CPersistStream Class

CPersistStream is the base class for persistent properties of filters (that is, filter properties in saved graphs).

The simplest way to use **CPersistStream** is to:

1. Arrange for your filter to inherit this class.
2. Implement WriteToStream and ReadFromStream in your class. These will override the functions here, which do nothing but act as placeholders.
3. Change your **NonDelegatingQueryInterface** to handle IPersistStream.
4. Implement SizeMax to return an upper bound on the number of bytes of data you save.

   If you save Unicode data, remember that a WCHAR is 2 bytes.

5. When your data changes, call SetDirty.

Version Numbers

At some point you might decide to alter or extend the format of your data. You will then wish you had a version number in all the old saved streams so you can tell, when you read them, whether they represent the old or new form. To assist you, this class writes and reads a version number. When it writes, it calls GetSoftwareVersion to inquire as to the version of the software being used at the moment. (In effect, this is a version number of the data layout in the file.) It writes this as the first thing in the data. If you want to change the version, implement (override) **GetSoftwareVersion**. It reads the version number from the file into mPS_dwFileVersion before calling ReadFromStream, so in **ReadFromStream** you can check **mPS_dwFileVersion** to see if you are reading an old version file. Usually you should accept files whose version is no newer than the software version that is reading them.

Protected Data Members

Name	Description
mPS_dwFileVersion	Version number of the file.
mPS_fDirty	Data for this stream must be saved.

Member Functions

Name	Description
CPersistStream	Constructs a CPersistStream object.
SetDirty	Indicates that the object must be saved to the stream.

Overridable Member Functions

Name	Description
GetClassID	Returns the class identifier of this stream.
GetSoftwareVersion	Returns the version number for this file format.
ReadFromStream	Reads the filter's data from the stream.
SizeMax	Returns the number of bytes needed for data (not including version number).
WriteToStream	Writes the filter's data to the stream.

CPersistStream implements IPersistStream. For more implementation information, see the COM Reference in the Microsoft Platform SDK.

Implemented IPersistStream Methods

Name	Description
GetSizeMax	Returns the number of bytes needed for data (including version number).
IsDirty	Checks if the object must be saved.
Load	Loads the data from the stream into memory.
Save	Saves the data from memory to the stream.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CPersistStream::CPersistStream

CPersistStream Class

Constructs a CPersistStream object.

```
CPersistStream(
  IUnknown *pUnk,
  HRESULT *phr
  );
```

Parameters

pUnk
IUnknown interface of the delegating object.

phr
Pointer to the general COM return value. Note that this value is changed only if this function fails.

Return Values

No return value.

Previous Home Topic Contents Index Next

CPersistStream::GetClassID

CPersistStream Class

Retrieves the class identifier for this filter.

```
HRESULT GetClassID(
  CLSID *pClsID
  );
```

Parameters

pClsID
Pointer to a CLSID structure. Copy your class ID to here.

Return Values

Returns an HRESULT value.

Previous Home Topic Contents Index Next

CPersistStream::GetSizeMax

CPersistStream Class

Returns the maximum byte size needed for the current stream, including the version number.

```
HRESULT GetSizeMax(
  ULARGE_INTEGER * pcbSize
  );
```

Parameters

pcbSize
Size in bytes needed to save this stream, including the version number.

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IPersistStream::GetSizeMax method.

Previous Home Topic Contents Index Next

CPersistStream::GetSoftwareVersion

CPersistStream Class

Returns the software version for this stream.

virtual DWORD GetSoftwareVersion(void);

Return Values

Returns a DWORD containing the version number. Each time the format of the stream is changed, this function should be altered to return a larger number than before.

Remarks

See Version Numbers for an explanation as to why file format version numbers are useful.

Previous Home Topic Contents Index Next

CPersistStream::IsDirty

CPersistStream Class

Indicates whether the object has changed since it was last saved to its current stream.

HRESULT IsDirty( );

Return Values

Returns S_OK if the filter needs saving and S_FALSE if it does not need saving.

Remarks

This member function implements the IPersistStream::IsDirty method.

Previous Home Topic Contents Index Next

CPersistStream::Load

CPersistStream Class

Loads the filter's data from a given stream.

HRESULT Load(
LPSTREAM *pStm*
);

Parameters

pStm
Pointer to the stream from which to be loaded.

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IPersistStream::Load method.

Previous Home Topic Contents Index Next

CPersistStream::ReadFromStream

CPersistStream Class

Reads the filter's data from the given stream.

virtual HRESULT ReadFromStream(
IStream **pStream*
);

Parameters

pStream
Pointer to an IStream interface from which data is to be read.

Return Values

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

Remarks

The default version reads nothing; it can be overridden to read data specific to your class.

Previous Home Topic Contents Index Next

CPersistStream::Save

CPersistStream Class

Saves the filter's data to the given stream.

HRESULT Save(
LPSTREAM *pStm*,
BOOL *fClearDirty*
);

Parameters

pStm
Pointer to the stream to which data is to be saved.
fClearDirty
Flag that indicates whether to reset the current stream's dirty flag. When called as part of Save, the flag is normally reset; when called as part of Save As, the flag is normally not reset.

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IPersistStream::Save method. It calls WriteInt with the software version, calls CPersistStream::WriteToStream with the stream in *pStm*, and resets mPS_fDirty.

Previous Home Topic Contents Index Next

CPersistStream::SetDirty

CPersistStream Class

Changes the dirty flag for the current stream.

```
HRESULT SetDirty(
  BOOL fDirty
  );
```

Parameters

fDirty
New dirty flag for this stream. TRUE means that the data has not been saved.

Return Values

Returns an HRESULT value.

Previous Home Topic Contents Index Next

CPersistStream::SizeMax

CPersistStream Class

Retrieves the maximum byte size needed for the current stream, not including the version

number.

virtual int SizeMax( );

Return Values

Returns the number of bytes needed for data, not including the version number.

Remarks

The default version returns zero; it should be overridden to provide some other appropriate value.

Previous Home Topic Contents Index Next

CPersistStream::WriteToStream

CPersistStream Class

Writes the filter's data to the given stream.

virtual HRESULT WriteToStream(
IStream **pStream*
);

Parameters

pStream
Pointer to an IStream interface that specifies the filter data's destination stream.

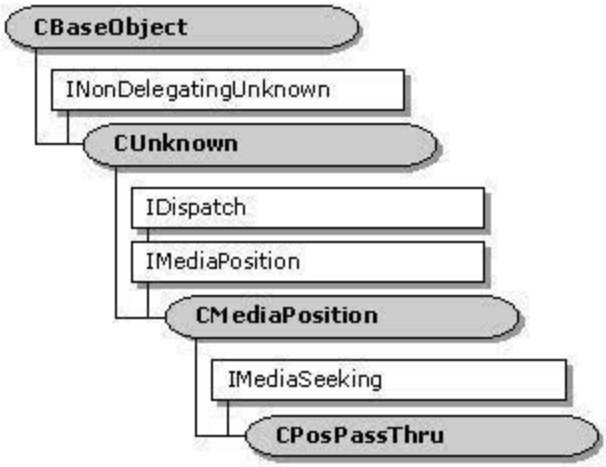
Return Values

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

Remarks

The default version writes nothing; it can be overridden to write data specific to your class.

CPosPassThru Class



The **CPosPassThru** class is a utility class that handles the IMediaPosition and IMediaSeeking interfaces for single-input pin renderers and transform filters.

IMediaPosition is the interface originally used for seeking in time-based media streams. IMediaSeeking is an interface intended to replace **IMediaPosition** in filter graphs that require seeking to units other than time, such as samples or fields, or that require more precise time-based seeking.

Renderers will use this class to implement IMediaPosition and IMediaSeeking from the filter; transform filters will use it to implement these two interfaces from the output pin. In both cases, the methods will be implemented by calls to the **IMediaPosition** or **IMediaSeeking** interface provided by the output pin of the connected upstream filter, effectively passing the position information through to the next filter.

Create a class derived from **CPosPassThru**, giving it the IPin pointer to your input pin, and delegate all IMediaPosition and IMediaSeeking methods to it. The class will find the output pin connected to the input pin, query this output pin for the **IMediaPosition** or **IMediaSeeking** interface, and respond appropriately.

Protected Data Members

Name	Description
m_Pin	Pointer to the input pin of the filter.

Member Functions

Name	Description
CPosPassThru	Constructs a CPosPassThru object.
ForceRefresh	Releases any cached interfaces held on the upstream pin.

Overridable Member Functions

Name	Description
GetMediaTime	Retrieves the starting and ending media times.

Implemented IMediaPosition Methods

Name	Description
CanSeekBackward	Determines if the current position can be moved backward in the media stream.
CanSeekForward	Determines if the current position can be moved forward in the media stream.
get_CurrentPosition	Retrieves the current position in terms of the total length of the media stream.
get_Duration	Retrieves the total duration of the media stream.
get_PrerollTime	Retrieves the time before the start position that the filter graph will start any nonrandom access device rolling.
get_Rate	Retrieves the playback rate, relative to normal playback of the media.
get_StopTime	Retrieves the position within the media at which playback should stop.
put_CurrentPosition	Sets the position within the media at which playback should start.
put_PrerollTime	Sets the time before the start position that the filter graph will start any nonrandom access device rolling.
put_Rate	Sets the playback rate, relative to normal playback of the media.
put_StopTime	Sets the position within the media at which playback should stop.

Implemented IMediaSeeking Methods

Name	Description
CheckCapabilities	Determines which capabilities exist on a media stream by applying seeking capability flags and checking the returned value.
ConvertTimeFormat	Converts a time from one time format to another.
GetAvailable	Returns the range of times in which seeking is efficient.
GetCapabilities	Retrieves the seeking capabilities of the media stream.
GetCurrentPosition	Retrieves the current position within the media stream.
GetDuration	Retrieves the length of time that the media stream will play.
GetPositions	Retrieves the current start and stop position settings.
GetPreroll	Retrieves the preroll settings.
GetRate	Retrieves the current rate.
GetStopPosition	Retrieves the position at which the media stream stops.
GetTimeFormat	Retrieves the current media time format.
IsFormatSupported	Determines if a specified time format is supported.
IsUsingTimeFormat	Determines if the time format being used in the call is the same as the one the interface currently uses.
QueryPreferredFormat	Retrieves the preferred time format the interface will use.
SetPositions	Sets current and stop positions and applies flags to both.
SetRate	Sets a new playback rate.
SetTimeFormat	Sets the time format, which determines the format of units used during seeking.

Implemented INonDelegatingUnknown Methods

Name	Description
NonDelegatingQueryInterface	Returns a specified reference-counted interface.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CPosPassThru::CanSeekBackward

CPosPassThru Class

Determines if the current position can be moved backward in the media stream.

HRESULT CanSeekBackward(
LONG *pCanSeekBackward*
);

Parameters

pCanSeekBackward
Set to OATRUE if able to seek backward; otherwise set to OAFALSE.

Return Values

Returns the HRESULT value returned from calling IMediaPosition::CanSeekBackward on the connected pin.

Remarks

This member function calls IMediaPosition::CanSeekBackward on the connected pin and returns the result.

Previous Home Topic Contents Index Next

CPosPassThru::CanSeekForward

CPosPassThru Class

Determines if the current position can be moved forward in the media stream.

HRESULT CanSeekForward(
LONG **pCanSeekForward*
);

Parameters

pCanSeekForward
Set to OATRUE if able to seek forward; otherwise set to OAFALSE.

Return Values

Returns the HRESULT value returned from calling IMediaPosition::CanSeekForward on the connected pin.

Remarks

This member function calls IMediaPosition::CanSeekForward on the upstream output pin connected to the peer input pin and returns the result.

Previous Home Topic Contents Index Next

CPosPassThru::CheckCapabilities

CPosPassThru Class

Determines which capabilities exist on a media stream by applying seeking capability flags and checking the returned value.

HRESULT CheckCapabilities(
DWORD * *pCapabilities*
);

Parameters

pCapabilities
Pointer to an **AM_SEEKING_SEEKING_CAPABILITIES** enumerator containing the seeking capabilities flags to apply. These flags can be any combination of the following:

AM_SEEKING_CanGetCurrentPos
AM_SEEKING_CanGetDuration
AM_SEEKING_CanGetStopPos
AM_SEEKING_CanPlayBackwards
AM_SEEKING_CanSeekAbsolute
AM_SEEKING_CanSeekBackwards
AM_SEEKING_CanSeekForwards

Return Values

Returns S_OK if all capabilities in *pCapabilities* are present, S_FALSE if some are present, or E_FAIL if none are present.

Remarks

This member function implements IMediaSeeking::CheckCapabilities, by calling the **IMediaSeeking::CheckCapabilities** method on the upstream output pin connected to the peer input pin. The pin that performs the seek operation will return whether the flags presented in the *pCapabilities* parameter are present. This returned value will then, in turn, propagate to calls made from **CPosPassThru::CheckCapabilities** member functions in intervening filters.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CPosPassThru::ConvertTimeFormat

CPosPassThru Class

Converts a time from one format to another.

HRESULT ConvertTimeFormat(
LONGLONG * *pTarget*,
const GUID * *pTargetFormat*,
LONGLONG *Source*,
const GUID * *pSourceFormat*
);

Parameters

pTarget
Time in converted format.
pTargetFormat

GUID of the format to convert to, or the currently selected format if NULL.

Source

Time in original format.

pSourceFormat

GUID of the format to be converted from, or the currently selected format if NULL.

Return Values

Returns the HRESULT value returned from calling IMediaSeeking::ConvertTimeFormat on the connected pin.

Remarks

This member function implements the IMediaSeeking::ConvertTimeFormat method by calling this same method on the upstream filter's output pin.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CPosPassThru::CPosPassThru

CPosPassThru Class

Constructs a CPosPassThru object.

```
CPosPassThru(
  const TCHAR *pName,
  LPUNKNOWN pUnk,
  HRESULT * phr,
  IPin * pPin
  );
```

Parameters

pName

Name of the object used in the CPosPassThru constructor for debugging purposes.

pUnk

Pointer to the owner of this object.

phr

Pointer to an HRESULT value for resulting information.

pPin

Pointer to the input pin for the filter.

Return Values

No return value.

Remarks

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

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CPosPassThru::ForceRefresh

CPosPassThru Class

Releases any cached interfaces on the upstream pin.

HRESULT ForceRefresh( );

Return Values

Returns S_OK.

Remarks

For efficiency, the CPosPassThru class can cache the IMediaPosition interface of the connected upstream output pin. This method releases any cached interface pointers and forces them to be obtained again via QueryInterface if needed.

Presently, this class does not cache the upstream IMediaPosition so this member function is not necessary. It is left in for future flexibility.

Previous Home Topic Contents Index Next

CPosPassThru::GetAvailable

CPosPassThru Class

Returns the range of times in which seeking is efficient.

HRESULT GetAvailable(
LONGLONG * *pEarliest*,
LONGLONG * *pLatest*
);

Parameters

pEarliest
Earliest time that can be efficiently seeked to.
pLatest
Latest time that can be efficiently seeked to.

Return Values

Return Values

Returns the HRESULT value returned from calling IMediaSeeking::GetAvailable on the connected pin.

Remarks

This member function implements the IMediaSeeking::GetAvailable method by calling this same method on the upstream filter's output pin.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CPosPassThru::GetCapabilities

CPosPassThru Class

Returns the seeking capabilities of the media stream.

HRESULT GetCapabilities(
DWORD * *pCapabilities*
);

Parameters

pCapabilities
Seeking capability flags, which can be any combination of the following.

AM_SEEKING_CanGetCurrentPos
AM_SEEKING_CanGetDuration
AM_SEEKING_CanGetStopPos
AM_SEEKING_CanPlayBackwards
AM_SEEKING_CanSeekAbsolute
AM_SEEKING_CanSeekBackwards
AM_SEEKING_CanSeekForwards

Return Values

Return Values

Returns the HRESULT value returned from calling IMediaSeeking::GetCapabilities on the connected pin.

