the COM IPropertyPage::Help method, but only as a placeholder. The function does nothing but return E_NOTIMPL.

Calls to this member function must occur between calls to CBasePropertyPage::Activate and CBasePropertyPage::Deactivate.

If the page fails this member function (such as E_NOTIMPL), the frame will attempt to use the *pszHelpFile* and *dwHelpContext* fields of the PROPPAGEINFO structure obtained through CBasePropertyPage::GetPageInfo. Therefore, the derived class should either implement **CBasePropertyPage::Help** or return Help information through **CBasePropertyPage::GetPageInfo**.

Previous Home Topic Contents Index Next

CBasePropertyPage::IsPageDirty

CBasePropertyPage Class

Indicates whether the property page has changed since activated or since the most recent call to CBasePropertyPage::Apply.

HRESULT IsPageDirty(void);

Return Values

Returns S_OK if the value state of the property page is dirty, that is, it has changed and is different from the state of the objects. Returns S_FALSE if the value state of the page has not changed and is current with that of the objects.

Remarks

This member function implements the COM IPropertyPage::IsPageDirty method. It returns the value of the m_bDirty data member.

Previous Home Topic Contents Index Next

CBasePropertyPage::Move

CBasePropertyPage Class

Positions and resizes the property page dialog box within the frame.

HRESULT Move(
LPCRECT *prect*
);

Parameters

prect
Pointer to the RECT structure containing the positioning information for the property page dialog box.

Return Values

Returns E_UNEXPECTED if the m_hwnd data member does not contain a Window handle for the property page.

Remarks

This member function implements the COM IPropertyPage::Move method by calling the Microsoft® Win32® MoveWindow function. This member function is called from the CBasePropertyPage::Activate member function to position the property page dialog box.

Previous Home Topic Contents Index Next

CBasePropertyPage::NonDelegatingAddRef

CBasePropertyPage Class

Increments the reference count for an interface.

ULONG NonDelegatingAddRef( );

Return Values

Returns the object's reference count.

Remarks

This member function implements the INonDelegatingUnknown::NonDelegatingAddRef method. It increments the owning filter's reference count.

Previous Home Topic Contents Index Next

CBasePropertyPage::NonDelegatingQueryInterfa

CBasePropertyPage Class

Returns 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 IPropertyPage interface. It then calls the CUnknown::NonDelegatingQueryInterface base class member function. Override this class to return other interfaces on the object in the derived class.

CBasePropertyPage::NonDelegatingRelease

CBasePropertyPage Class

Decrements the reference count for an interface.

ULONG NonDelegatingRelease( );

Return Values

Returns the reference count.

Remarks

This member function implements the INonDelegatingUnknown::NonDelegatingRelease method. It releases a reference count to the owning filter.

CBasePropertyPage::OnActivate

CBasePropertyPage Class

Called when the property page is activated.

virtual HRESULT OnActivate(void);

Return Values

Returns NOERROR by default. The overriding member function should return a valid HRESULT value.

Remarks

This member function is called from the CBasePropertyPage::Activate member function to notify the derived class when the property page is displayed. Override this member function to

initialize values in the dialog box. This can be done by calling the Win32 SetDlgItemText function with data member values previously initialized when the property page was connected (in the overridden CBasePropertyPage::OnConnect member function).

For example, the Vidprop.cpp file in the sample video renderer, SampVid, does this as follows:

```
// Set the text fields in the property page

HRESULT CQualityProperties::OnActivate()
{
    SetEditFieldData();
    return NOERROR;
}


// Initialize the property page fields

void CQualityProperties::SetEditFieldData()
{
    ASSERT(m_pQualProp);
    TCHAR buffer[50];

    wsprintf(buffer,"%d", m_iDropped);
    SendDlgItemMessage(m_Dlg, IDD_QDROPPED, WM_SETTEXT, 0, (DWORD) (LPSTR) buffer);
    wsprintf(buffer,"%d", m_iDrawn);
    SendDlgItemMessage(m_Dlg, IDD_QDRAWN, WM_SETTEXT, 0, (DWORD) (LPSTR) buffer);
    wsprintf(buffer,"%d.%02d", m_iFrameRate/100, m_iFrameRate%100);
    SendDlgItemMessage(m_Dlg, IDD_QAVGFRM, WM_SETTEXT, 0, (DWORD) (LPSTR) buffer);
    wsprintf(buffer,"%d", m_iFrameJitter);
    SendDlgItemMessage(m_Dlg, IDD_QJITTER, WM_SETTEXT, 0, (DWORD) (LPSTR) buffer);
    wsprintf(buffer,"%d", m_iSyncAvg);
    SendDlgItemMessage(m_Dlg, IDD_QSYNCAVG, WM_SETTEXT, 0, (DWORD) (LPSTR) buffer);
    wsprintf(buffer,"%d", m_iSyncDev);
    SendDlgItemMessage(m_Dlg, IDD_QSYNCDEV, WM_SETTEXT, 0, (DWORD) (LPSTR) buffer);
}
```


Previous Home Topic Contents Index Next

CBasePropertyPage::OnApplyChanges

CBasePropertyPage Class

Called when the user applies changes to the property page.

virtual HRESULT OnApplyChanges(void);

Return Values

Returns NOERROR by default. The overriding member function should return a valid HRESULT value.