Remarks

This member function implements IMediaSeeking::GetCapabilities by calling the **IMediaSeeking::GetCapabilities** method on the upstream output pin connected to the peer input pin. The pin that performs the seek operation will return the capabilities present in the *pCapabilities* parameter. These returned capabilities will then, in turn, propagate to calls made from **CPosPassThru::GetCapabilities** member functions in intervening filters.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CPosPassThru::GetCurrentPosition

CPosPassThru Class

Retrieves the current position in terms of the media stream's total length.

HRESULT GetCurrentPosition(
LONGLONG* *pCurrent*
);

Parameters

pCurrent
Current position in current time format units.

Return Values

Returns NOERROR if successful. Otherwise, returns the HRESULT value returned from calling IMediaSeeking::GetCurrentPosition on the connected pin.

Remarks

This member function implements the IMediaSeeking::GetCurrentPosition interface. It calls the CPosPassThru::GetMediaTime virtual member function, which you should override and implement in your derived class to return the current position. If this fails (which it does by default), the **IMediaSeeking::GetCurrentPosition** on the upstream filter's output pin is called.

Previous Home Topic Contents Index Next

CPosPassThru::get_CurrentPosition

CPosPassThru Class

Retrieves the current position in terms of the total length of the media stream.

HRESULT get_CurrentPosition(
REFTIME* *pllTime*
);

Parameters

pllTime
Returned start time as a double value in seconds.

Return Values

Return Values

Returns the HRESULT value returned from calling IMediaPosition::get_CurrentPosition on the connected pin.

Remarks

The start position is applied before the rate and therefore is the position at typical playback speed.

Previous Home Topic Contents Index Next

CPosPassThru::GetDuration

CPosPassThru Class

Retrieves the length of time that the media stream will play.

HRESULT GetDuration(
LONGLONG* *pDuration*
);

Parameters

pDuration
Returned length of the media stream.

Return Values

Return Values

Returns the HRESULT value returned from calling IMediaSeeking::GetDuration on the connected pin.

Remarks

This member function implements the IMediaSeeking::GetDuration method by calling this same method on the upstream filter's output pin.

Previous Home Topic Contents Index Next

CPosPassThru::get_Duration

CPosPassThru Class

Retrieves the total duration of the media stream.

HRESULT get_Duration(
REFTIME * *plength*
);

Parameters

plength
Returned length of the media stream.

Return Values

Return Values

Returns the HRESULT value returned from calling IMediaPosition::get_Duration on the connected pin.

Remarks

The duration assumes normal playback speed, and it is therefore unaffected by the rate.

Previous Home Topic Contents Index Next

CPosPassThru::GetMediaTime

CPosPassThru Class

Retrieves the starting and ending media times.

virtual HRESULT GetMediaTime(
LONGLONG* *pStartTime*,
LONGLONG* *pEndTime*
);

Parameters

pStartTime
Returned starting media time.
pEndTime
Returned ending media time.

Return Values

Returns an HRESULT value (E_FAIL by default).

Remarks

Override this virtual member function to return the current samples' media time. This represents the current position in terms of media time (for example, frame 20 of a total 130 frames).

Previous Home Topic Contents Index Next

CPosPassThru::GetPositions

CPosPassThru Class

Returns the current and stop position settings.

HRESULT GetPositions(
LONGLONG * *pCurrent*,
LONGLONG * *pStop*
);

Parameters

pCurrent
Start time in the current time format.
pStop
Stop time in the current time format.

Return Values

Return Values

Returns the HRESULT value returned from calling IMediaSeeking::GetPositions on the connected pin.

Remarks

This member function implements the IMediaSeeking::GetPositions method by calling this same method on the upstream filter's output pin. It allows the retrieval of several values with only one call.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CPosPassThru::GetPreroll

CPosPassThru Class

Retrieves the preroll settings.

HRESULT GetPreroll(
LONGLONG * *pllPreroll*
);

Parameters

pllPreroll
Returned preroll time.

Return Values

Return Values

Returns the HRESULT value returned from calling IMediaSeeking::GetPreroll on the connected pin.

Remarks

This member function implements the IMediaSeeking::GetPreroll method by calling this same method on the upstream filter's output pin.

Previous Home Topic Contents Index Next

CPosPassThru::get_PrerollTime

CPosPassThru Class

Retrieves the time prior to the start position that devices should start rolling.

HRESULT get_PrerollTime(
REFTIME* *pllTime*
);

Parameters

pllTime
Returned preroll time as a double value in seconds.

Return Values

Return Values

Returns the HRESULT value returned from calling IMediaPosition::get_PrerollTime on the connected pin.

Remarks

Preroll time is the time prior to the start position at which nonrandom access devices, such as tape players, should start rolling.

Previous Home Topic Contents Index Next

CPosPassThru::GetRate

CPosPassThru Class

Retrieves the current rate.

HRESULT GetRate(
double * *pdRate*
);

Parameters

pdRate
Current rate, where 1 is the normal rate.

Return Values

Return Values

Returns the HRESULT value returned from calling IMediaSeeking::GetRate on the connected pin.

Remarks

This member function implements the IMediaSeeking::GetRate method by calling this same method on the upstream filter's output pin.

Previous Home Topic Contents Index Next

CPosPassThru::get_Rate

CPosPassThru Class

Retrieves the rate of playback relative to normal playback speed.

HRESULT get_Rate(
double * *pdRate*
);

Parameters

pdRate
Returned rate.

Return Values

Return Values

Returns the HRESULT value returned from calling IMediaPosition::get_Rate on the connected pin.

Remarks

A rate of 1.0 indicates normal playback speed. A rate of 0.5 indicates half speed. A rate of –1.0 indicates normal speed in reverse.

Previous Home Topic Contents Index Next

CPosPassThru::GetStopPosition

CPosPassThru Class

Retrieves the position at which the media stream stops.

HRESULT GetStopPosition(
LONGLONG* *pStop*
);

Parameters

pStop
Returned stop time.

Return Values

Return Values

Returns the HRESULT value returned from calling IMediaSeeking::GetStopPosition on the connected pin.

Remarks

This member function implements the IMediaSeeking::GetStopPosition method by calling this same method on the upstream filter's output pin. The stop position is a time between zero and the duration of the media at which playback should stop.

The stop position is applied before the rate and therefore is the position at typical playback speed.

Previous Home Topic Contents Index Next

CPosPassThru::get_StopTime

CPosPassThru Class

Retrieves the time at which the media stream stops.

HRESULT get_StopTime(
REFTIME* *pllTime*
);

Parameters

pllTime
Returned stop time as a double value in seconds.

Return Values

Return Values

Returns the HRESULT value returned from calling IMediaPosition::get_StopTime on the connected pin.

Remarks

The stop time is a position between zero and the duration of the media at which playback should stop.

The stop position is applied before the rate and therefore is the position at typical playback speed.

Previous Home Topic Contents Index Next

CPosPassThru::GetTimeFormat

CPosPassThru Class

Retrieves the current time format, which determines the format of units used during seeking.

HRESULT GetTimeFormat(
const GUID * *pFormat*
);

Parameters

pFormat
Media time format currently supported by this interface.

Return Values

Return Values

Returns the HRESULT value returned from calling IMediaSeeking::GetTimeFormat on the connected pin.

Remarks

This member function implements the IMediaSeeking::GetTimeFormat method by calling this same method on the upstream filter's output pin.

See the IMediaSeeking::IsFormatSupported method for a list of time formats.

Previous Home Topic Contents Index Next

CPosPassThru::IsFormatSupported

CPosPassThru Class

Determines if a specified time format is supported.

HRESULT IsFormatSupported(

const GUID * *pFormat*
);

Parameters

pFormat
Time format to compare.

Return Values

Return Values

Returns the HRESULT value returned from calling IMediaSeeking::IsFormatSupported on the connected pin.

Remarks

This member function implements the IMediaSeeking::IsFormatSupported method. See that method for a list of valid time formats.

Previous Home Topic Contents Index Next

CPosPassThru::IsUsingTimeFormat

CPosPassThru Class

Determines if the time format being used in the call is the same as the one currently in use by the interface.

HRESULT IsUsingTimeFormat(
const GUID * *pFormat*
);

Parameters

pFormat
Time format to check.

Return Values

Returns S_OK if *pFormat* is the current time format; otherwise returns S_FALSE.

Remarks

This member function implements the IMediaSeeking::IsUsingTimeFormat method by calling this same method on the upstream filter's output pin. This can be used in place of

IMediaSeeking::GetTimeFormat to save copying the GUID.

Previous Home Topic Contents Index Next

CPosPassThru::NonDelegatingQueryInterface

CPosPassThru Class

Returns a specified reference-counted interface.

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

Parameters

riid
Reference identifier.
ppv
Pointer to the interface.

Return Values

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

Remarks

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

This member function implements the INonDelegatingUnknown::NonDelegatingQueryInterface method.

Previous Home Topic Contents Index Next

CPosPassThru::put_CurrentPosition

CPosPassThru Class

Sets the time that the media stream begins.

HRESULT put_CurrentPosition(
REFTIME *llTime*
);

Parameters

llTime
Start time expressed as a double value in seconds.

Return Values

Returns the HRESULT value returned from calling IMediaPosition::put_CurrentPosition on the connected pin.

Remarks

The start time is a position between zero and the duration of the media at which playback should begin when the next run command is issued. Do not call this method when the filter graph is running, only when it is paused or stopped.

Setting the start position when paused causes playback to resume from the new start position when the run command is issued.

The start position is applied before the rate and therefore is the position at typical playback speed.

Previous Home Topic Contents Index Next

CPosPassThru::put_PrerollTime

CPosPassThru Class

Sets the time prior to the start position that devices should start rolling.

HRESULT put_PrerollTime(
REFTIME *llTime*
);

Parameters

llTime
Preroll time to be set.

Return Values

Returns the HRESULT value returned from calling IMediaPosition::put_PrerollTime.

Remarks

Preroll time is the time prior to the start position at which nonrandom access devices, such as tape players, should start rolling.

Note that while this member function passes the call upstream, the IMediaPosition::put_PrerollTime method is not implemented on any Microsoft source filter.

Previous Home Topic Contents Index Next

CPosPassThru::put_Rate

CPosPassThru Class

Sets the rate of playback relative to normal speed.

HRESULT put_Rate(
double *dRate*
);

Parameters

dRate
Rate to set.

Return Values

Returns E_INVALIDARG if *dRate* is zero. Otherwise, returns the HRESULT value returned from calling IMediaPosition::put_Rate on the connected pin.

Remarks

This property allows an application to speed up or slow down playback relative to the normal default playback speed. A rate of 1.0 indicates normal playback speed. Specifying 2.0 causes playback at twice the normal rate: a video created for 10 frames per second (fps) will be played back at 20 fps, if resources permit. Audio streams played back at above-normal speed increase the pitch rather than drop frames.

Negative rates indicate reverse play. Not all media will support reverse play.

Previous Home Topic Contents Index Next

CPosPassThru::put_StopTime

CPosPassThru Class

Sets the time at which the media stream will stop.

HRESULT put_StopTime(
REFTIME *llTime*
);

Parameters

llTime
Stop time as a double value in seconds.

Return Values

Returns the HRESULT value returned from calling IMediaPosition::put_StopTime on the connected pin.

Remarks

The stop time is a position between zero and the duration of the media at which playback should stop.

The stop position is applied before the rate and therefore is the position at typical playback speed.

Previous Home Topic Contents Index Next

CPosPassThru::QueryPreferredFormat

CPosPassThru Class

Retrieves the preferred time format to be used by the interface.

HRESULT QueryPreferredFormat(
GUID **pFormat*

);

Parameters

pFormat
Time format preferred by the interface.

Return Values

Returns the HRESULT value returned from calling IMediaSeeking::QueryPreferredFormat on the connected pin.

Remarks

This member function implements the IMediaSeeking::QueryPreferredFormat method by calling this same method on the upstream filter's output pin.

See the description for IMediaSeeking::IsFormatSupported for a list of available time formats. If the time format returned is not satisfactory, use the **IMediaSeeking::IsFormatSupported** method to query for supported time formats that you can use.

Previous Home Topic Contents Index Next

CPosPassThru::SetPositions

CPosPassThru Class

Sets current and stop positions and applies flags to both.

HRESULT SetPositions(
LONGLONG * *pCurrent*,
DWORD *dwCurrentFlags*,
LONGLONG * *pStop*,
DWORD *dwStopFlags*
);

Parameters

pCurrent
Start position if stopped, or position to continue from if paused.
dwCurrentFlags
When seeking, one of these flags must be set to indicate the type of seek. See the IMediaSeeking::SetPositions method for a description of these flags.
pStop
Position in the stream at which to quit.
dwStopFlags

Stop position seeking options to be applied. These are the same as listed for *dwCurrentFlags*.

Return Values

Returns the HRESULT value returned from calling IMediaSeeking::SetPositions on the connected pin.

Remarks

This member function implements the IMediaSeeking::SetPositions method by calling this same method on the upstream filter's output pin. It allows the retrieval of several values with only one call.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CPosPassThru::SetRate

CPosPassThru Class

Sets a new playback rate.

```
HRESULT SetRate(
  double dRate
  );
```

Parameters

dRate
New rate, where 1 is the normal rate, 2 is twice as fast, and so on.

Return Values

Returns E_INVALIDARG if *dRate* is zero. Otherwise, returns the HRESULT value returned from calling IMediaSeeking::SetRate on the connected pin.

Remarks

This member function implements the IMediaSeeking::SetRate method by calling this same method on the upstream filter's output pin. It is an error to set the rate to 0.

CPosPassThru::SetTimeFormat

CPosPassThru Class

Sets the time format, which determines the format of units used during seeking.

HRESULT SetTimeFormat(
const GUID * *pFormat*
);

Parameters

pFormat
Time format to be supported by this interface.

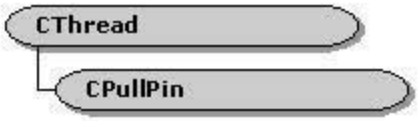
Return Values

Returns the HRESULT value returned from calling IMediaSeeking::SetTimeFormat on the connected pin.

Remarks

This member function implements the IMediaSeeking::SetTimeFormat method by calling this same method on the upstream filter's output pin. See the IMediaSeeking::IsFormatSupported method for a list of time formats.

CPullPin Class



The **CPullPin** class is provided to allow a filter downstream from the source to create a thread and pull a media stream from an asynchronous source filter that supports the IAsyncReader interface. Typically this class is implemented on the input pin of a parser filter, since the Microsoft® DirectShow™ asynchronous reader filter just reads a media stream from a file and provides no parsing.

Protected Data Members

Name	Description
m_pAlloc	Pointer to the IMemAllocator interface used by the connection.

Member Functions

Name	Description
Active	Instructs the pin to start pulling data from the asynchronous reader.
AlignDown	Aligns a LONGLONG value down to the next LONG boundary.
AlignUp	Aligns a LONGLONG value up to the next LONG boundary.
Connect	Initiates a connection from this pin to the asynchronous reader.
CPullPin	Constructs a CPullPin object.
Disconnect	Breaks a connection to the asynchronous reader.
Duration	Retrieves the total duration of the media stream.
GetReader	Retrieves the asynchronous reader interface.
Inactive	Instructs the pin to stop pulling data from the asynchronous reader.
Seek	Sets the start and stop times of the media stream.

Overridable Member Functions

Name	Description
BeginFlush	Flushes this pin and all downstream pins.
DecideAllocator	Proposes an allocator for use by the asynchronous reader.
EndFlush	Signals end of flushing operation.
EndOfStream	Sends end-of-stream notification downstream.
OnError	Handles run-time errors that caused pulling to stop.
Receive	Handles the arrival of data from the asynchronous reader.