Remarks

Override this member function if your property page allows users to set property values. When this member function is called, process the changed properties. For example, set appropriate data members in the derived class to the new values, or call methods in the filter to set the properties. The overriding member function is responsible for calling CBasePropertyPage::IsPageDirty to set the m_bDirty data member to TRUE if the properties in the object do not reflect those in the property page when this member function exits.

Previous Home Topic Contents Index Next

CBasePropertyPage::OnConnect

CBasePropertyPage Class

Called when the property page is connected to the filter.

virtual HRESULT OnConnect(
IUnknown **pUnknown*
);

Parameters

pUnknown
IUnknown interface of the filter associated with the property page.

Return Values

Returns NOERROR by default. The overriding member function should return a valid HRESULT value.

Remarks

This member function is called from the CBasePropertyPage::SetObjects member function with the *ppUnk* parameter of that member function, which should be the filter's IUnknown interface. Override this member function to acquire property values to be sent to the property page dialog box later (in CBasePropertyPage::OnActivate).

The following excerpt from the sample video renderer (SampVid) Vidprop.cpp file illustrates the use of this member function.

```
// Give us the filter to communicate with

HRESULT CQualityProperties::OnConnect(IUnknown *pUnknown)
{
```

```
    ASSERT(m_pQualProp == NULL);

    // Ask the renderer for its IQualProp interface

    HRESULT hr = pUnknown->QueryInterface(IID_IQualProp,(void **)&m_pQualProp);
    if (FAILED(hr)) {
        return E_NOINTERFACE;
    }

    ASSERT(m_pQualProp);

    // Get quality data for the page

    m_pQualProp->get_FramesDroppedInRenderer(&m_iDropped);
    m_pQualProp->get_FramesDrawn(&m_iDrawn);
    m_pQualProp->get_AvgFrameRate(&m_iFrameRate);
    m_pQualProp->get_Jitter(&m_iFrameJitter);
    m_pQualProp->get_AvgSyncOffset(&m_iSyncAvg);
    m_pQualProp->get_DevSyncOffset(&m_iSyncDev);
    return NOERROR;
}
```


Previous Home Topic Contents Index Next

CBasePropertyPage::OnDeactivate

CBasePropertyPage Class

Called when the property page is dismissed.

virtual HRESULT OnDeactivate(void);

Return Values

Returns NOERROR by default. The overriding member function should return a valid HRESULT value.

Remarks

This member function is called from the CBasePropertyPage::Deactivate member function when the user closes the property page. Override this member function to handle any special requirements at that time.

Previous Home Topic Contents Index Next

CBasePropertyPage::OnDisconnect

CBasePropertyPage Class

Called when the property page is disconnected from the owning filter.

virtual HRESULT OnDisconnect(void);

Return Values

Returns NOERROR by default. The overriding member function should return a valid HRESULT value.

Remarks

This member function is called from the CBasePropertyPage::SetObjects member function when the property page is disconnected from the filter. Override this member function to handle any special requirements at that time, such as releasing reference counts on underlying property interfaces.

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

```
// Release any IQualProp interface we have

HRESULT CQualityProperties::OnDisconnect()
{
    // Release the interface

    if (m_pQualProp == NULL) {
        return E_UNEXPECTED;
    }

    m_pQualProp->Release();
    m_pQualProp = NULL;
    return NOERROR;
}
```


CBasePropertyPage::OnReceiveMessage

CBasePropertyPage Class

Called when a message is sent to the property page dialog box.

virtual BOOL OnReceiveMessage(
HWND *hwnd*,
UINT *uMsg*,
WPARAM *wParam*,
LPARAM *lParam*
);

Parameters

hwnd
Window procedure that received the message.
uMsg
Message.
wParam
Additional message information. This parameter's content depends on the value of the *uMsg* parameter.
lParam
Additional message information. This parameter's content depends on the value of the *uMsg* parameter.

Return Values

By default, returns the value returned by the Win32 DefWindowProc function.

Remarks

The default implementation of this member function calls DefWindowProc with the supplied parameters. Override this member function for special handling of messages.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBasePropertyPage::SetObjects

CBasePropertyPage Class

Provides the property page with an IUnknown pointer for objects associated with this property page.

HRESULT SetObjects(
ULONG *cObjects*,
LPUNKNOWN **ppUnk*
);

Parameters

cObjects
Number of IUnknown pointers in the array pointed to by *ppUnk*. This number should be 1 or 0. If it is 0, the property page must release any pointers previously passed to this method.

ppUnk
Pointer to a single IUnknown interface pointer identifying a unique object affected by the property sheet in which this (and possibly other) property page is displayed. The property page must cache this pointer by calling IUnknown::AddRef.

Return Values

Returns E_POINTER if *ppUnk* is null, E_UNEXPECTED if *cObjects* is greater than 1, and otherwise returns the value returned by the CBasePropertyPage::OnConnect or CBasePropertyPage::OnDisconnect member function that it calls.

Remarks

This member function implements the COM IPropertyPage::SetObjects method. This member function calls the virtual CBasePropertyPage::OnConnect member function when the *cObjects* value is 1, or the virtual CBasePropertyPage::OnDisconnect member function when the *cObjects* value is 0. Override these virtual member functions to acquire (by calling IUnknown::AddRef) or release (by calling IUnknown::Release) interfaces to which the property page applies.

Note that the caller must provide the property page with this object before calling CBasePropertyPage::Activate, and should call **CBasePropertyPage::SetObjects** with 0-v as the parameter when deactivating the page or when releasing the object entirely.

This member function allows only one object to be associated with the property page.

Previous Home Topic Contents Index Next

CBasePropertyPage::SetPageSite

CBasePropertyPage Class

Initializes a property page and provides the page with a pointer to the IPropertyPageSite interface through which the property page communicates with the property frame.

HRESULT SetPageSite(
LPPROPERTYPAGESITE *pPageSite*
);

Parameters

pPageSite
Pointer to the IPropertyPageSite interface of the page site that manages and provides services to this property page within the entire property sheet.

Return Values

Returns E_UNEXPECTED if if the m_pPageSite data member has not been initialized with a pointer to the filter's property page.

Remarks

This member function implements the COM IPropertyPage::SetPageSite method. When passed an IPropertyPageSite interface, it reference counts the interface and assigns it to m_pPageSite. When passed a null value, it releases the reference count on the **IPropertyPageSite** interface.

Previous Home Topic Contents Index Next

CBasePropertyPage::Show

CBasePropertyPage Class

Makes the property page dialog box visible or invisible.

HRESULT Show(
UINT *nCmdShow*
);

Parameters

nCmdShow
Command describing whether to become visible. Only SW_SHOWNORMAL, SW_SHOW, and SW_HIDE are accepted.

Return Values

Returns E_UNEXPECTED if the data member m_hwnd does not contain a Window handle for the property page. Returns E_INVALIDARG if the *nCmdShow* parameter is not equal to SW_SHOW or SW_SHOWNORMAL (show) or SW_HIDE (hide).

Remarks

If the page is made visible, the page should set the focus to itself, specifically to the first

property on the page. This member function implements the COM IPropertyPage::Show method. This is called just before exiting the CBasePropertyPage::Activate member function with the *nCmdShow* SHOW_NORMAL value.

Previous Home Topic Contents Index Next

CBasePropertyPage::TranslateAccelerator

CBasePropertyPage Class

Provides a pointer to an MSG structure that specifies a keystroke to process.

HRESULT TranslateAccelerator(
LPMSG *lpMsg*
);

Parameters

lpMsg
Pointer to the MSG structure describing the keystroke to process.

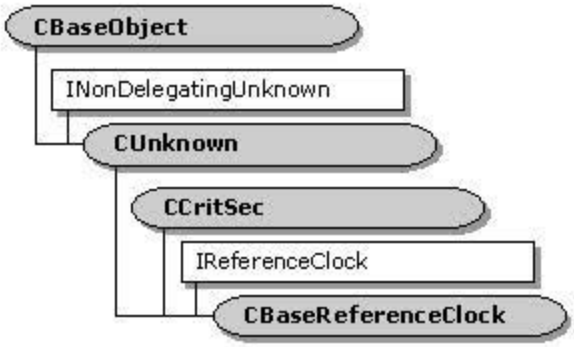
Return Values

Returns E_NOTIMPL by default.

Remarks

This member function implements the COM IPropertyPage::TranslateAccelerator method. Calls to this member function must occur after a call to CBasePropertyPage::Activate and before the corresponding call to CBasePropertyPage::Deactivate. Override this member function to implement keyboard accelerators for the property page.

CBaseReferenceClock Class



This base class implements the IReferenceClock interface.

The **CBaseReferenceClock** class provides a full implementation of IReferenceClock. It uses CCritSec locking support and CAMSchedule scheduler support.

Each *advise* call defines a point in time when the caller wants to be notified. A *periodic advise* is a regular series of such events. A list of these advise requests is maintained by the reference clock. The clock calculates the delay until the first requested advise, and signals an event at the due time.

Clients are not advised through callbacks. One-shot clients have an event set, while periodic clients have a semaphore released for each event notification. A semaphore allows a client to know exactly how many events were actually triggered, because multiple time periods might elapse before the client code executes.

During class construction, a worker thread is created. This thread executes a series of Microsoft® Win32® WaitForSingleObject calls, waking up when a command is given to the thread or the next wake-up point is reached. The wake-up points are determined by clients making advise calls.

Protected Data Members

m_pSchedule Pointer to the CAMSchedule object associated with this CBaseReferenceClock object.

Member Functions

Name	Description
CBaseReferenceClock	Constructs a CBaseReferenceClock object.
GetSchedule	Returns the CAMSchedule pointer stored in the CBaseReferenceClock::m_pSchedule data member.
SetTimeDelta	Adjusts the values returned from CBaseReferenceClock::GetPrivateTime by the amount specified in this member function.

TriggerThread	Triggers the advise thread's event. If you override CBaseReferenceClock::GetPrivateTime, you should either reuse or abandon this method.

Implemented IReferenceClock Methods

Name	Description
AdvisePeriodic	Requests an asynchronous periodic notification that a time has elapsed.
AdviseTime	Requests an asynchronous notification that a time has elapsed.
GetTime	Returns a reference time.
Unadvise	Removes a previously established advise link.

Overridable Member Functions

Name	Description
GetPrivateTime	Gets the current time from the real clock. Override this member function to implement your own clock.

Implemented INonDelegatingUnknown Methods

Name	Description
NonDelegatingQueryInterface	Returns a pointer to interfaces supported, that is, IReferenceClock.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseReferenceClock::AdvisePeriodic

CBaseReferenceClock Class

Sets up a recurring advise with the reference clock.

```
HRESULT AdvisePeriodic(
  REFERENCE_TIME StartTime,
  REFERENCE_TIME PeriodTime,
  HSEMAPHORE hSemaphore,
  DWORD *pdwAdviseToken
  );
```

Parameters

StartTime
Start at this time.
PeriodTime
Time between notifications.

hSemaphore
Advise through a semaphore.
pdwAdviseToken
Advise token that identifies the link with the clock.