Previous Home Topic Contents Index Next

CPullPin::Active

CPullPin Class

Instructs the pin to start pulling data from the asynchronous reader.

HRESULT Active(void);

Return Values

Returns an HRESULT value.

Remarks

The reader interface must be retrieved and the allocator decided before calling this member function. This is handled by the CPullPin::Connect member function.

Previous Home Topic Contents Index Next

CPullPin::AlignDown

CPullPin Class

Aligns a LONGLONG value down to the next LONG boundary.

LONGLONG AlignDown(
LONGLONG *ll*,
LONG *lAlign*
);

Parameters

ll
Element to be aligned.
lAlign
Alignment boundary.

Return Values

Returns the *ll* value aligned to *lAlign*.

Remarks

Aligning downward is a truncation operation.

Previous Home Topic Contents Index Next

CPullPin::AlignUp

CPullPin Class

Aligns a LONGLONG value up to the next LONG boundary.

```
LONGLONG AlignUp(
  LONGLONG ll,
  LONG lAlign
  );
```

Parameters

ll
Element to be aligned.
lAlign
Alignment boundary.

Return Values

Returns an HRESULT value.

Previous Home Topic Contents Index Next

CPullPin::BeginFlush

CPullPin Class

Override to flush this pin and all downstream pins.

virtual HRESULT BeginFlush(void);

Return Values

Returns an HRESULT value.

Remarks

This member function is called by the CPullPin::Seek member function before pausing the thread prior to a seeking operation. You must implement this member function to call the IPin::BeginFlush method on the connected downstream pin.

Previous Home Topic Contents Index Next

CPullPin::Connect

CPullPin Class

Initiates a connection from this pin to the asynchronous reader.

HRESULT Connect(
IUnknown* *pUnk*,
IMemAllocator* *pAlloc*,
BOOL *bSync*
);

Parameters

pUnk
Object to query for existence of asynchronous reader (IAsyncReader).
pAlloc
Optional allocator to propose as preferred allocator if necessary.
bSync
Set TRUE if the reader uses synchronous rather than asynchronous reads.

Return Values

Returns S_OK if successfully connected to the IAsyncReader interface from the object specified by *pUnk*.

Previous Home Topic Contents Index Next

CPullPin::CPullPin

CPullPin Class

Constructs a CPullPin object.

CPullPin(void);

Return Values

No return value.

Previous Home Topic Contents Index Next

CPullPin::DecideAllocator

CPullPin Class

Negotiates an allocator to use with the asynchronous reader.

virtual HRESULT DecideAllocator(
IMemAllocator* *pAlloc*,
ALLOCATOR_PROPERTIES * *pProps*
);

Parameters

pAlloc
Allocator to propose as the preferred allocator (optional). Pass NULL if you aren't proposing an allocator.
pProps
Size, count, and alignment of the allocator (optional). Pass 0 if not requesting the allocator properties.

Return Values

Returns S_OK if successful, VFW_E_BADALIGN if *eProps* contains an invalid alignment property, E_OUTOFMEMORY if there is not enough memory available to create an allocator, and E_NOINTERFACE if the created IMemAllocator interface is invalid.

Remarks

This member function calls the IAsyncReader::RequestAllocator method to negotiate an

allocator.

Previous Home Topic Contents Index Next

CPullPin::Disconnect

CPullPin Class

Breaks a connection to the asynchronous reader.

HRESULT Disconnect(void);

Return Values

Returns NOERROR if there is no connection.

Remarks

This member function disconnects any connection to an asynchronous file reader made in the CPullPin::Connect method.

Previous Home Topic Contents Index Next

CPullPin::Duration

CPullPin Class

Retrieves the total duration of the media stream.

HRESULT Duration(
REFERENCE_TIME* *ptDuration*
);

Parameters

ptDuration
Duration measured in bytes multiplied by UNIT (10,000,000).

Return Values

Returns an HRESULT value.

Previous Home Topic Contents Index Next

CPullPin::EndFlush

CPullPin Class

Override to signal the end of a flushing operation.

virtual HRESULT EndFlush(void) PURE;

Return Values

Returns an HRESULT value.

Remarks

This member function is called by the CPullPin::Seek member function after pausing the thread prior to a seeking operation. You must implement this member function to call the IPin::EndFlush method on the connected downstream pin.

Previous Home Topic Contents Index Next

CPullPin::EndOfStream

CPullPin Class

Override to send an end-of-stream notification downstream.

virtual HRESULT EndOfStream(void) PURE;

Return Values

Returns an HRESULT value.

Remarks

This member function is called during processing of received samples when the end of the stream is reached. You must implement this member function to call the IPin::EndOfStream method on the connected downstream pin.

Previous Home Topic Contents Index Next

CPullPin::GetReader

CPullPin Class

Returns the asynchronous reader.

IAsyncReader* GetReader(void);

Return Values

Returns a reference-counted IAsyncReader interface.

Previous Home Topic Contents Index Next

CPullPin::Inactive

CPullPin Class

Instructs the pin to stop pulling data from the asynchronous reader.

HRESULT Inactive(void);

Return Values

Returns an HRESULT value.

Remarks

This member function calls the IAsyncReader::BeginFlush method, ends the thread, calls the IAsyncReader::EndFlush method and then decommits the allocator.

Previous Home Topic Contents Index Next

CPullPin::OnError

CPullPin Class

Override to handle run-time errors that caused pulling to stop.

virtual void OnError(
HRESULT *hr*
) PURE;

Parameters

hr
HRESULT value of the trapped error.

Return Values

No return value.

Remarks

These errors are returned from the upstream filter (the asynchronous reader), which will have already reported errors to the filter graph manager. This member function must be implemented since it is called by several CPullPin member functions when trapping errors.

Previous Home Topic Contents Index Next

CPullPin::Receive

CPullPin Class

Override this member function to handle the arrival of data from the asynchronous reader.

virtual HRESULT Receive(
IMediaSample * *pSample*
) PURE;

Parameters

pSample
[in] Pointer to a media sample.

Return Values

Returns an HRESULT value. Returning a value other than S_OK will stop the data.

Remarks

You must implement this member function in your derived class. This member function is called whenever a new sample arrives while processing the sample stream. It should be written in the same manner as the IMemInputPin::Receive method on an input pin.

Previous Home Topic Contents Index Next

CPullPin::Seek

CPullPin Class

Sets the start and stop times of the media stream.

```
HRESULT Seek(
  REFERENCE_TIME tStart,
  REFERENCE_TIME tStop
  );
```

Parameters

tStart
Start time (defaults to zero).
tStop
Stop time (defaults to the value of CPullPin::Duration).

Return Values

Returns an HRESULT value.

Remarks

If the filter graph is running (active), the media rendering will start immediately at the new position defined by *tStart*.

‹Previous Home Topic Contents Index Next›

CQueue Class

CQueue

This class implements a simple Queue Abstract Data Type (ADT). The queue contains a finite number of objects, and a semaphore controls access to these objects. The semaphore is created with an initial count (N). Each time an object is added, a call to the Microsoft® Win32® WaitForSingleObject function is made on the handle of the semaphore. When this function returns, it reserves a slot in the queue for the new object. If no slots are available, the member function blocks until it becomes available. Each time an object is removed from the queue, the Win32 ReleaseSemaphore function is called on the handle of the semaphore, thus freeing a slot in the queue. If no objects are present in the queue, the function blocks until an object has been added.

Member Functions

Name	Description
CQueue	Constructs a **CQueue** object.
GetQueueObject	Retrieves an object from the queue.
PutQueueObject	Puts an object into the queue.

‹Previous Home Topic Contents Index Next›

CQueue::CQueue

CQueue Class

Constructs a CQueue object.

```
CQueue(
  int n
  );
CQueue( );
```

Parameters

n

Size of the queue to create.

Return Values

No return value.

Remarks

If constructed with no parameters, the size of the queue is set to DEFAULT_QUEUESIZE.

Previous Home Topic Contents Index Next

CQueue::GetQueueObject

CQueue Class

Retrieves an object from the queue.

T GetQueueObject( );

Return Values

Returns an object of type T (template).

Remarks

This member function blocks until an object is available on the queue. It uses a CCritSec object for security.

Previous Home Topic Contents Index Next

CQueue::PutQueueObject

CQueue Class

Puts an object into the queue.

void PutQueueObject(
T *object*
);

Parameters

object
Template object to be inserted into the queue.

Return Values

No return value.

Remarks

This member function blocks if there is no open slot into which to put the object. It releases any blocking CQueue::GetQueueObject member function that is waiting for an object to retrieve.

Previous Home Topic Contents Index Next

CRefTime Class

CRefTime

This class is used to manage reference times. It shares the same data layout as the REFERENCE_TIME data type, but adds some (nonvirtual) functions that provide simple comparison, conversion, and arithmetic capabilities.

A *reference time* is a unit of time represented in 100-nanosecond units. This time unit is the same time unit used by the Microsoft® Win32® FILETIME structure, although the two types cannot be interchanged. Note that the time a REFERENCE_TIME represents is not the time elapsed since 1/1/1601. It is either stream time or reference time, depending on the context.

Data Members

Name	Description
m_time	REFERENCE_TIME value of this object.

Member Functions

Name	Description
CRefTime	Constructs a CRefTime object.
GetUnits	Returns the reference time in units of 100 nanoseconds.
Millisecs	Returns the reference time in milliseconds.

Operators

Name	Description
operator (REFERENCE_TIME)	Casts the CRefTime object to a REFERENCE_TIME data type. The result is the m_time value.
operator =	Implements the copy constructor for the CRefTime class.
operator +=	Adds two CRefTime objects and makes this object equal to the result.
operator -=	Subtracts one CRefTime object from another **CRefTime** object and makes this object equal to the result.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CRefTime::CRefTime

CRefTime Class

Constructs a CRefTime object.

```
CRefTime( );
CRefTime(
  LONG msecs
  );
CRefTime(
  REFERENCE_TIME rt
  );
```

Parameters

msecs
CRefTime value in milliseconds.

rt
CRefTime object to copy.

Return Values

No return value.

Remarks

When constructed without parameters, the reference time value defaults to zero.

‹Previous Home Topic Contents Index Next›

CRefTime::GetUnits

CRefTime Class

Returns the reference time in 100-nanosecond units.

LONGLONG GetUnits(void);

Return Values

Returns the reference time value.

Previous Home Topic Contents Index Next

CRefTime::Millisecs

CRefTime Class

Returns the reference time in milliseconds.

LONG Millisecs(void);

Return Values

Returns the reference time value.

Previous Home Topic Contents Index Next

CRefTime::operator (REFERENCE_TIME)

CRefTime Class

Cast operator that allows a CRefTime object to be used in place of a REFERENCE_TIME object.

```
operator REFERENCE_TIME() const;
```

Return Values

Returns the value of m_time.

Remarks

The following examples show how this cast operator can be used.

```
CRefTime cRT(1000);
REFERENCE_TIME rt = (REFERENCE_TIME)cRT;
```


Previous Home Topic Contents Index Next

CRefTime::operator =

CRefTime Class

Assigns a new value to the object from an existing value.

```
CRefTime& operator=(
  const CRefTime& rt
  );
CRefTime& operator=(
  const LONGLONG ll
  );
```

Parameters

rt
Object to copy during the assignment operation.

ll
LONGLONG reference time value.

Return Values

Returns a reference to this object after the operation.

Previous Home Topic Contents Index Next

CRefTime::operator +=

CRefTime Class

Adds the value of another CRefTime object to this **CRefTime** object.

```
CRefTime& operator+=(
  const CRefTime& rt
  );
```

Parameters

rt
CRefTime object to be added.

Return Values

Returns a reference to this object.

CRefTime::operator -=

CRefTime Class

Subtracts another CRefTime object from this **CRefTime** object.

```
CRefTime& operator-=(
  const CRefTime& rt
  );
```

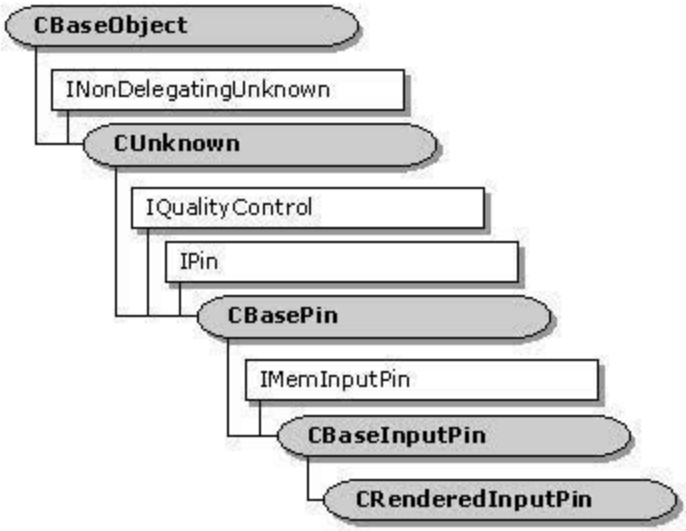
Parameters

rt
CRefTime object to be subtracted.

Return Values

Returns the result.

CRenderedInputPin Class



This input pin class is provided for renderer filters that render the stream delivered from an input pin (that is, that do not have an output pin to pass the data on). It overrides CBaseInputPin to handle the end-of-stream notification and is implemented specifically so that more than one stream can be handled using this renderer (since each pin must handle the end-of-stream independently). For an example of a filter that uses this class, see Dump Sample (Dump Filter).

Protected Data Members

Name	Description
m_bAtEndOfStream	Set to TRUE when the end-of-stream notification has been received.
m_bCompleteNotified	Set to TRUE when the EC_COMPLETE notification has been sent to the filter graph manager.

Member Functions

Name	Description
CRenderedInputPin	Constructs a CRenderedInputPin object.

Overridable Member Functions

Name	Description
Active	Notifies the pin that the filter has changed state from stopped to paused.
EndFlush	Informs the pin to end a flush operation.
EndOfStream	Informs the pin that no additional data is expected until a new run command is issued.
Run	Notifies the pin that the filter has changed state from paused to running.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CRenderedInputPin::Active

CRenderedInputPin Class

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

HRESULT Active( );

Return Values

Returns an HRESULT value.

Remarks

This member function overrides CBasePin::Active. It sets both m_bAtEndOfStream and m_bCompleteNotified to FALSE before calling the base class implementation.

Previous Home Topic Contents Index Next

CRenderedInputPin::CRenderedInputPin

CRenderedInputPin Class

Constructs a CRenderedInputPin object.

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

Parameters

pObjectName

Name of the object for debugging purposes.

pFilter
Pointer to the pin's owning filter.

pLock
Critical section for the pin.

phr
Pointer to an HRESULT value.

pName
Pin name.

Return Values

No return value.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CRenderedInputPin::EndFlush

CRenderedInputPin Class

Informs the pin to end a flush operation.

HRESULT EndFlush(void);

Return Values

Returns an HRESULT value.

Remarks

This member function overrides CBaseInputPin::EndFlush. It sets both m_bAtEndOfStream and m_bCompleteNotified to FALSE before calling the base class implementation.

Previous Home Topic Contents Index Next

CRenderedInputPin::EndOfStream

CRenderedInputPin Class

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

HRESULT EndOfStream(void);

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IPin::EndOfStream method. It calls CheckStreaming to see that the filter is in a streaming state, sets m_bAtEndOfStream to TRUE, and then sends the EC_COMPLETE notification to the filter graph manager.

The EC_COMPLETE filter graph notification should be issued only after all the data delivered to the pin prior to calling this member function has been processed. If the filter performs processing asynchronously, override **CRenderedInputPin::EndOfStream** to postpone sending the EC_COMPLETE notification until processing of all input data has completed.