Return Values

Returns one of the following HRESULT values:

Value	Meaning
E_OUTOFMEMORY	Failure.
E_INVALIDARG	Invalid argument.
NOERROR	No error.

Remarks

This member function implements the IReferenceClock::AdvisePeriodic method. A semaphore is used, rather than an event, to ensure that multiple notifications can be seen by the user. Each time an amount of time specified in the *PeriodTime* parameter elapses, the semaphore will be released.

When no further notifications are required, call CBaseReferenceClock::Unadvise and pass the *pdwAdviseToken* value returned from this call.

For example, the following code extract sets up an advise link that fires its first advise five seconds from now and then signals every second until Unadvise is called. By using a semaphore with a count of 10, the clock is effectively able to cache 10 events.

```
HANDLE hSemaphore = CreateSemaphore(NULL, 0, 10, NULL);
    // assume pRefClock is an IReferenceClock* variable

    REFERENCE_TIME rtPeriodTime = 1000 * (UNITS / MILLISECONDS);
        // a one-second interval
    REFERENCE_TIME rtNow;
    DWORD    dwAdviseToken;
    pRefClock->GetTime(&rtNow);

    pRefClock->AdvisePeriodic(rtNow+(5*rtPeriodTime),
        PeriodTime, hSemaphore, &dwAdviseToken);
```


Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseReferenceClock::AdviseTime

CBaseReferenceClock Class

Sets up a one-shot notification with the clock.

HRESULT AdviseTime(
REFERENCE_TIME *baseTime*,
REFERENCE_TIME *streamTime*,
HEVENT *hEvent*,
DWORD **pdwAdviseToken*
);

Parameters

baseTime
Base reference time.
streamTime
Stream offset time.
hEvent
Advise through this event.
pdwAdviseToken
Where the advise token goes.

Return Values

Returns one of the following HRESULT values:

Value	Meaning
E_OUTOFMEMORY	Failure.
E_INVALIDARG	Invalid argument.
NOERROR	No error.

Remarks

This member function implements the IReferenceClock::AdviseTime method. At the time specified in the *baseTime* plus the *streamTime* parameters, the event specified in *hEvent* will be set. It is correct to call CBaseReferenceClock::Unadvise to remove the link after the event has occurred, but it is not necessary. One-shot notifications are automatically cleared by the clock once they have signaled.

To cancel a one-shot notification before the time is reached, call Unadvise and pass the *pdwAdviseToken* value returned from this call.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseReferenceClock::CBaseReferenceClock

CBaseReferenceClock Class

Constructs a CBaseReferenceClock object.

CBaseReferenceClock(
TCHAR **pName*,
LPUNKNOWN *pUnk*,
HRESULT **phr*,
CAMSchedule * *pSched*
);

Parameters

pName
Name of the CBaseReferenceClock object.
pUnk
IUnknown interface of the delegating object.
phr
Address of an HRESULT value.
pSched
Address of a CAMSchedule object that will be associated with this CBaseReferenceClock object. If *pSched* is NULL, the constructor creates a new **CAMSchedule** object and associates it with this **CBaseReferenceClock** object.

Return Values

No return value.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseReferenceClock::GetPrivateTime

CBaseReferenceClock Class

Retrieves the current reference time.

virtual REFERENCE_TIME GetPrivateTime( );

Return Values

Returns the current reference time, in 100-nanosecond units.

Remarks

GetPrivateTime represents the actual clock. GetTime is the externally used member function. Derived classes will probably override this method, but not **GetTime** itself. The important point about **GetPrivateTime** is that it is allowed to go backward. This class's **GetTime** member function will keep returning the last time returned by **GetTime** until **GetPrivateTime** catches up.

Previous Home Topic Contents Index Next

CBaseReferenceClock::GetSchedule

CBaseReferenceClock Class

Retrieves the CAMSchedule pointer stored in the CBaseReferenceClock::m_pSchedule data member.

CAMSchedule * GetSchedule( );

Previous Home Topic Contents Index Next

CBaseReferenceClock::GetTime

CBaseReferenceClock Class

Retrieves the current reference time, in 100-nanosecond units.

HRESULT GetTime(
REFERENCE_TIME *pTime*
);

Parameters

pTime
Where the current time is returned.

Return Values

Returns one of the following HRESULT values:

Value	Meaning
E_POINTER	NULL pointer argument.
S_FALSE	Failure.
S_OK	Success.

Remarks

This member function implements the IReferenceClock::GetTime method. It reads the time from the implemented clock and returns the current time.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseReferenceClock::NonDelegatingQueryInterf

CBaseReferenceClock Class

Accesses supported interfaces.

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

Parameters

riid
IID of the interface being requested. Only IID_IReferenceClock is supported by the clock interface.

ppv
Where the IReferenceClock pointer is returned.

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.

CBaseReferenceClock::SetTimeDelta

CBaseReferenceClock Class

Sets a delta on the time that CBaseReferenceClock::GetPrivateTime will return.

HRESULT SetTimeDelta(
const REFERENCE_TIME& *TimeDelta*
);

Parameters

TimeDelta
REFERENCE_TIME delta to be added.

Return Values

Returns NOERROR.