Previous Home Topic Contents Index Next

CRenderedInputPin::Run

CRenderedInputPin Class

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

HRESULT Run(
REFERENCE_TIME *tStart*
);

Parameters

tStart
Start time. Unreferenced by this class; for possible use by the derived class.

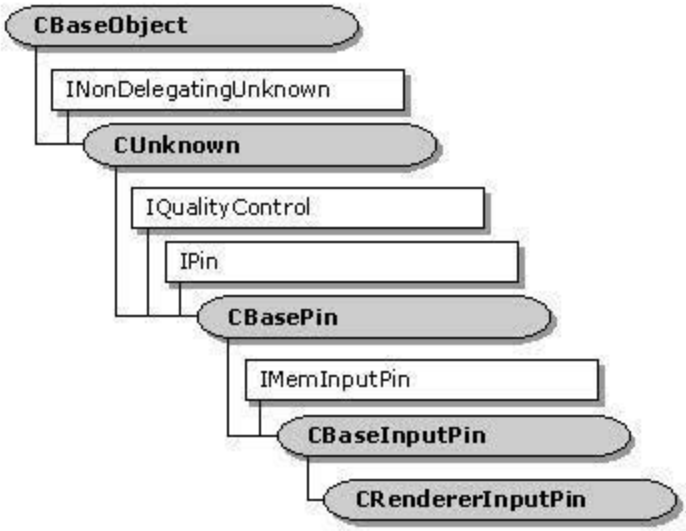
Return Values

Returns an HRESULT value (S_OK by default).

Remarks

This member function overrides the CBasePin::Run member function. It sends the EC_COMPLETE notification to the filter graph manager if m_bAtEndOfStream is TRUE.

CRendererInputPin Class



This input pin class channels calls to the rendering filter.

Protected Data Members

Name	Description
m_pRenderer	Pointer to the CBaseRenderer object.

Member Functions

Name	Description
Allocator	Retrieves a pointer to the default memory allocator.
CRendererInputPin	Constructs a CRendererInputPin object.

Overridable Member Functions

Name	Description
Active	Switches the pin to the active (paused or running) mode.
BeginFlush	Informs the pin to begin a flush operation.
BreakConnect	Adds customized code upon breaking a connection.
CheckMediaType	Determines if the pin can support a specific media type.
CompleteConnect	Completes the connection.
EndFlush	Informs the pin to end a flush operation.
EndOfStream	Informs the pin that no additional data is expected until a new run command is issued.
Inactive	Switches the pin to an inactive state and releases the memory of the allocator.
Receive	Returns the next block of data from the stream.

SetMediaType	Sets the media type of the pin.

Implemented IPin Methods

Name	Description
QueryId	Retrieves an identifier for the pin.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CRendererInputPin::Active

CRendererInputPin Class

Switches the pin to the active (paused or running) mode.

HRESULT Active( );

Return Values

Returns an HRESULT value.

Remarks

This member function overrides CBasePin::Active and calls the renderer filter's CBaseRenderer::Active member function.

Previous Home Topic Contents Index Next

CRendererInputPin::Allocator

CRendererInputPin Class

Retrieves a pointer to the default memory allocator inherited from CBaseInputPin.

IMemAllocator* Allocator( ) const;

Return Values

Returns a pointer to an IMemAllocator interface.

Remarks

The returned pointer is CBaseInputPin::m_pAllocator.

Previous Home Topic Contents Index Next

CRendererInputPin::BeginFlush

CRendererInputPin Class

Informs the pin to begin a flush operation.

HRESULT BeginFlush( );

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IPin::BeginFlush method. It overrides CBaseInputPin::BeginFlush and calls the renderer filter's BeginFlush member function before calling the base class implementation.

Previous Home Topic Contents Index Next

CRendererInputPin::BreakConnect

CRendererInputPin Class

Override this member function to add customized code upon breaking a connection.

HRESULT BreakConnect( );

Return Values

Returns an HRESULT value.

Remarks

This member function overrides CBasePin::BreakConnect and calls the renderer filter's BreakConnect member function before calling the base class implementation.

Previous Home Topic Contents Index Next

CRendererInputPin::CheckMediaType

CRendererInputPin Class

Override this member function to determine if the pin can support this specific media type.

HRESULT CheckMediaType(
const CMediaType *pmt*
);

Parameters

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

Return Values

Returns an HRESULT value.

Remarks

This member function is typically called before calling the CRendererInputPin::SetMediaType member function.

This member function overrides CBasePin::CheckMediaType and calls the pure virtual CBaseRenderer::CheckMediaType member function, which must be overridden.

Previous Home Topic Contents Index Next

CRendererInputPin::CompleteConnect

CRendererInputPin Class

Override this member function to inform the derived class when the connection process has completed.

HRESULT CompleteConnect(
IPin **pReceivePin*
);

Parameters

pReceivePin
Pointer to the connected (receiving) pin.

Return Values

Returns an HRESULT value.

Remarks

This member function overrides CBasePin::CompleteConnect and calls the renderer filter's CompleteConnect member function before calling the base class implementation.

Previous Home Topic Contents Index Next

CRendererInputPin::CRendererInputPin

CRendererInputPin Class

Constructs a CRendererInputPin object.

CRendererInputPin(
CBaseRenderer **pRenderer*,
HRESULT **phr*,
LPCWSTR *Name*
);

Parameters

pRenderer
Pointer to the rendering filter in the base class.
phr
Pointer to an HRESULT value.
Name

Pin name.

Return Values

No return value.

Previous Home Topic Contents Index Next

CRendererInputPin::EndFlush

CRendererInputPin Class

Informs the pin to end a flush operation.

HRESULT EndFlush(void);

Return Values

Returns an HRESULT value.

Remarks

This member function overrides CBaseInputPin::EndFlush and calls the renderer filter's EndFlush member function before calling the base class implementation.

Note that because this is a renderer, it does not pass the flush on downstream.

Previous Home Topic Contents Index Next

CRendererInputPin::EndOfStream

CRendererInputPin Class

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

HRESULT EndOfStream(void);

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IPin::EndOfStream method. It calls CheckStreaming to see that the filter is in a streaming state and then calls the CBaseRenderer::EndOfStream member function.

Previous Home Topic Contents Index Next

CRendererInputPin::Inactive

CRendererInputPin Class

Informs the pin that it is going into the inactive state.

HRESULT Inactive(void);

Return Values

Returns an HRESULT value.

Remarks

This member function overrides CBaseInputPin::Inactive. It calls the renderer filter's CBaseRenderer::Inactive member function, which returns NOERROR by default.

Previous Home Topic Contents Index Next

CRendererInputPin::QueryId

CRendererInputPin 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. It overrides the CBasePin::QueryId member function and assigns the string "In" to *Id*. Note that it uses the Microsoft® Win32® CoTaskMemAlloc function to initialize *Id*, so the user is responsible for freeing the format block by using CoTaskMemFree.

Previous Home Topic Contents Index Next

CRendererInputPin::Receive

CRendererInputPin Class

Returns the next block of data from the stream.

HRESULT Receive(
IMediaSample **pMediaSample*
);

Parameters

pMediaSample
Media sample.

Return Values

Returns an HRESULT value.

Remarks

This member function implements the IMemInputPin::Receive method, and it overrides the CBaseInputPin::Receive member function, which it calls to verify formats.

This is a blocking synchronous member function. It blocks and waits until it is time for the sample to be rendered. (It calls CBaseRenderer::Receive, which actually does the blocking.) Because only one sample is ever outstanding, this member function checks the media type and

calls CRendererInputPin::SetMediaType to change the pin's media type if the sample's type has changed.

Call the IUnknown::AddRef method if you must hold the returned data block beyond the completion of the **CRendererInputPin::Receive** member function. If you call AddRef, be sure to call the IUnknown::Release method upon completion of **AddRef**.

Previous Home Topic Contents Index Next

CRendererInputPin::SetMediaType

CRendererInputPin Class

Override this member function to set the media type of the pin.

HRESULT SetMediaType(
const CMediaType **pmt*
);

Parameters

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

Return Values

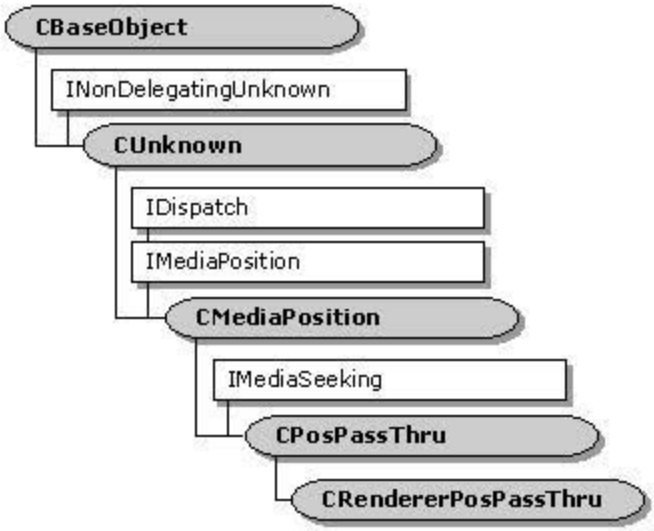
Returns an HRESULT value.

Remarks

This member function overrides CBasePin::SetMediaType and calls the renderer filter's SetMediaType member function, which returns NOERROR by default, after calling the base class implementation.

Previous Home Topic Contents Index Next

CRendererPosPassThru Class



The IMediaSeeking interface is used to seek to a specific sample, frame, or indexed field. These values are indicated by a whole number, such as frame 20 of a sequence of 530. However, when asked for a reference start or end time (in seconds), the sample must have this information previously set. The **CRendererPosPassThru** class, implemented on the video renderer, performs this service because the renderer is responsible for keeping track of reference time and stream time.

Member Functions

Name	Description
CRendererPosPassThru	Constructs a CRendererPosPassThru object.
GetMediaTime	Returns the media start and end times registered in the object.
RegisterMediaTime	Registers the media start and end times with the object.
ResetMediaTime	Resets the object's media start and end times.

Previous Home Topic Contents Index Next

CRendererPosPassThru::CRendererPosPassThru

CRendererPosPassThru Class

Constructs a CRendererPosPassThru object.

CRendererPosPassThru(
const TCHAR **pName*,
LPUNKNOWN *pUnk*,
HRESULT * *phr*,
IPin * *pPin*
);

Parameters

pName
Name of the object used in the CRendererPosPassThru constructor for debugging purposes.
pUnk
Pointer to the owner of this object.
phr
Pointer to an HRESULT value for resulting information.
pPin
Pointer to the input pin for the filter.

Return Values

No return value.

Remarks

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

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CRendererPosPassThru::GetMediaTime

CRendererPosPassThru Class

Retrieves the current media start and end times registered in the object.

HRESULT GetMediaTime(
LONGLONG* *pStartTime*,
LONGLONG* *pEndTime*
);

Parameters

pStartTime
Returned starting media time.
pEndTime
Returned ending media time.

Return Values

Returns an HRESULT value from the call to CPosPassThru::ConvertTimeFormat for the start and end times.

Remarks

This member function returns the media times set by the CRendererPosPassThru::RegisterMediaTime member function. The starting media time is always returned. Set *pEndTime* to a nonzero value to retrieve the ending media time.

Previous Home Topic Contents Index Next

CRendererPosPassThru::RegisterMediaTime

CRendererPosPassThru Class

Registers the media start and end times with the object.

```
HRESULT RegisterMediaTime(
  IMediaSample *pMediaSample
  ),

  HRESULT RegisterMediaTime(
  LONGLONG pStartTime,
  LONGLONG pEndTime
  );
```

Parameters

pMediaSample
IMediaSample object containing the media times.
pStartTime
Returned starting media time.
pEndTime
Returned ending media time.

Return Values

Returns VFW_E_MEDIA_TIME_NOT_SET if the sample does not have its media times set. Otherwise, returns and HRESULT from the call to IMediaSample::GetTime.

Previous Home Topic Contents Index Next

CRendererPosPassThru::ResetMediaTime

CRendererPosPassThru Class

Resets the object's media start and end times.

HRESULT ResetMediaTime(void);

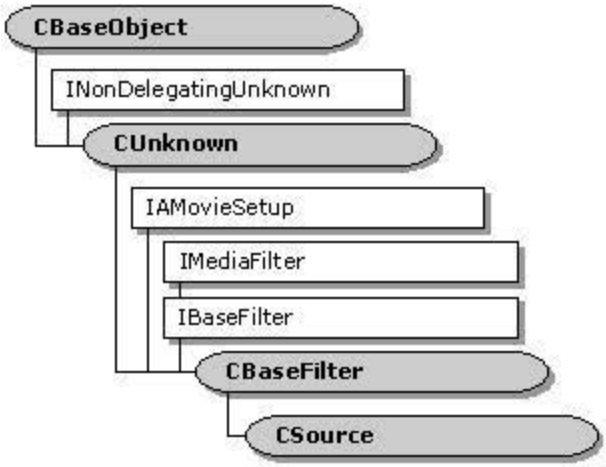
Return Values

Returns NOERROR.

Remarks

Sets the start and stop times to zero.

CSource Class



This class and its corresponding class, CSourceStream, simplify the construction of source filters that produce continuous streams of data comparable to the way the CTransformFilter class assists in the creation of transform filters.

The **CSource** class provides a wrapper for the CBaseFilter class that performs the pin management and works with the CSourceStream class to provide the pins.

To use the **CSource** class to build a filter:

- Derive your filter-level class from the **CSource** class. Provide a **CreateInstance** member function in it to create a new object of the class.
- Provide a means of adding objects that are derived from the CSourceStream class to support the output pins during construction of the class. You can either create them yourself during construction or provide the developer with a means of creating them later.

For an example of using the **CSource** class, see the Ball sample in the \Samples\DS\Ball directory of the Microsoft® DirectShow™ SDK Software Development Kit (SDK).

This class does not help build an asynchronous file reader source filter, which requires support of an IAsyncReader interface and a downstream parser filter that supports the CPullPin class.

Protected Data Members

Name	Description
m_cStateLock	Locks this data member to serialize access to the filter state.
m_iPins	Number of pins on this filter; updated by the CSource::AddPin and CSource::RemovePin member functions.
m_paStreams	Array of streams associated with this filter.

Member Functions

Name	Description
AddPin	Adds a pin to the source filter.
CSource	Constructs a CSource object.
FindPinNumber	Retrieves the number of the pin through the *IPin* parameter.
GetPin	Returns a pointer to a specified pin.
GetPinCount	Gets the number of pins contained by the filter.
pStateLock	Returns a pointer to the filter-critical section.
RemovePin	Removes a pin from the source filter.

Implemented IBaseFilter Methods

Name	Description
FindPin	Retrieves a pointer to the pin with the specified identifier.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CSource::AddPin

CSource Class

Adds a pin to the source filter.

HRESULT AddPin(
CSourceStream * *pStream*
);

Parameters

pStream
Pointer to the CSourceStream object associated with the pin.

Return Values

Returns S_OK if successful, or E_OUTOFMEMORY if no memory is available.

Previous Home Topic Contents Index Next

CSource::CSource

CSource Class

Initializes the CSource object.

CSource(
TCHAR **pName*,
LPUNKNOWN *lpunk*,
CLSID *clsid*
);

Parameters

pName
Debugging name of this object.
lpunk
Controlling IUnknown passed to the derived class's **CreateInstance** function.
clsid
Class identifier of the filters.

Return Values

No return value.

Remarks

The derived class could create the pins here, unless it provides a means for the developer to do this.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CSource::FindPinNumber

CSource Class

Retrieves the number of the pin supporting a given IPin interface.

int FindPinNumber(
IPin **iPin*

```
);
```

Parameters

iPin
IPin interface of the pin to retrieve.

Return Values

Returns the pin number or –1 if no matching pin number is found.