Remarks

Note that CBaseReferenceClock::GetTime will never return a time earlier than a previously returned time. Thus, if you set the clock to a time in the past, successive calls to **CBaseReferenceClock::GetTime** will return the current value until this new time is reached, and the clock will start to increment again.

CBaseReferenceClock::TriggerThread

CBaseReferenceClock Class

Triggers the advise thread's event.

void TriggerThread( );

Return Values

No return value.

Remarks

The clock uses a worker thread that should wake up and call CAMSchedule::Advise at the appropriate time. If the clock adds an event that should be fired earlier than any currently outstanding event, the worker thread needs to be awoken in order to reevaluate its wait time. The **TriggerThread** member function will wake up the worker thread so this can take place. If a derived clock causes time to jump forward for some reason, **TriggerThread** should be called so that the wait time can be reevaluated; otherwise, the events will fire late.

Previous Home Topic Contents Index Next

CBaseReferenceClock::Unadvise

CBaseReferenceClock Class

Removes a previously established advise link.

HRESULT Unadvise(
  DWORD *dwAdviseToken*
  );

Parameters

dwAdviseToken
Identifier (token) of the link that is being reset. This is the same value that was returned on CBaseReferenceClock::AdviseTime or CBaseReferenceClock::AdvisePeriodic.

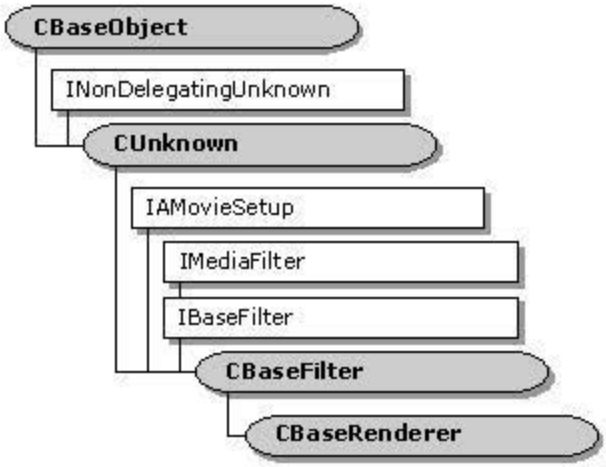
Return Values

Returns S_OK if the successful, S_FALSE if not.

Remarks

This member function implements the IReferenceClock::Unadvise method. Call **Unadvise** to remove the previously established clock advise links. It is mandatory to call **Unadvise** on periodic advises in order to stop further calls. It is recommended to call **Unadvise** for single-shot advise links.

CBaseRenderer Class



CBaseRenderer is the base renderer class for writing renderers. This class handles a single input pin, all state changes, and synchronization.

Protected Data Members

Name	Description
m_bAbort	Stop rendering data.
m_bEOS	Indicator for whether there are more samples in the stream.
m_bEOSDelivered	Indicator for whether an EC_COMPLETE event has been delivered.
m_bRepaintStatus	Flag to determine if an EC_REPAINT message can be signaled.
m_bStreaming	Indicator for whether the filter graph is currently streaming.
m_dwAdvise	Timer advise token returned by the clock.
m_EndOfStreamTimer	Time that specifies the end of the stream.
m_evComplete	Event signaled when the pause state is complete.
m_InterfaceLock	Critical section for interfaces.
m_pInputPin	Renderer input pin object.
m_pMediaSample	Current media sample about to be, or being rendered.
m_pPosition	CRendererPosPassThru object for passing positioning data upstream.
m_pQSink	Quality control sink.
m_RendererLock	Controller for access to current media sample.
m_RenderEvent	Used to signal timer events.
m_SignalTime	Amount of time that must elapse before CBaseRenderer returns EC_COMPLETE.
m_ThreadSignal	Event signaled to release the source filter thread.

Member Functions

Name	Description
Active	Called when the state is switched to paused or running. Override to add functionality.
CBaseRenderer	Constructs a CBaseRenderer object.
CheckReady	Determines if the event is set.
DisplayRendererState	Displays the status of the video renderer. This function is available only in debug mode.
GetRealState	Retrieves the actual state of the renderer.
GetRenderEvent	Retrieves the event to render.
IsEndOfStream	Determines if the end of the stream has been reached.
IsEndOfStreamDelivered	Determines if the end of the stream has been delivered to the filter graph manager.
IsStreaming	Determines if the filter is currently rendering data.
NotifyEndOfStream	Sends an EC_COMPLETE event to the filter graph manager.
NotReady	Forces the m_evComplete event into a nonsignaled state.
Ready	Puts the m_evComplete event into a signaled state.
ResetEndOfStreamTimer	Sets the end of stream timer to zero.
ScheduleSample	Sets up an advise link with the clock.
SendNotifyWindow	Passes the notification window handle to the upstream filter.
SendRepaint	Conditionally signals an EC_REPAINT message to the filter graph.
SetAbortSignal	Sets the abort signal flag.
SetRepaintStatus	Resets the repaint status flag.
SignalTimerFired	Resets the current advise time to zero after a timer fires.
TimerCallback	Checks if it is time to signal the end of the current data stream.