Previous Home Topic Contents Index Next

CSource::GetPin

CSource Class

Returns a pointer to the specified pin.

```
CBasePin *GetPin(
  int n
  );
```

Parameters

n
Pin number of the requested pin.

Return Values

Returns the pointer to the pin or NULL if the index is out of range.

Remarks

This member function is specified in CBaseFilter and is implemented here. Note that this pin interface will not have been reference counted when obtained.

Previous Home Topic Contents Index Next

CSource::GetPinCount

CSource Class

Retrieves the number of pins contained by the filter.

int GetPinCount(void);

Return Values

Returns the pin count.

Remarks

This member function is specified in CBaseFilter and is implemented here.

Previous Home Topic Contents Index Next

CSource::pStateLock

CSource Class

Retrieves a pointer to the filter-critical section.

CCritSec* pStateLock(void);

Return Values

Returns the critical section.

Remarks

Locking consists of holding the filter-critical section by calling the **pStateLock** member function and using the returned object to serialize access to functions. Typically, this lock can be held by a function when the worker thread might want to hold it. Therefore, to access a shared state from the worker thread, add another critical-section object. The exception occurs during the processing loop of the thread when it is safe to retrieve the filter-critical section from within CSourceStream::FillBuffer.

Previous Home Topic Contents Index Next

CSource::RemovePin

CSource Class

Removes a pin from the source filter.

HRESULT RemovePin(
CSourceStream * *pStream*
);

Parameters

pStream
CSourceStream object associated with the pin.

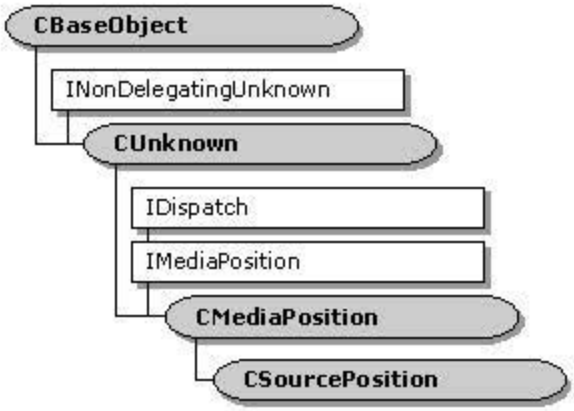
Return Values

Returns S_OK if successful or S_FALSE if unsuccessful.

Remarks

The *pStream* parameter is not deleted. This member function adjusts pin locations in the m_paStreams array.

CSourcePosition Class



CSourcePosition is an abstract class that assists source filters with the implementation of IMediaPosition methods.

Derive from this class and set the duration and default stop positions. This class supports IMediaPosition, and calls the pure virtual member functions CSourcePosition::ChangeStart, CSourcePosition::ChangeStop, and CSourcePosition::ChangeRate when CSourcePosition::put_CurrentPosition, CSourcePosition::put_StopTime, or CSourcePosition::put_Rate is called, to allow a source filter to handle these commands and start sending new data.

Override the CSourcePosition::ChangeStart, CSourcePosition::ChangeStop, and CSourcePosition::ChangeRate member functions to do something when the properties change.

Protected Data Members

Name	Description
m_Duration	Duration of the stream.
m_pLock	Pointer to a CCritSec object for locking.
m_Rate	Sample rate.
m_Start	Start time.
m_Stop	Stop time.

Member Functions

Name	Description
CSourcePosition	Constructs a CSourcePosition object.

Overridable Member Functions

Name	Description
ChangeRate	Override this pure virtual to handle notification that the rate property has changed.
ChangeStart	Override this pure virtual to handle notification that the start position property has changed.
ChangeStop	Override this pure virtual to handle notification that the stop position property has changed.

Implemented IMediaPosition Methods

Name	Description
get_CurrentPosition	Not currently implemented.
get_Duration	Retrieves the total duration of the media.
get_PrerollTime	Not currently implemented.
get_Rate	Retrieves the playback rate, relative to normal playback of the media.
get_StopTime	Retrieves the position within the media at which playback should stop.
put_CurrentPosition	Sets the position within the media at which playback should start.
put_PrerollTime	Not currently implemented.
put_Rate	Sets the playback rate, relative to normal playback of the media.
put_StopTime	Sets the position within the media at which playback should stop.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CSourcePosition::ChangeRate

CSourcePosition Class

Override this member function to handle notification of a change of sample rate.

virtual HRESULT ChangeRate( ) PURE;

Return Values

Returns an HRESULT value that depends on the implementation. **HRESULT** can be one of the following standard constants, or other values not listed:

Value	Meaning
E_FAIL	Failure.
E_POINTER	Null pointer argument.
E_INVALIDARG	Invalid argument.
E_NOTIMPL	Method isn't supported.
S_OK or NOERROR	Success.

Remarks

This member function is called when a change to the rate has been made by a call to IMediaPosition::put_Rate. Override this and change the rate of data sent; typically, this will be by a call to CBaseInputPin::BeginFlush and CBaseInputPin::EndFlush, and then sending samples marked with new time stamps.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CSourcePosition::ChangeStart

CSourcePosition Class

Override this member function to handle notification of a change of start time.

virtual HRESULT ChangeStart( ) PURE;

Return Values

Returns an HRESULT value that depends on the implementation. **HRESULT** can be one of the following standard constants, or other values not listed:

Value	Meaning
E_FAIL	Failure.
E_POINTER	Null pointer argument.
E_INVALIDARG	Invalid argument.
E_NOTIMPL	Method isn't supported.
S_OK or NOERROR	Success.

Remarks

This member function is called when a new start position has been requested by a call to IMediaPosition::put_CurrentPosition. Override this and change the data sent; typically, this will be by a call to CBaseInputPin::BeginFlush and CBaseInputPin::EndFlush, and then sending samples marked with new time stamps.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CSourcePosition::ChangeStop

CSourcePosition Class

Override this member function to handle notification of a change in stop time.

virtual HRESULT ChangeStop( ) PURE;

Return Values

Returns an HRESULT value that depends on the implementation. **HRESULT** can be one of the following standard constants, or other values not listed:

Value	Meaning
E_FAIL	Failure.
E_POINTER	Null pointer argument.
E_INVALIDARG	Invalid argument.
E_NOTIMPL	Method isn't supported.
S_OK or NOERROR	Success.

Remarks

This member function is called when a change to the stop position has been made by a call to IMediaPosition::put_StopTime. Override this and ensure that the correct stop time is being observed; typically, this will be a call to CBaseInputPin::BeginFlush and CBaseInputPin::EndFlush, and then resending data.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CSourcePosition::CSourcePosition

CSourcePosition Class

Constructs a CSourcePosition object.

CSourcePosition(
  const TCHAR * *pName*,
  LPUNKNOWN *pUnk*,

HRESULT * *phr*,
CCritSec * *pLock*
);

Parameters

pName
Name of the object used in the CSourcePosition constructor for debugging purposes.
pUnk
Pointer to the owner of this object.
phr
Pointer to an HRESULT value for resulting information.
pLock
Pointer to a CCritSec object used for locking.

Return Values

No return value.

Remarks

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

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CSourcePosition::get_CurrentPosition

CSourcePosition Class

Currently not implemented.

HRESULT get_CurrentPosition(
REFTIME* *pllTime*
);

Parameters

pllTime
Returned start time as a double value in seconds.

Return Values

Returns E_NOTIMPL.

Remarks

Override this method if you can return the data you are actually working on. The start position is applied before the rate and therefore is the position at typical playback speed.

Previous Home Topic Contents Index Next

CSourcePosition::get_Duration

CSourcePosition Class

Retrieves the total duration of the media stream.

```
HRESULT get_Duration(
  REFTIME* plength
  );
```

Parameters

plength
Returned length of the media stream.

Return Values

Returns E_POINTER if *pLength* is invalid. Otherwise, returns S_OK.

Remarks

The duration assumes normal playback speed; it is therefore unaffected by the rate.

Previous Home Topic Contents Index Next

CSourcePosition::get_PrerollTime

CSourcePosition Class

Validates the pointer, but the preroll retrieval is not currently implemented.

HRESULT get_PrerollTime(
REFTIME* *pllTime*
);

Parameters

pllTime
Returned preroll time as a double value in seconds.

Return Values

Returns E_POINTER if *pllTime* is invalid. Otherwise, returns E_NOTIMPL.

Remarks

Preroll time is the time prior to the start position at which nonrandom access devices, such as tape players, should start rolling.

Previous Home Topic Contents Index Next

CSourcePosition::get_Rate

CSourcePosition Class

Retrieves the rate of playback relative to normal playback speed.

HRESULT get_Rate(
double * *pdRate*
);

Parameters

pdRate
Returned rate.

Return Values

Returns E_POINTER if *pdRate* is invalid. Otherwise, returns S_OK.

Remarks

A rate of 1.0 indicates normal playback speed. A rate of 0.5 indicates half speed. A rate of –1.0 indicates normal speed in reverse.

Previous Home Topic Contents Index Next

CSourcePosition::get_StopTime

CSourcePosition Class

Retrieves the time at which the media stream stops.

HRESULT get_StopTime(
REFTIME* *pllTime*
);

Parameters

pllTime
Returned stop time as a double value in seconds.

Return Values

Returns E_POINTER if *pllTime* is invalid. Otherwise, returns S_OK.

Remarks

The stop time is a position between zero and the duration of the media at which playback should stop.

The stop position is applied before the rate and therefore is the position at typical playback speed.

Previous Home Topic Contents Index Next

CSourcePosition::put_CurrentPosition

CSourcePosition Class

Sets the time within the media stream that playback should begin.

HRESULT put_CurrentPosition(
REFTIME *llTime*
);

Parameters

llTime
Start time expressed as a double value in seconds.

Return Values

Returns an HRESULT value from the call to CSourcePosition::ChangeStart.

Remarks

The start time is a position between zero and the duration of the media at which playback should begin when the next run command is issued.

Setting the start position when paused causes playback to resume from the new start position when the run command is issued.

The start position is applied before the rate and therefore is the position at typical playback speed.

Previous Home Topic Contents Index Next

CSourcePosition::put_PrerollTime

CSourcePosition Class

Not currently implemented.

```
HRESULT put_PrerollTime(
  REFTIME llTime
  );
```

Parameters

llTime
Preroll time to be set.

Return Values

Returns E_NOTIMPL.

Remarks

Preroll time is the time prior to the start position at which nonrandom access devices, such as tape players, should start rolling.

CSourcePosition::put_Rate

CSourcePosition Class

Sets the rate of playback relative to normal speed.

```
HRESULT put_Rate(
  double dRate
  );
```

Parameters

dRate
Rate to set.

Return Values

Returns an HRESULT value from the call to CSourcePosition::ChangeRate.

Remarks

This property allows an application to speed up or slow down playback relative to the normal default playback speed. A rate of 1.0 indicates normal playback speed. Specifying 2.0 causes playback at twice the normal rate: a video created for 10 frames per second (fps) will be played back at 20 fps, if resources permit. Audio streams played back at above-normal speed increase the pitch rather than drop frames.

Negative rates indicate reverse play. Not all media will support reverse play.

CSourcePosition::put_StopTime

CSourcePosition Class

Sets the time at which the media stream will stop.

HRESULT put_StopTime(
REFTIME *llTime*
);

Parameters

llTime
Stop time as a double value in seconds.

Return Values

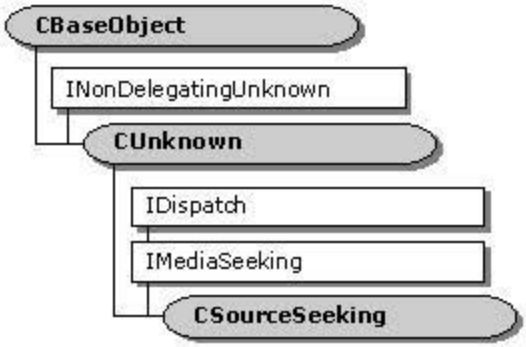
Returns an HRESULT value from the call to CSourcePosition::ChangeStop.

Remarks

The stop time is a position between zero and the duration of the media at which playback should stop.

The stop position is applied before the rate and therefore is the position at typical playback speed.

CSourceSeeking Class



CSourceSeeking is an abstract class that assists source filters with the implementation of IMediaSeeking interface methods. This class enables a source filter to handle calls that change the start and stop positions in the media stream and the playback rate.

Derive from this class and set the positions. This class supports IMediaSeeking, and calls the pure virtual member functions CSourceSeeking::ChangeStart, CSourceSeeking::ChangeStop, and CSourceSeeking::ChangeRate when CSourceSeeking::SetPositions or CSourceSeeking::SetRate is called, to enable a source filter to handle these commands and start sending new data.

Override the CSourceSeeking::ChangeStart, CSourceSeeking::ChangeStop, and CSourceSeeking::ChangeRate member functions to do something when the properties change.

Protected Data Members

Name	Description
m_dRateSeeking	Playback rate. Set to 1 by default.
m_dwSeekingCaps	Seeking capabilities returned in the GetCapabilities function. Can be one or more of the following values: AM_SEEKING_CanSeekForwards, AM_SEEKING_CanSeekBackwards, AM_SEEKING_CanSeekAbsolute, AM_SEEKING_CanGetStopPos, AM_SEEKING_CanGetDuration. Set to all of these by default.
m_pLock	Pointer to a CCritSec object for locking.
m_rtDuration	Duration of the stream. Set to m_rtStop by default.
m_rtStart	Start time. Set to zero by default.
m_rtStop	Stop time. Set to the largest positive 64-bit integer possible (9223372036854775807) by default.

Member Functions

Name	Description
CSourceSeeking	Constructs a CSourceSeeking object.

Overridable Member Functions

Name	Description
ChangeRate	Override this pure virtual to handle notification of a change of sample rate.
ChangeStart	Override this pure virtual to handle notification of a change of start time.
ChangeStop	Override this pure virtual to handle notification of a change in stop time.

Implemented IMediaSeeking Methods

Name	Description
CheckCapabilities	Checks that all requested capabilities are in m_dwSeekingCaps.
ConvertTimeFormat	Checks that the time format is TIME_FORMAT_MEDIA_TIME. This is the only format currently available.
GetAvailable	Retrieves the range of seeking times. Earliest is zero and latest is the media stream's duration.
GetCapabilities	Retrieves the current seeking capabilities in m_dwSeekingCaps.
GetCurrentPosition	Not currently implemented.
GetDuration	Retrieves the length of time the media stream will play.
GetPositions	Retrieves the current start and stop position settings.
GetPreroll	Sets the preroll time to zero.
GetRate	Retrieves the current playback rate.
GetStopPosition	Retrieves the position within the media stream at which playback should stop.
GetTimeFormat	Sets the time format to TIME_FORMAT_MEDIA_TIME. This is the only format currently supported.
IsFormatSupported	Determines if the requested format is TIME_FORMAT_MEDIA_TIME. This is the only format currently supported.
IsUsingTimeFormat	Determines if the requested format is TIME_FORMAT_MEDIA_TIME. This is the only format currently supported.
QueryPreferredFormat	Sets the preferred time format to TIME_FORMAT_MEDIA_TIME. This is the only format currently supported.
SetPositions	Sets current and stop positions, first checking that the seeking options are valid.
SetRate	Sets the playback rate.
SetTimeFormat	Checks that the time format is TIME_FORMAT_MEDIA_TIME. This is the only format currently supported.

Implemented INonDelegatingUnknown Methods

Name	Description
NonDelegatingQueryInterface	Retrieves an interface and increments the reference count on the interface.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CSourceSeeking::ChangeRate

CSourceSeeking Class

Override this member function to handle notification of a change of sample rate.

virtual HRESULT ChangeRate( ) PURE;

Return Values

Returns an HRESULT value that depends on the implementation. **HRESULT** can be one of the following standard constants, or other values not listed:

Value	Meaning
E_FAIL	Failure.
E_POINTER	Null pointer argument.
E_INVALIDARG	Invalid argument.
E_NOTIMPL	Method isn't supported.
S_OK or NOERROR	Success.

Remarks

This member function is called when a change to the rate has been made by a call to the CSourceSeeking::SetRate function. Override this and change the rate of data sent. Typically, you do this by calling CBaseInputPin::BeginFlush and CBaseInputPin::EndFlush, and then send samples marked with new time stamps, for example, with an implementation NewSegment method.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CSourceSeeking::ChangeStart

CSourceSeeking Class

Override this member function to handle notification of a change of start time.

virtual HRESULT ChangeStart( ) PURE;

Return Values

Returns an HRESULT value that depends on the implementation. **HRESULT** can be one of the following standard constants, or other values not listed:

Value	Meaning
E_FAIL	Failure.
E_POINTER	Null pointer argument.
E_INVALIDARG	Invalid argument.
E_NOTIMPL	Method isn't supported.
S_OK or NOERROR	Success.

Remarks

This member function is called when a new start position has been requested by a call to CSourceSeeking::SetPositions. Override this and change the data sent. Typically, you do this by calling CBaseInputPin::BeginFlush and CBaseInputPin::EndFlush, and then send samples marked with new start time, for example, with an implementation NewSegment method.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CSourceSeeking::ChangeStop

CSourceSeeking Class

Override this member function to handle notification of a change in stop time.

virtual HRESULT ChangeStop( ) PURE;

Return Values

Returns an HRESULT value that depends on the implementation. **HRESULT** can be one of the following standard constants, or other values not listed:

Value	Meaning
E_FAIL	Failure.
E_POINTER	Null pointer argument.
E_INVALIDARG	Invalid argument.
E_NOTIMPL	Method isn't supported.
S_OK or NOERROR	Success.

Remarks

This member function is called when a change to the stop position has been made by a call to

CSourceSeeking::SetPositions. Override this and ensure that the correct stop time is being observed. Typically, you do this by calling CBaseInputPin::BeginFlush and CBaseInputPin::EndFlush, and then send samples marked with new stop time, for example, with an implementation NewSegment method.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CSourceSeeking::CheckCapabilities

CSourceSeeking Class

Checks that all the requested capabilities are among the flags in m_dwSeekingCaps.

HRESULT CheckCapabilities(
DWORD * *pCapabilities*
);

Parameters

pCapabilities
Pointer to an **AM_SEEKING_CAPABILITIES** enumerator containing the desired seeking capabilities flags in m_dwSeekingCaps. This value can be any combination of the following flags:

AM_SEEKING_CanGetCurrentPos
AM_SEEKING_CanGetDuration
AM_SEEKING_CanGetStopPos
AM_SEEKING_CanPlayBackwards
AM_SEEKING_CanSeekAbsolute
AM_SEEKING_CanSeekBackwards
AM_SEEKING_CanSeekForwards

Return Values

Returns E_POINTER if *pCapabilities* is not a valid pointer, S_OK if all the requested capabilities in *pCapabilities* are supported, or S_FALSE if they are not.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CSourceSeeking::ConvertTimeFormat

CSourceSeeking Class

Checks that the time format is TIME_FORMAT_MEDIA_TIME. This is the only format currently available.

HRESULT ConvertTimeFormat(
LONGLONG * *pTarget*,
const GUID * *pTargetFormat*,
LONGLONG *Source*,
const GUID * *pSourceFormat*
);

Parameters

pTarget
Time set to *Source* time if the format is TIME_FORMAT_MEDIA_TIME or NULL.
pTargetFormat
GUID of the TIME_FORMAT_MEDIA_TIME format, or NULL.
Source
Time in original format.
pSourceFormat
GUID of the TIME_FORMAT_MEDIA_TIME format, or NULL.

Return Values

Returns E_POINTER if *pTarget* is not a valid pointer, or E_INVALIDARG if *pTargetFormat* and *pSourceFormat* are not equal to TIME_FORMAT_MEDIA_TIME or NULL; otherwise, returns S_OK.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CSourceSeeking::CSourceSeeking

CSourceSeeking Class

Constructs a CSourceSeeking object.

CSourceSeeking(
const TCHAR * *pName*,
LPUNKNOWN *pUnk*,
HRESULT * *phr*,
CCritSec * *pLock*
);

Parameters

pName
Name of the object used in the CSourceSeeking constructor for debugging purposes.
pUnk
Pointer to the owner of this object.
phr
Pointer to an HRESULT value for information about the results of creating this object.
pLock
Pointer to a CCritSec object used for synchronization within a process.

Return Values

No return value.

Remarks

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

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CSourceSeeking::GetAvailable

CSourceSeeking Class

Returns the range of seeking times.

HRESULT GetAvailable(
LONGLONG * *pEarliest*,
LONGLONG * *pLatest*
);

Parameters

pEarliest

Earliest time that can be seeked to. Set to zero.

pLatest

Latest time that can be seeked to. Set to m_rtDuration.

Return Values

Returns S_OK.

Previous Home Topic Contents Index Next

CSourceSeeking::GetCapabilities

CSourceSeeking Class

Retrieves the seeking capabilities of the media stream.

HRESULT GetCapabilities(
DWORD * *pCapabilities*
);

Parameters

pCapabilities

Set to the seeking capability flags in m_dwSeekingCaps, which can be any combination of the following:

AM_SEEKING_CanGetCurrentPos
AM_SEEKING_CanGetDuration
AM_SEEKING_CanGetStopPos
AM_SEEKING_CanPlayBackwards
AM_SEEKING_CanSeekAbsolute
AM_SEEKING_CanSeekBackwards
AM_SEEKING_CanSeekForwards

Return Values

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

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CSourceSeeking::GetCurrentPosition

CSourceSeeking Class

Not currently implemented.

HRESULT GetCurrentPosition(
LONGLONG* *pCurrent*
);

Parameters

pCurrent
Current position in current time format units.

Return Values

Returns E_NOTIMPL.

Remarks

This function is typically supported only in renderers and not in source filters.

Previous Home Topic Contents Index Next

CSourceSeeking::GetDuration

CSourceSeeking Class

Retrieves the length of time that the media stream will play.

HRESULT GetDuration(
LONGLONG* *pDuration*
);

Parameters

pDuration
Duration of the media stream set to the value in m_rtDuration.

Return Values

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

Remarks

The duration in m_rtDuration is set to the stop time in m_rtStop. Set the stop time with the CSourceSeeking::SetPositions function.

Previous Home Topic Contents Index Next

CSourceSeeking::GetPositions

CSourceSeeking Class

Retrieves the current and stop position settings.

```
HRESULT GetPositions(
  LONGLONG * pCurrent,
  LONGLONG * pStop
  );
```

Parameters

pCurrent
Current start time set to the value in m_rtStart.
pStop
Current stop time set to the value in m_rtStop.

Return Values

Returns S_OK.

Remarks

The start and stop times are set in the CSourceSeeking::SetPositions function.

Previous Home Topic Contents Index Next

CSourceSeeking::GetPreroll

CSourceSeeking Class

Sets the preroll time to zero.

HRESULT GetPreroll(
LONGLONG * *pPreroll*
);

Parameters

pPreroll
Returned preroll time of zero.

Return Values

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

Previous Home Topic Contents Index Next

CSourceSeeking::GetRate

CSourceSeeking Class

Retrieves the current playback rate.

HRESULT GetRate(
double * *pdRate*
);

Parameters

pdRate
Returned playback rate set to the value in m_dRateSeeking, where 1 is the normal rate.

Return Values

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

Remarks

Set the rate in the CSourceSeeking::SetRate function.

Previous Home Topic Contents Index Next

CSourceSeeking::GetStopPosition

CSourceSeeking Class

Retrieves the position within the media stream at which playback should stop.

HRESULT GetStopPosition(
LONGLONG* *pStop*
);

Parameters

pStop
Returned stop time set to the value in m_rtStop.

Return Values

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

Remarks

Set the stop time in the CSourceSeeking::SetPositions function.

Previous Home Topic Contents Index Next

CSourceSeeking::GetTimeFormat

CSourceSeeking Class

Sets the time format to TIME_FORMAT_MEDIA_TIME, which determines the format of units used during seeking.

HRESULT GetTimeFormat(
const GUID * *pFormat*
);

Parameters

pFormat
Media time format set to TIME_FORMAT_MEDIA_TIME.

Return Values

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

Remarks

TIME_FORMAT_MEDIA_TIME is the only time format currently supported.

Previous Home Topic Contents Index Next

CSourceSeeking::IsFormatSupported

CSourceSeeking Class

Determines if the requested format is TIME_FORMAT_MEDIA_TIME.

HRESULT IsFormatSupported(
const GUID * *pFormat*
);

Parameters

pFormat
Time format to compare to TIME_FORMAT_MEDIA_TIME.

Return Values

Returns E_POINTER if *pFormat* is invalid, S_OK if the format in *pFormat* is TIME_FORMAT_MEDIA_TIME, or S_FALSE if the format in *pFormat* is not TIME_FORMAT_MEDIA_TIME.

Remarks

TIME_FORMAT_MEDIA_TIME is the only time format currently supported. As implemented, this function is the same as CSourceSeeking::IsUsingTimeFormat.

Previous Home Topic Contents Index Next

CSourceSeeking::IsUsingTimeFormat

CSourceSeeking Class

Determines if the requested format is TIME_FORMAT_MEDIA_TIME.

HRESULT IsUsingTimeFormat(
const GUID * *pFormat*
);

Parameters

pFormat
Time format to compare to TIME_FORMAT_MEDIA_TIME.

Return Values

Returns E_POINTER if *pFormat* is invalid, S_OK if the format in *pFormat* is TIME_FORMAT_MEDIA_TIME, or S_FALSE if the format in *pFormat* is not TIME_FORMAT_MEDIA_TIME.

Remarks

TIME_FORMAT_MEDIA_TIME is the only time format currently supported. As implemented, this method is the same as CSourceSeeking::IsFormatSupported.

‹Previous Home Topic Contents Index Next›

CSourceSeeking::NonDelegatingQueryInterface

CSourceSeeking Class

Retrieves an interface and increments the reference count on the interface.

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

Parameters

riid
Reference identifier.

ppv
Pointer to the interface.

Return Values

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

Remarks

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

This member function implements the INonDelegatingUnknown::NonDelegatingQueryInterface method.

Previous Home Topic Contents Index Next

CSourceSeeking::QueryPreferredFormat

CSourceSeeking Class

Sets the preferred time format to TIME_FORMAT_MEDIA_TIME.

HRESULT QueryPreferredFormat(
GUID **pFormat*
);

Parameters

pFormat
Time format set to TIME_FORMAT_MEDIA_TIME.

Return Values

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

Remarks

TIME_FORMAT_MEDIA_TIME is the only time format currently supported.

Previous Home Topic Contents Index Next

CSourceSeeking::SetPositions

CSourceSeeking Class

Sets current and stop positions, first checking that the seeking options are valid.

HRESULT SetPositions(
LONGLONG * *pCurrent*,
DWORD *CurrentFlags*,
LONGLONG * *pStop*,
DWORD *StopFlags*
);

Parameters

pCurrent
Start position if stopped, or position to continue from if paused.
CurrentFlags
Flags that indicate the type of seek. Valid values are AM_SEEKING_AbsolutePositioning and AM_SEEKING_RelativePositioning. See the IMediaSeeking::SetPositions method for a description of these flags.
pStop
Position in the stream at which to quit playback.
StopFlags
Flags that indicate stop position seeking options. Valid values are AM_SEEKING_AbsolutePositioning, AM_SEEKING_RelativePositioning, and AM_SEEKING_IncrementalPositioning. See the IMediaSeeking::SetPositions method for a description of these flags.

Return Values

Returns E_INVALIDARG if *CurrentFlags* and *StopFlags* are not one of the values listed, or E_POINTER if *pCurrent* or *pStop* is invalid; otherwise, returns the HRESULT returned by calls to the CSourceSeeking::ChangeStart and CSourceSeeking::ChangeStop functions.

Remarks

You must implement ChangeStart and ChangeStop to use this method.

See Also

CSourceSeeking::GetPositions, CSourceSeeking::GetStopPosition, CSourceSeeking::GetDuration

Previous Home Topic Contents Index Next

CSourceSeeking::SetRate

CSourceSeeking Class

Sets a new playback rate.

HRESULT SetRate(
double *dRate*
);

Parameters

dRate
New rate, where 1.0 is the normal normal playback speed. Specifying 2.0 causes playback at twice the normal rate: a video created for 10 frames per second (fps) will be played back at 20 fps, if resources permit. Audio streams played back at above-normal speed increase the pitch rather than drop samples. A rate of 0.5 specifies half speed.

Return Values

Returns the HRESULT value returned by the call to the CSourceSeeking::ChangeRate function.

Remarks

You must implement ChangeRate to use this method. The m_dRateSeeking data member is set to the new rate. Setting the rate to zero causes an error.

See Also

CSourceSeeking::GetRate

Previous Home Topic Contents Index Next

CSourceSeeking::SetTimeFormat

CSourceSeeking Class

Checks that the requested format is TIME_FORMAT_MEDIA_TIME.

HRESULT SetTimeFormat(
const GUID * *pFormat*
);

Parameters

pFormat
Time format to compare to TIME_FORMAT_MEDIA_TIME.

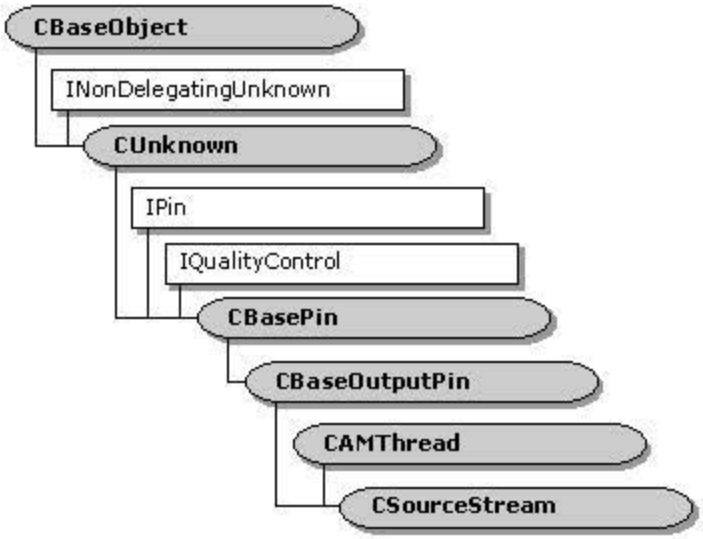
Return Values

Returns E_POINTER if *pFormat* is invalid, S_OK if the format in *pFormat* is TIME_FORMAT_MEDIA_TIME, or E_INVALIDARG if the format in *pFormat* is not TIME_FORMAT_MEDIA_TIME.

Remarks

TIME_FORMAT_MEDIA_TIME is the only time format currently supported.

CSourceStream Class



Derive from this class to provide a class that creates the data stream from one of the output pins. It should be used with an object that is derived from the CSource class derived object to provide the filter-level object.

The **CSourceStream** class creates a worker thread to push data downstream when the filter enters a paused or running state. The thread first calls the CSourceStream::OnThreadCreate member function. If this succeeds, it will loop, calling the CSourceStream::FillBuffer member function until the CSourceStream::Inactive member function stops it. As the thread quits, it calls the CSourceStream::OnThreadDestroy member function. If OnThreadCreate fails, OnThreadDestroy is called, and the active member function will fail.

To use the **CSourceStream** class, supporting a single media type, carry out the following steps.

1. Override the CSourceStream::GetMediaType member function to report the supported output format.
2. Override the CSourceStream::FillBuffer member function with a means of filling out each IMediaSample object with data.