Overridable Member Functions

Name	Description
BeginFlush	Signals the start of flushing on the input pin.
BreakConnect	Breaks the input pin connection and resets the end-of-stream flags.
CancelNotification	Cancels any currently scheduled notification with the clock.
CheckMediaType	Determines if the renderer will accept a given media type.
ClearPendingSample	Called to release the pending sample after it has been rendered.
CompleteConnect	Called as part of the connection protocol. Override to add functionality.
CompleteStateChange	Ensures that a sample is waiting before allowing a pause.
DoRenderSample	Called when a sample is ready to render.
GetCurrentSample	Retrieves the current sample waiting at the video renderer.
GetPin	Returns a CBasePin object to the renderer.
GetPinCount	Returns the number of input pins supported.
EndFlush	Called when the input pin receives an end-flush notification.
EndOfStream	Called when the input pin receives an end-of-stream notification.
GetMediaPositionInterface	Retrieves IMediaPosition and IMediaSeeking interfaces for the video renderer.
GetSampleTimes	Retrieves sample time information for this sample.

HaveCurrentSample	Determines if a sample is waiting at the renderer.
Inactive	Called when going into a stopped state. Override to add functionality.
NonDelegatingQueryInterface	Returns an interface and increments the reference count.
OnReceiveFirstSample	Provides derived classes with an opportunity to render static data.
OnRenderEnd	Notifies the derived class that a sample has just finished rendering.
OnRenderStart	Notifies the derived class that a sample is about to be rendered.
OnStartStreaming	Notifies the derived class that rendering has started.
OnStopStreaming	Notifies the derived class that rendering has stopped.
OnWaitEnd	Notifies the derived class that a wait for a rendering time has just ended.
OnWaitStart	Notifies the derived class that a wait for a rendering time is about to start.
Pause	Tells the renderer to transition to the new (paused) state.
PrepareReceive	Called to schedule a clock time when the renderer receives a sample.
PrepareRender	Allows derived classes to set themselves just before a sample is rendered.
Receive	Called by the source filter when a sample is available to render.
Render	Asks the derived class to render the sample.
ResetEndOfStream	Resets the end-of-stream flag.
Run	Transitions the renderer to State_Running if it is not already in this state.
SendEndOfStream	Sets the end-of-stream flag.
SetMediaType	Informs the derived class of the selected media type.
ShouldDrawSampleNow	Determines if the sample should be drawn between the start and stop times given.
SourceThreadCanWait	Sets or resets the thread event.
StartStreaming	Called to schedule any pending sample with the clock, and to display any timing information.
Stop	Tells the renderer to transition to the new (stopped) state.
StopStreaming	Sets an internal flag to indicate not to schedule arrival of any more samples.
WaitForRenderTime	Waits for either the time to arrive or for rendering to be stopped.

Implemented IMediaFilter Methods

Name	Description
GetState	Determines the state of the renderer.

Implemented IBaseFilter Methods

Name	Description
FindPin	Retrieves a pointer to the pin with the specified identifier. (There is only one pin.)

Helper Function

Name	Description
WaitForReceiveToComplete	Waits for the CBaseRenderer::Receive method to complete.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseRenderer::Active

CBaseRenderer Class

Called when the state is switched to paused or running.

virtual HRESULT Active(void);

Return Values

Returns an HRESULT value. Returns NOERROR by default.

Remarks

This member function does nothing by default. Derived classes can optionally override this member function to add functionality.

Previous Home Topic Contents Index Next

CBaseRenderer::BeginFlush

CBaseRenderer Class

Informs the renderer that flushing has started.

virtual HRESULT BeginFlush(void);

Return Values

Returns an HRESULT value.

Remarks

This member function is called by CRendererInputPin::BeginFlush when informed of a flush from the upstream filter. It releases the source thread and signals the start of flushing on the input pin. Any samples received by the renderer when it is in a flushing state will be rejected.

Previous Home Topic Contents Index Next

CBaseRenderer::BreakConnect

CBaseRenderer Class

Called when a connection is broken.

virtual HRESULT BreakConnect(void);

Return Values

Returns an HRESULT value.

Remarks

This member function resets the end-of-stream flag and checks for a valid connection, or that the filter is in a stopped state. Override to customize.

Previous Home Topic Contents Index Next

CBaseRenderer::CancelNotification

CBaseRenderer Class

Cancels any currently scheduled notification.

virtual HRESULT CancelNotification(void);

Return Values

Returns an HRESULT value.

Remarks

This member function is called when the renderer is told to stop streaming. If there is no timer link outstanding, calling this member function does nothing; otherwise, this function stops the advise link and resets the render event. The normal process when running is to receive a sample, wait until it is time to render it and then render it. The clock is given an event to signal when the desired time arrives.

Previous Home Topic Contents Index Next

CBaseRenderer::CBaseRenderer

CBaseRenderer Class

Constructs a CBaseRenderer object.

CBaseRenderer(
REFCLSID *RenderClass*,
TCHAR **pName*,
LPUNKNOWN *pUnk*,
HRESULT **phr*
);

Parameters

RenderClass
Class identifier for this renderer.
pName
Name used for debugging purposes.
pUnk
Owner object.
phr
Pointer to the HRESULT return code.

Return Values

No return value.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseRenderer::CheckMediaType

CBaseRenderer Class

Determines if the renderer will accept a given media type.

virtual HRESULT CheckMediaType(
const CMediaType * *pmt*
) PURE;

Parameters

pmt
Pointer to a media type object that contains the proposed media type.

Return Values

Returns an HRESULT value.

Remarks

This member function must be overridden and implemented, typically to return the media type of the display. It is called from the CRendererInputPin::CheckMediaType member function during the connection process.

Previous Home Topic Contents Index Next

CBaseRenderer::CheckReady

CBaseRenderer Class

Determines if the renderer is ready to process the next sample.

BOOL CheckReady(void);

Return Values

Returns TRUE if the m_evComplete event is currently set, but does not block.

Remarks

This member function calls the CAMEvent::Check member function. This is mainly used in

transitioning to paused states. When a renderer is paused, it should not complete the state change until it has received some data. So although the call to IMediaFilter::Pause completes immediately, if the application calls IMediaFilter::GetState it will return VFW_S_STATE_INTERMEDIATE. When a sample arrives at the renderer, the event that is initially reset during the pause call will be signaled. At this point, an application calling **IMediaFilter::GetState** will return NOERROR. This process allows an application to pause a filter graph and then wait until data is actually queued and ready to be rendered.

Previous Home Topic Contents Index Next

CBaseRenderer::ClearPendingSample

CBaseRenderer Class

Called to clear the pending sample when in a stopped or inactivated state.

virtual HRESULT ClearPendingSample(void);

Return Values

Returns an HRESULT value.

Remarks

This member function releases the IMediaSample interface. This allows the allocator to reuse it and allocate it to the upstream filter again. If the state is being changed to inactive, IMemAllocator::GetBuffer will return an error. This function also resets the current media sample to NULL to indicate that no data is now available.

Previous Home Topic Contents Index Next

CBaseRenderer::CompleteConnect

CBaseRenderer Class

Called as part of the connection protocol.

virtual HRESULT CompleteConnect(
IPin *pReceivePin*
);

Parameters

pReceivePin
Connecting pin.

Return Values

Returns an HRESULT value (NOERROR by default).

Remarks

This member function calls the SetRepaintStatus member function to set the m_bRepaintStatus data member to TRUE so that EC_REPAINT notifications can be sent in the future. (To prevent unnecessary EC_REPAINT notifications from being sent, **m_bRepaintStatus** is set to FALSE when an end-of-stream notification arrives.)

Previous Home Topic Contents Index Next

CBaseRenderer::CompleteStateChange

CBaseRenderer Class

Ensures that a sample is waiting before allowing a pause.

virtual HRESULT CompleteStateChange(
FILTER_STATE *OldState*
);

Parameters

OldState
State prior to the transition.

Return Values

Returns S_OK if the filter can be paused; otherwise, returns S_FALSE.

Remarks

This member function is called from the CBaseRenderer::Pause member function. If the filter is being paused and there is no sample waiting, the transition is not completed and the function returns S_FALSE until the first sample arrives. However, if the m_bAbort flag has been set, all samples are rejected so there is no point waiting for one. If a sample is available, this member function returns NOERROR.

Previous Home Topic Contents Index Next

CBaseRenderer::DisplayRendererState

CBaseRenderer Class

Displays the status of the video renderer. This function is available only in debug mode.

void CBaseRenderer::DisplayRendererState( );

Return Values

No return value.

Remarks

Use this function to monitor the activity of the video renderer. The following is a sample output of this function.

```
Timed out in WaitForRenderTime
Signal sanity check 0
Filter state 1
Abort flag 0
Streaming flag 0
Clock advise link 0
Current media sample 0
EOS signalled 0
EOS delivered 0
Repaint status 1
End of stream timer 0
Deliver time 0x000000000
Flushing sanity check 0
Last run time 0x000000000
Clock time 0x22C2CD23430
Time difference 238875379ms
```


Previous Home Topic Contents Index Next

CBaseRenderer::DoRenderSample

CBaseRenderer Class

Called when a sample is ready to render.

virtual HRESULT DoRenderSample(
IMediaSample **pMediaSample*
) PURE;

Parameters

pMediaSample
Media sample.

Return Values

Returns an HRESULT value.

Remarks

This member function must be overridden in the derived class. It is called by CBaseRenderer::Render.

The derived class should render the object at this time. For example, the sample video renderer (SAMPVID) calls its drawing object (a CDrawImage object):

```
// Have the drawing object render the current image

HRESULT CVideoRenderer::DoRenderSample(IMediaSample *pMediaSample)
{
    return m_DrawImage.DrawImage(pMediaSample);
}
```


Previous Home Topic Contents Index Next

CBaseRenderer::EndFlush

CBaseRenderer Class

Called when the input pin receives an end-flush notification.

virtual HRESULT EndFlush(void);

Return Values

Returns an HRESULT value.

Remarks

This member function is called from the CRendererInputPin::EndFlush member function. It calls CBaseRenderer::SourceThreadCanWait with a TRUE value to allow the upstream filter's thread to wait in CBaseRenderer::Receive again.

Previous Home Topic Contents Index Next

CBaseRenderer::EndOfStream

CBaseRenderer Class

Called when the input pin receives an end-of-stream notification.

HRESULT EndOfStream(void);

Return Values

Returns an HRESULT value.

Remarks

If all received samples have been rendered, this member function notifies EC_COMPLETE. If samples have been received and not yet rendered, this function sets m_bEOS and checks for it on completing samples. If the filter is waiting to be paused, this function completes the transition to paused state by setting the state event.

Previous Home Topic Contents Index Next

CBaseRenderer::FindPin

CBaseRenderer Class

Retrieves a pointer to 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 renderer has been restored.

Return Values

Returns NOERROR if successful; otherwise, returns VFW_E_NOT_FOUND.

Remarks

This member function implements the IBaseFilter::FindPin method. It assumes that the default pin name is "In" and checks for this. If the pin is found, its reference count is incremented. The *ppPin* parameter is set to NULL if the identifier cannot be matched.