To use the **CSourceStream** class, supporting multiple media types, carry out the following.

1. Override the CSourceStream::CheckMediaType and CSourceStream::GetMediaType member functions to report the supported media types (for more information, see the CBaseMediaFilter class).
2. Override the CSourceStream::FillBuffer member function with a means of filling out each IMediaSample object with data.

See SAMPLES\DS\BALL in the Microsoft® DirectShow™ SDK Software Development Kit (SDK)

for an example of a pin supporting multiple types.

If you want more complex management of your worker thread, you can override most of the associated member functions. See Samples\DS\Vidcap in the Microsoft DirectX Media Software Development Kit (SDK) for an example.

Member Functions

Name	Description
Active	Called by the CBaseMediaFilter class to start the worker thread.
CheckRequest	Determines if a command is waiting for the thread.
CSourceStream	Constructs a CSourceStream object.
Exit	Called by the CSourceStream::Inactive member function to exit the worker thread.
GetRequest	Retrieves the next command for the thread.
Inactive	Called by the CBaseMediaFilter member function to shut down the worker thread.
Init	Called by the CSourceStream::Active member function to initialize the worker thread.
Pause	Pauses the stream of the worker thread. This will acquire all necessary resources.
Run	Starts the worker thread generation of a media sample stream.
Stop	Stops the stream.

Overridable Member Functions

Name	Description
CheckMediaType	Determines if a specific media type is supported. Override this member function if you use multiple types.
DoBufferProcessingLoop	Loops, collecting a buffer and calling the CSourceStream::FillBuffer processing function.
FillBuffer	Override this member function to fill the stream buffer during the creation of a media sample.
GetMediaType	Retrieves the media type or types that this pin supports; override the appropriate version of this member function to support one or multiple media types.
OnThreadCreate	Called as the worker thread is created; override this member function for special processing.
OnThreadDestroy	Called during the destruction of a worker thread; override this member function for special processing.
OnThreadStartPlay	Called at the start of processing Pause or Run command; override this member function for special processing.
ThreadProc	Override this member function to create a custom thread procedure.

Implemented IPin Methods

Name	Description
QueryId	Retrieves an identifier for the pin.

Previous Home Topic Contents Index Next

CSourceStream::Active

CSourceStream Class

Starts the worker thread.

HRESULT Active(void);

Return Values

Returns an HRESULT, which can be one of the following:

Value	Meaning
E_FAIL	Thread could not start.
S_FALSE	Pin is already active.
S_OK	Thread was started successfully.

Previous Home Topic Contents Index Next

CSourceStream::CheckMediaType

CSourceStream Class

Determines if this pin supports the supplied media type.

virtual HRESULT CheckMediaType(
CMediaType *pMediaType*
);

Parameters

pMediaType
Media type to check.

Return Values

Returns S_OK if the media type is supported, S_FALSE if it isn't, or E_INVALIDARG if *pMediaType* is invalid.

Remarks

Override this member function if you support multiple media types. Test explicitly for S_OK to determine if this function succeeded; do not use the SUCCEEDED macro.

Previous Home Topic Contents Index Next

CSourceStream::CheckRequest

CSourceStream Class

Determines if a command is waiting for the thread.

```
BOOL CheckRequest(
  Command *pCom
  );
```

Parameters

pCom
Pointer to the location to which to return a command, if any.

Return Values

Returns TRUE if the *pCom* parameter contains a command; otherwise, returns FALSE.

Remarks

This member function does not block. This is a type safe override of the method in the CAMThread class.

Previous Home Topic Contents Index Next

CSourceStream::CSourceStream

CSourceStream Class

Creates a CSourceStream object.

CSourceStream(
TCHAR **pObjectName*,
HRESULT **phr*,
CSource **pms*,
LPCWSTR *pName*
);

Parameters

pObjectName
Name of the object.
phr
Resulting value for this constructor.
pms
Pointer for the CSource object.
pName
Name of the pin.

Return Values

No return value.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CSourceStream::DoBufferProcessingLoop

CSourceStream Class

Loops, collecting a buffer and calling the CSourceStream::FillBuffer processing function.

virtual HRESULT DoBufferProcessingLoop(void);

Return Values

Returns an HRESULT value.

Previous Home Topic Contents Index Next

CSourceStream::Exit

CSourceStream Class

Causes the thread to exit.

HRESULT Exit(void);

Return Values

Returns NOERROR if the member function was received.

Remarks

If the thread returns an error, it sets the return value of the CSourceStream::ThreadProc member function.

Previous Home Topic Contents Index Next

CSourceStream::FillBuffer

CSourceStream Class

Fills the stream buffer during the creation of a media sample that the current pin provides.

virtual HRESULT FillBuffer(
IMediaSample *pSample*
) PURE;

Parameters

pSample
IMediaSample buffer to contain the data.

Return Values

Returns an HRESULT value.

Remarks

The CSourceStream::ThreadProc member function calls the **CSourceStream::FillBuffer** member function. The derived class must supply an implementation of this member function.

Previous Home Topic Contents Index Next

CSourceStream::GetMediaType

CSourceStream Class

Fills out the fields of the CMediaType object to the supported media type.

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

virtual HRESULT GetMediaType(
CMediaType **pMediaType*
);

Parameters

iPosition
Position of the media type within a list of multiple media types. Range is zero through *n*.
pMediaType
Pointer to a CMediaType object to be set to the requested format.

Return Values

Returns one of the following HRESULT values.

Value	Meaning
Error Code	Media type could not be set.
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 member function sets the requested media type. If only a single media type is supported, override this member function with the single-parameter definition. Only the default implementations of the CSourceStream::CheckMediaType and **CSourceStream::GetMediaType** member functions call the single media type member function.

Override the single-version **GetMediaType** or the two-parameter version, CheckMediaType.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CSourceStream::GetRequest

CSourceStream Class

Retrieves the next command for the thread.

Command GetRequest(void);

Return Values

Returns the next command.

Remarks

This member function blocks until a command is available. It is a type safe override of the member function in the CAMThread class.

Previous Home Topic Contents Index Next

CSourceStream::Inactive

CSourceStream Class

Identifies a pin as inactive and shuts down the worker thread.

HRESULT Inactive(void);

Return Values

Returns an HRESULT value, including the following values.

Value	Meaning
S_OK	Thread exited successfully.
S_FALSE	Pin is already inactive.

Previous Home Topic Contents Index Next

CSourceStream::Init

CSourceStream Class

Initializes the worker thread.

HRESULT Init(void);

Return Values

Returns an HRESULT value, including the following values.

Value	Meaning
S_OK	Thread was initialized successfully.
S_FALSE	Thread was already initialized.

Remarks

The CSourceStream::Active member function calls this member function.

Previous Home Topic Contents Index Next

CSourceStream::OnThreadCreate

CSourceStream Class

Starts or stops a process upon the creation of a thread.

virtual HRESULT OnThreadCreate(void);

Return Values

Returns an HRESULT value, including the following values.

Value	Meaning
NOERROR	No error occurred.
Error code	Thread should exit.

Previous Home Topic Contents Index Next

CSourceStream::OnThreadDestroy

CSourceStream Class

Starts or stops a process upon the destruction of a thread.

virtual HRESULT OnThreadDestroy(void);

Return Values

Returns either NOERROR or an HRESULT value (greater than zero) that indicates an error.

Previous Home Topic Contents Index Next

CSourceStream::OnThreadStartPlay

CSourceStream Class

Starts a process upon the beginning of the playing of the thread.

virtual HRESULT OnThreadStartPlay(void);

Return Values

Default implementation returns NOERROR.

Previous Home Topic Contents Index Next

CSourceStream::Pause

CSourceStream Class

Pauses a media sample stream.

HRESULT Pause(void);

Return Values

Returns an HRESULT value, including the following value.

Value	Meaning
S_OK	Thread paused successfully.

Previous Home Topic Contents Index Next

CSourceStream::Run

CSourceStream Class

Starts a media sample stream.

HRESULT Run(void);

Return Values

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

Previous Home Topic Contents Index Next

CSourceStream::Stop

CSourceStream Class

Stops a media sample stream.

HRESULT Stop(void);

Return Values

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

CSourceStream::ThreadProc

CSourceStream Class

Implements the thread procedure.

virtual DWORD ThreadProc(void);

Return Values

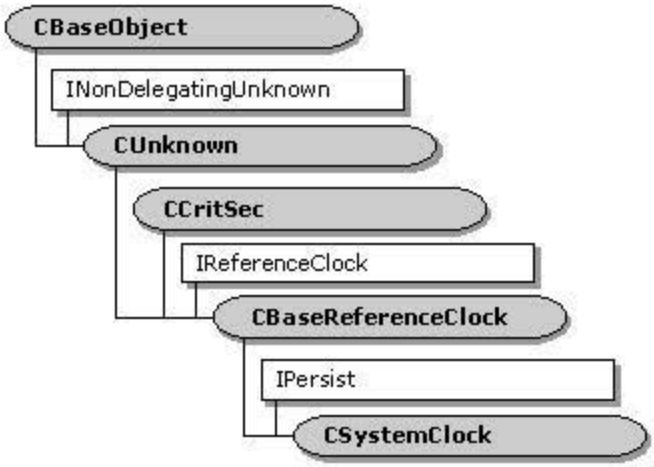
Returns 0 if the thread completed successfully and 1 otherwise. If 1, the thread's resources might still be allocated.

Remarks

When this member function returns, the thread exits. Override this member function if the provided version is not sophisticated enough.

Previous Home Topic Contents Index Next

CSystemClock Class



The **CSystemClock** class implements a system clock that provides time information and timing signals to an application. It uses the CBaseReferenceClock base class to provide most of that functionality, adding persistence.

CSystemClock implements the IPersist interface. For more implementation information, see "OLE Programmers Reference (Vol. 1): Structured Storage Overview."

Member Functions

Name	Description
CreateInstance	Creates an instance of a system clock.
CSystemClock	Constructs a CSystemClock object.

Implemented IPersist Methods

Name	Description
GetClassID	Returns the class identifier for this clock (CLSID_SystemClock).

Implemented INonDelegatingUnknown Methods

Name	Description
NonDelegatingQueryInterface	Passes out pointers to IID_IPersist, and calls the base clock for other interface queries.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CSystemClock::CreateInstance

CSystemClock Class

Creates a new instance when placed in the factory template table.

```
static CUnknown * WINAPI CreateInstance(
  LPUNKNOWN pUnk,
  HRESULT *phr
  );
```

Parameters

pUnk
Pointer to LPUNKNOWN.
phr
Pointer to an HRESULT value in which to return resulting information.

Return Values

Returns a pointer to a new Component Object Model (COM) object.

Remarks

This member function is required to create objects using the class factory. It calls the class constructor.

The *phr* parameter will be modified only if a failure occurs. If it is a failure code on input, construction can be terminated, but in any case the destructor will be called by the creator when the HRESULT error is detected.

CSystemClock::CSystemClock

CSystemClock Class

Constructs a CSystemClock object.

```
CSystemClock(
  TCHAR *pName,
```

LPUNKNOWN *pUnk*,
HRESULT **phr*
);

Parameters

pName
Name of this object (used for debugging only).
pUnk
Pointer to the controlling IUnknown interface.
phr
Pointer to the HRESULT value that will be set if an error occurs.

Return Values

No return value.

Remarks

If *phr* points to something other than S_OK upon entry, the object will not be constructed. If an error occurs during construction, this variable will be set; otherwise, its contents will not be altered.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CSystemClock::GetClassID

CSystemClock Class

Retrieves the class identifier for this clock.

HRESULT GetClassID(
CLSID **pClsID*
);

Parameters

pClsID
Pointer to a CLSID structure which is filled with CLSID_SystemClock.

Return Values

Returns an HRESULT value.

CSystemClock::NonDelegatingQueryInterface

CSystemClock Class

Returns an interface and increments the reference count.

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

Parameters

riid
Reference identifier.
ppv
Pointer to the interface.

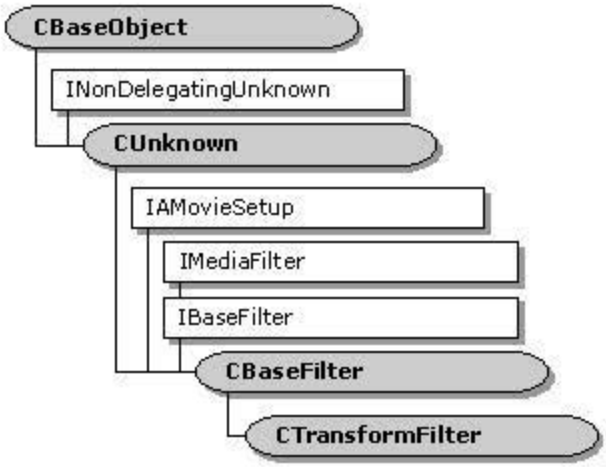
Return Values

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 IReferenceClock, IPersist, and IUnknown interfaces.

CTransformFilter Class



CTransformFilter is an abstract base class that supports a simple transform filter with a single input and a single output. It is derived from the CUnknown class, and it supports the IBaseFilter interface. Each pin, declared as Friends in this class, supports the IPin interface and uses the shared memory transport based on the IMemInputPin interface. The filter uses classes derived from the CBaseFilter class to support **IBaseFilter**; the CTransformInputPin input pin class is derived from the CBaseInputPin class, and the CTransformOutputPin output pin class is derived from the CBaseOutputPin class.

Note that, while most member functions in this class are designed to be overridden, the following pure virtual member functions must be overridden.

- CheckInputType
- CheckTransform
- DecideBufferSize
- GetMediaType
- Transform

For more information about using this class to create a transform filter, see Creating a Transform Filter.

Protected Data Members

Name	Description
m_bEOSDelivered	End-of-stream delivery status flag.
m_bQualityChanged	Status flag that indicates if the stream has degraded. This is set to TRUE in CTransformFilter::Receive if the call to the derived class's **Transform** member function fails (**CTransformFilter::Receive** returns NOERROR in this case because returning S_FALSE indicates that the end-of-stream has arrived).
m_bSampleSkipped	Status flag that indicates if a frame was skipped.

m_csFilter	Critical section that protects the filter state. This critical section is held whenever the state is currently changing or might change. It is passed to the CBaseMediaFilter constructor so that the base class uses it too.
m_csReceive	Critical section that is held when processing events that occur on the receiving thread (CTransformInputPin::Receive and CTransformInputPin::EndOfStream).
m_idTransform	Identifier used for performance measurement. Available only when PERF is defined.
m_pInput	Pointer to the input pin class object.
m_pOutput	Pointer to the output pin class object.

Member Functions

Name	Description
CTransformFilter	Constructs a CTransformFilter object.

Overridable Member Functions

Name	Description
AlterQuality	Receives a quality-control notification from the output pin and provides an opportunity to alter the media stream's quality.
BeginFlush	Receives notification of entering the flushing state and passes it downstream.
BreakConnect	Informs the derived class when the connection is broken.
CheckConnect	Informs the derived class when the connection process is starting.
CheckInputType	Verifies that the input pin supports the media type and proposes the media type of the output pin (pure virtual).
CheckTransform	Verifies that the input and output pins support the media type (pure virtual).
CompleteConnect	Informs the derived class when the connection process has completed.
DecideBufferSize	Sets the number and size of buffers required for the transfer (pure virtual).
EndFlush	Receives notification of leaving the flushing state and passes it downstream.
EndOfStream	Receives an end-of-stream notification and passes it downstream.
GetMediaType	Returns one of the media types that the output pin supports (pure virtual).
GetPin	Returns the pin for the index specified.
GetPinCount	Returns the number of pins on the filter.
NewSegment	Informs the derived class that a new segment has started and delivers it downstream.
Receive	Receives the media sample, calls the CTransformFilter::Transform member function, and then delivers the media sample.
RegisterPerfId	Registers a performance measurement identifier.
SetMediaType	Informs the derived class when the media type is established for the connection.
StartStreaming	Informs the derived class that streaming is starting.
StopStreaming	Informs the derived class that streaming is ending.
Transform	Performs transform operations, reading from the input IMediaSample interface and writing the data to the output **IMediaSample** interface (pure virtual).

Implemented IBaseFilter Methods

Name	Description
FindPin	Retrieves the pin with the specified identifier.
Pause	Transitions the filter to State_Paused state if it is not in this state already, and informs the derived class.
Stop	Transitions the filter to State_Stopped state if it is not in this state already, and informs the derived class.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CTransformFilter::AlterQuality

CTransformFilter Class

Receives a quality-control notification and provides an opportunity to alter the media stream's quality.

```
virtual HRESULT AlterQuality(
  Quality q
  );
```

Parameters

q
Quality-control notification message.

Return Values

Returns an HRESULT value. S_FALSE means to pass the message to the upstream filter (whether or not any action has been taken). An overriding member function can return NOERROR to indicate that the message has been handled completely (or as completely as possible) and no further action should be taken.

Remarks

This member function returns S_FALSE by default. It is called by the CTransformOutputPin::Notify member function before calling the CBaseInputPin::PassNotify member function to pass the quality control message upstream. If the filter is responsible for affecting the quality of the media stream, override this member function and respond to the quality-notification message.

Previous Home Topic Contents Index Next

CTransformFilter::BeginFlush

CTransformFilter Class

Receives notification of entering the flushing state and passes it downstream.

virtual HRESULT BeginFlush( );

Return Values

Returns an HRESULT value.

Remarks

By default, this member function calls the CBaseOutputPin::DeliverEndFlush member function on the output pin to send the **BeginFlush** notification to the next filter. Override this member function if you are using queued data or a worker thread.

Previous Home Topic Contents Index Next

CTransformFilter::BreakConnect

CTransformFilter Class

Informs the derived class when the connection is broken.

virtual HRESULT BreakConnect(
PIN_DIRECTION *dir*
);

Parameters

dir
Direction of the pin.

Return Values

Returns NOERROR by default. The overriding member function returns an HRESULT value.

Remarks

This member function is called by both CTransformInputPin::BreakConnect and CTransformOutputPin::BreakConnect. It returns NOERROR by default. Override this member function to handle special cases in both input and output pin connections. Special cases might typically be releasing interfaces obtained in the CTransformFilter::CheckConnect member function.

Previous Home Topic Contents Index Next

CTransformFilter::CheckConnect

CTransformFilter Class

Informs the derived class when the connection process is starting.

virtual HRESULT CheckConnect(
PIN_DIRECTION *dir*,
IPin **pPin*
);

Parameters

dir
Direction of the pin.
pPin
Pointer to the pin making the connection.

Return Values

Returns NOERROR by default. The overriding member function returns an HRESULT value.

Remarks

This member function is called by both CTransformInputPin::CheckConnect and CTransformOutputPin::CheckConnect. Override this member function to handle special cases in both input and output pin connections. Special cases might include querying (obtaining) other interfaces.

Previous Home Topic Contents Index Next

CTransformFilter::CheckInputType

CTransformFilter Class

Verifies that the input pin supports the media type, and proposes the output pin's media type.

virtual HRESULT CheckInputType(
const CMediaType* *mtIn*
)
PURE;

Parameters

mtIn
Pointer to an input media type object.

Return Values

The overriding member function returns an HRESULT value.

Remarks

You must override this member function to verify the media type. This member function must return an error if it cannot support the media type as an input. If it can, the overriding member function should propose the output media type supplied by the output pin.

Previous Home Topic Contents Index Next

CTransformFilter::CheckTransform

CTransformFilter Class

Verifies that the input and output pins support the media type.

virtual HRESULT CheckTransform(
const CMediaType* *mtIn*,
const CMediaType* *mtOut*
) PURE;

Parameters

mtIn
Pointer to the input media type object.
mtOut
Pointer to the output media type object.

Return Values

The overriding member function returns an HRESULT value.

Remarks

The derived class must implement this member function by overriding it. It should return an error if the filter cannot accept these types as its input and output types.

Previous Home Topic Contents Index Next

CTransformFilter::CompleteConnect

CTransformFilter Class

Informs the derived class when the connection process has completed.

```
virtual HRESULT CompleteConnect(
  PIN_DIRECTION direction,
  IPin *pReceivePin
  );
```

Parameters

direction
Pin direction.
pReceivePin
Pointer to the output pin that is being connected to.

Return Values

Returns an HRESULT value.

Remarks

This member function is called by both CTransformInputPin::CompleteConnect and CTransformOutputPin::CompleteConnect. It returns NOERROR by default. Override this member function to handle special cases in both input and output pin connections.

Previous Home Topic Contents Index Next

CTransformFilter::CTransformFilter

CTransformFilter Class

Constructs a CTransformFilter object.

CTransformFilter(
TCHAR * *pObjectName*,
LPUNKNOWN *lpUnk*,
CLSID *clsid*
);

Parameters

pObjectName
Name given to the CTransformFilter object.
lpUnk
Pointer to LPUNKNOWN.
clsid
Class identifier of the CTransformFilter class.

Return Values

No return value.

Remarks

The constructor of the derived class calls this member function. The pin objects are not created at this time; they are created when calling the CTransformFilter::GetPin member function. Thus the pins (m_pInput and m_pOutput) cannot be referred to in the constructor unless GetPin is first called. (An external object can find pins only by enumerating them or by calling IBaseFilter::FindPin. These each call **GetPin**, so the pins are, in fact, created as soon as they are needed.)

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CTransformFilter::DecideBufferSize

CTransformFilter Class

Sets 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

This member function is called by the CTransformOutputPin::DecideBufferSize member function. Override and implement this member function to call the CMemAllocator::SetProperties member function with appropriate values for the output stream. This call might fail if the allocator cannot satisfy the request.

Previous Home Topic Contents Index Next

CTransformFilter::EndFlush

CTransformFilter Class

Receives notification of leaving the flushing state and passes it downstream.

virtual HRESULT EndFlush( );

Return Values

Returns an HRESULT value.

Remarks

This member function is called by the CTransformInputPin::EndFlush member function. Override this member function if you are using queued data or a worker thread. It calls CBaseOutputPin::DeliverEndFlush to deliver the notification downstream.

Previous Home Topic Contents Index Next

CTransformFilter::EndOfStream

CTransformFilter Class

Receives an end-of-stream notification and passes it downstream.

virtual HRESULT EndOfStream( );

Return Values

Returns an HRESULT value.

Remarks

By default, this member function calls the CBaseOutputPin::DeliverEndOfStream member function on the output pin to send the end-of-stream notification to the next filter. Override this member function if you are using queued data or a worker thread. If you overrode CTransformFilter::Receive and have queued data, you must handle this condition and deliver EOS after all queued data is sent.

Previous Home Topic Contents Index Next

CTransformFilter::FindPin

CTransformFilter Class

Retrieves the pin with the specified identifier.

HRESULT FindPin(
LPCWSTR *Id*,
IPin ***ppPin*
);

Parameters

Id
Identifier of the pin.
ppPin

Pointer to the IPin interface for this pin after the filter has been restored.

Return Values

Returns NOERROR if the pin name was found; otherwise, returns VFW_E_NOT_FOUND.

Remarks

This member function overrides the CBaseFilter::FindPin member function. If the *Id* parameter is "In", it retrieves the input pin's IPin pointer; if the *Id* parameter is "Out", it retrieves the output pin's **IPin** pointer.

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

Previous Home Topic Contents Index Next

CTransformFilter::GetMediaType

CTransformFilter Class

Returns one of the media types that the output pin supports (pure virtual).

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

Parameters

iPosition
Position of the media type in the media type list.
pMediaType
Returned media type object.

Return Values

Returns an HRESULT value by the overriding member function. It returns VFW_S_NO_MORE_ITEMS when asked for a media type beyond the position list. It might return S_FALSE to indicate that the media type exists but is not currently usable. In this case, the IEnumMediaTypes::Next method skips this media type.

Remarks

The derived class is responsible for implementing this member function and maintaining the list of media types that it supports.

The base transform class assumes that only the output pin proposes media types, because the output pin depends on the type of connection of the input pin. For this reason, it is only the CTransformOutputPin::GetMediaType member function of the output pin that is routed to this function.

Previous Home Topic Contents Index Next

CTransformFilter::GetPin

CTransformFilter Class

Returns a pin for a specified index.

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

Parameters

n
Index of the pin to return.

Return Values

Returns a pointer to a CBasePin object.

Remarks

This member function overrides the CBaseFilter::GetPin member function and need not be overridden unless one or more of the transform pin classes (CTransformInputPin or CTransformOutputPin) are overridden. Upon successful return, both pins are valid.

Previous Home Topic Contents Index Next

CTransformFilter::GetPinCount

CTransformFilter Class

Returns the number of pins on the filter.

virtual int GetPinCount( );

Return Values

Returns 2. If you override this class to support more pins, this member function returns the total number of pins on the filter.

Remarks

This member function overrides the CBaseFilter::GetPinCount member function. The CTransformFilter class supports only one input pin and one output pin.

Previous Home Topic Contents Index Next

CTransformFilter::NewSegment

CTransformFilter Class

Informs the derived class that a new segment has started and delivers it downstream.

virtual 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.

Remarks

This member function is called by the CTransformInputPin::NewSegment member function and calls the CBaseOutputPin::DeliverNewSegment member function.

Previous Home Topic Contents Index Next

Previous Home Topic Contents Index Next

CTransformFilter::Pause

CTransformFilter Class

Transitions the filter to State_Paused state if it is not in this state already, and informs the derived class.

HRESULT Pause (void);

Return Values

Returns an HRESULT value.

Remarks

This member function overrides the CBaseFilter::Pause member function and implements the IMediaFilter::Pause method. It checks the input and output pin connections, calls CTransformFilter::StartStreaming, and finally calls the base class implementation (**CBaseFilter::Pause**).

Previous Home Topic Contents Index Next

CTransformFilter::Receive

CTransformFilter Class

Receives the media sample, calls the CTransformFilter::Transform member function, and then delivers the media sample.

HRESULT Receive(
IMediaSample **pSample*
);

Parameters

pSample
Media sample to receive.

Return Values

Returns an HRESULT value.

Remarks

This member function is called by the CTransformInputPin::Receive member function, which implements the IMemInputPin::Receive method. If you override this member function, you might must also override CTransformFilter::EndOfStream, CTransformFilter::BeginFlush, and CTransformFilter::EndFlush.

Previous Home Topic Contents Index Next

CTransformFilter::RegisterPerfId

CTransformFilter Class

Registers a performance measurement identifier.

virtual void RegisterPerfId( );

Return Values

No return value.

Remarks

By default, this member function registers the performance identifier (m_idTransform) with the string "Transform". Override this member function to register a performance measurement with a less generic string. This should be done to avoid confusion with other filters. This member function is enabled only when PERF is defined.

Previous Home Topic Contents Index Next

CTransformFilter::SetMediaType

CTransformFilter Class

Informs the derived class when the media type is established for the connection.

virtual HRESULT SetMediaType(
PIN_DIRECTION *direction*,
const CMediaType **pmt*
) PURE;

Parameters

direction
Pin direction.
pmt
Pointer to the media type object.

Return Values

Returns NOERROR by default. The overriding member function returns an HRESULT value.

Remarks

Override this member function to detect when the media type is set. The implementations of CTransformInputPin::SetMediaType and CTransformOutputPin::SetMediaType call this member function.

Previous Home Topic Contents Index Next

CTransformFilter::StartStreaming

CTransformFilter Class

Informs the derived class that streaming is starting.

virtual HRESULT StartStreaming( );

Return Values

Returns NOERROR by default. The overriding member function returns an HRESULT value.

Remarks

The filter is in the process of switching to active mode (paused or running). Alternatively, you can override this member function to allocate any necessary resources.

CTransformFilter::Stop

CTransformFilter Class

Transitions the filter to State_Stopped state if it is not in this state already, and informs the derived class.

HRESULT Stop(void);

Return Values

Returns an HRESULT value.

Remarks

This member function overrides the CBaseFilter::Stop member function and implements the IMediaFilter::Stop method. It first decommits on the input and output pins by calling CBaseInputPin::Inactive and CBaseOutputPin::Inactive, and then calls CTransformFilter::StopStreaming to inform the derived class.

CTransformFilter::StopStreaming

CTransformFilter Class

Informs the derived class that streaming is ending.

virtual HRESULT StopStreaming( );

Return Values

Returns NOERROR by default. The overriding member function returns an HRESULT value.

Remarks

The filter is in the process of leaving active mode and entering stopped mode. Override this member function to free any resources allocated in StartStreaming.

Previous Home Topic Contents Index Next

CTransformFilter::Transform

CTransformFilter Class

Performs transform operations of the filter.

```
virtual HRESULT Transform(
  IMediaSample * pIn,
  IMediaSample *pOut
  ) PURE;
```

Parameters

pIn
Pointer to the input IMediaSample interface.

pOut
Pointer to the output IMediaSample interface.

Return Values

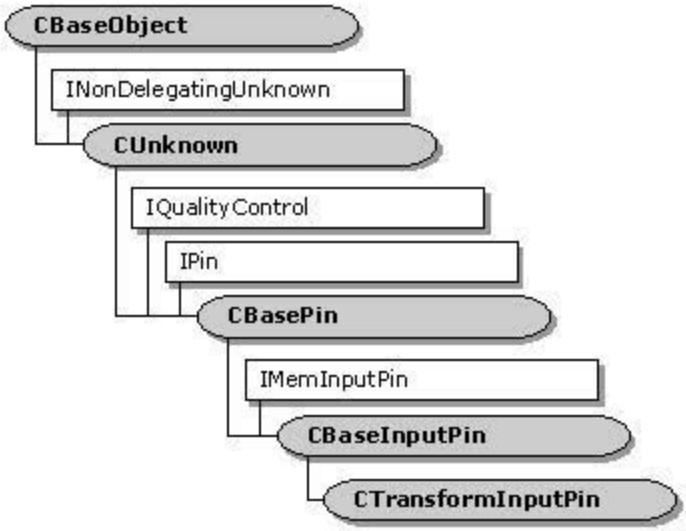
The overriding member function returns an HRESULT value. If it returns S_FALSE, the default implementation of the sample will not be delivered by the default implementation of the CTransformFilter::Receive member function.

Remarks

The CTransformFilter::Receive member function calls this member function, which must be overridden with a member function that implements the transform intended for the filter.

Perform your transform operation in the implementation of this member function, reading the data from the input IMediaSample interface and writing the data to the output **IMediaSample** interface. The member function returns when the transform is complete, without releasing or delivering either of the samples. Change properties on the output sample if they are not the same as the input sample. For example, change the start and stop time IMediaSample::SetTime), sample status flags (IMediaSample::IsSyncPoint), and so on.

CTransformInputPin Class



The **CTransformInputPin** class implements the input pin of a simple transform filter. It is the class assigned to the m_pInput data member of the CTransformFilter class. Typically, you can create objects derived from **CTransformFilter** without modifying the **CTransformInputPin** class. That is, you can usually override the member functions in **CTransformFilter** that are called by member functions of this class. This means that you need not derive your own classes for either of the pin classes.

However, if you want to override this class and have your filter class derived from CTransformFilter, you must override the CTransformFilter::GetPin member function to create pins of your derived class.

Protected Data Members

Name	Description
m_pTransformFilter	Pointer to the owning CTransformFilter object.

Member Functions

Name	Description
CTransformInputPin	Constructs a CTransformInputPin object.
CurrentMediaType	Retrieves the media type currently assigned to the filter.

Overridable Member Functions