Previous Home Topic Contents Index Next

CBaseRenderer::GetCurrentSample

CBaseRenderer Class

Retrieves the current sample waiting at the video renderer, or NULL if there is not one.

virtual IMediaSample *GetCurrentSample(void);

Return Values

Returns a pointer to the sample.

Remarks

The reference count for the sample is incremented before returning. This is so that if the sample comes due for rendering, it is not added back to the allocator free list until the caller of this member function releases it.

Previous Home Topic Contents Index Next

CBaseRenderer::GetMediaPositionInterface

CBaseRenderer Class

Retrieves IMediaPosition and IMediaSeeking interfaces for the video renderer.

virtual HRESULT GetMediaPositionInterface(
REFIID *riid*,
void ***ppv*
);

Parameters

riid
Reference identifier.
ppv
Pointer to the interface.

Return Values

Returns an HRESULT value.

Remarks

A CRendererPosPassThru helper object is created dynamically when this is called to support passing the IMediaPosition or IMediaSeeking interface calls from the filter graph manager to the upstream filter.

Previous Home Topic Contents Index Next

CBaseRenderer::GetPin

CBaseRenderer Class

Returns a CBasePin object on the renderer.

virtual CBasePin *GetPin(
int *n*
);

Parameters

n
Number of the specified pin, which is always zero in the case of the renderer.

Return Values

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

Remarks

This member function overrides CBaseFilter::GetPin. Only one pin is supported on the renderer; it is numbered zero. A call to this member function with *n* equal to zero will result in an input pin of type CRendererInputPin being returned. It will be created if it does not yet exist.

Previous Home Topic Contents Index Next

CBaseRenderer::GetPinCount

CBaseRenderer Class

Retrieves the number of input pins supported.

virtual int GetPinCount(void);

Return Values

The default implementation returns one, since only one pin is supported. Override to support more than one pin. Because the base renderer class is specifically designed for single-pin operation, considerably more of the base class functionality would have to be changed to make a multipin renderer. Future versions of the SDK might provide this functionality.

Remarks

This member function overrides CBaseFilter::GetPinCount.

Previous Home Topic Contents Index Next

CBaseRenderer::GetRealState

CBaseRenderer Class

Retrieves the actual state of the renderer.

FILTER_STATE GetRealState(void);

Return Values

Returns m_State, the state flag for the renderer.

Remarks

This member function provides an internal way of getting the real state. Calling through the IBaseFilter interface to get the state would require the main filter critical section to be taken; this internal method does not do this.

Previous Home Topic Contents Index Next

CBaseRenderer::GetRenderEvent

CBaseRenderer Class

Retrieves the event to render.

CAMEvent *GetRenderEvent(void);

Return Values

Returns the value of m_RenderEvent.

Previous Home Topic Contents Index Next

CBaseRenderer::GetSampleTimes

CBaseRenderer Class

Retrieves sample time information for this sample.

virtual HRESULT GetSampleTimes(
IMediaSample *pMediaSample*,

REFERENCE_TIME **pStartTime*,
REFERENCE_TIME **pEndTime*
);

Parameters

pMediaSample
Media sample.
pStartTime
Start time.
pEndTime
End time.

Return Values

Returns S_FALSE if the sample should be scheduled according to the times specified in the sample; returns S_OK to indicate that the sample should be rendered immediately.

Remarks

Note that the sample times are passed in by reference, not value.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CBaseRenderer::GetState

CBaseRenderer Class

Determines the state of the renderer.

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

Parameters

dwMilliSecsTimeout
Duration of the time-out, in milliseconds.
State
Returned state of the renderer.

Return Values

Returns an HRESULT value. Returns VFW_S_STATE_INTERMEDIATE if paused and waiting for a sample; otherwise, returns NOERROR.

Remarks

This member function overrides the CBaseFilter::GetState member function. It returns the value of m_State. Because the renderer does not complete the full transition to the paused state until it has a sample to render, if the state is requested while it is waiting for a sample, it will return VFW_S_STATE_INTERMEDIATE along with the state.

Previous Home Topic Contents Index Next

CBaseRenderer::HaveCurrentSample

CBaseRenderer Class

Determines if a sample is waiting at the renderer.

virtual BOOL HaveCurrentSample(void);

Return Values

Returns TRUE if a sample is ready to be rendered, or FALSE if no data is available.

Previous Home Topic Contents Index Next

CBaseRenderer::Inactive

CBaseRenderer Class

Called when going into a stopped state.

virtual HRESULT Inactive(void);

Return Values

Returns NOERROR by default; overriding member function should return a valid HRESULT value.

Remarks

This member function is a placeholder that derived classes can optionally override to add functionality when the filter is stopped.

Previous Home Topic Contents Index Next

CBaseRenderer::IsEndOfStream

CBaseRenderer Class

Determines if the end of the stream has been reached.

BOOL IsEndOfStream(void);

Return Values

Returns TRUE if the stream's end has been reached, or FALSE if it hasn't.

Remarks

The default implementation returns m_bEOS.

Previous Home Topic Contents Index Next

CBaseRenderer::IsEndOfStreamDelivered

CBaseRenderer Class

Determines if the end of the stream has been delivered to the filter graph manager.

BOOL IsEndOfStreamDelivered(void);

Return Values

Returns TRUE if the stream's end has been delivered, or FALSE if it hasn't.

Remarks