An NDoc Documented Class Library
MediaObjectTemplate Namespace

This namespace (and the classes in it) are designed to be used to implement DMOs in .NET. The IMediaObjectImpl class is an abstract class which contains implementations for all the IMediaObject, IMediaParamInfo, and IMediaParams interfaces. There are a number of abstract methods in this class which must be implemented, and some virtual methods which can be overridden if necessary.

Namespace hierarchy

Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMediaObjectImpl</td>
<td>This abstract class can be used to implement a DMO in .NET.</td>
</tr>
</tbody>
</table>

Enumerations

<table>
<thead>
<tr>
<th>Enumeration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMediaObjectImpl.TimeFormatFlags</td>
<td>Used by the IMediaObjectImpl constructor to specify which timeformats are supported</td>
</tr>
</tbody>
</table>
An NDoc Documented Class Library
**IMediaObjectImpl Class**

This abstract class can be used to implement a DMO in .NET.

For a list of all members of this type, see IMediaObjectImpl Members.

```
System.Object MediaObjectTemplate.IMediaObjectImpl
```

```csharp
public abstract class IMediaObjectImpl : IMediaObject, IMediaParamInfo, IMediaParams
```

**Thread Safety**

Public static *(Shared in Visual Basic)* members of this type are safe for multithreaded operations. Instance members are not guaranteed to be thread-safe.

**Remarks**

Before attempting to use this class, read the MSDN docs on DMOs! In particular read about IMediaObject, IMediaParamInfo, IMediaParams, and the DMO Wrapper Filter (if you are using DirectShow graphs).

When you read the MSDN docs about creating a DMO, they refer to a template that you can use to make things easier. That template served as the inspiration for this class. To create a DMO, you can just create a class that implements this abstract class, write code for the abstract methods, and you should be good to go.

Here is a more detailed description of the steps you need to take. Note that you can look at the sample code for examples of these steps.

1) Other than ripping out the rather lame logging, you shouldn’t need to change any code in IMediaObjectImpl.cs. It is the initial entry point for all the IMediaObject interfaces. It performs parameter checking, makes sure the call is appropriate for the current state, etc. As needed it will make calls to the abstract and virtual methods of the
class.

2) Create a class which implements the abstract IMediaObjectImpl class:

```
[ComVisible(true), Guid("7EF28FD7-E88F-45bb-9CDD-8A62956F2D75
ClassInterface(ClassInterfaceType.None)]
public class DmoFlip : IMediaObjectImpl
```

3) Generate your own guid so the samples won't interfere with your code: If you are running Dev Studio, go to Tools/Create Guid, choose "Registry Format", click "Copy", then paste into your code.

4) Create the constructor for your class. It must not take any parameters:

```
public DmoFlip() : base(InputPinCount, OutputPinCount)
```

If you are planning to use this DMO with the DirectShow DMO Wrapper Filter, note that (up to and including DX v9.0) InputPinCount must be 1, and OutputPinCount must be > 0. The ParamCount is the number of parameters your DMO supports, and can be zero. In general, you should use TimeFormatFlags.Reference for the last paramter.

5) Register the parameters your DMO supports using `ParamDefine`. This must be done in the constructor (unless you have no parameters). Doing this allows you to support IMediaParamInfo and IMediaParams. You will also need to use `ParamCalcValueForTime` to find out what parameter value you should use at any given point during the stream. See the docs for these two methods for details.

6) Create the COM register/unregister methods:

```
[ComRegisterFunctionAttribute]
static private void DoRegister(Type t)

[ComUnregisterFunctionAttribute]
static private void UnregisterFunction(Type t)
```

These tell the OS about your DMO. If you are distributing your code, you will need to make sure they get called during installation (read
about RegAsm in the .NET docs). At a minimum, you will need to call DMORegister to register your DMO. See the sample for how this is done.

**WARNING:** If you use the "Register for COM Interop" compiler switch, the compiler will attempt to register DirectShowLib.dll as well as your DMO. Since DirectShowLib has no registration to perform, this generates an error. That is why the sample uses pre/post build events to perform the registration. You may need to adjust this command for your particular installation.

7) Do everything else. There are 7 abstract methods for which you must write code. These methods are listed in the IMediaObjectImpl Methods page in the Protected Instance Methods section. These methods will contain the information specific to your DMO, and describe what type of data you are willing to process, and perform the actual processing. Note that since the abstract class has verified the parameters, you do not need to re-check them in your implementation. See the descriptions for each method and the sample for details about what each of these methods must do.

You may also need to override some of the 11 virtual methods if their default implementation doesn't match your specific needs. See the documentation for each of these specific methods for details.

If you aren't already knowledgeable about COM and writing multi-threaded apps, this is probably a good time to do a little research. You may have multiple instances of your DMO running in the same process, in multiple processes, called on different threads, etc.

As a simple example of the things you should be thinking of, the logging in (debug builds of) IMediaObjectImpl.cs opens the file as non-sharable. However, if two applications try to instantiate your DMO, the second will fail, solely due to not being able to open the log file. Probably not the desired behavior (told you the logging was lame).

**Requirements**

**Namespace:** [MediaObjectTemplate](#)
Assembly: DmoFlip (in DmoFlip.dll)

See Also

IMediaObjectImpl Members | MediaObjectTemplate Namespace
An NDoc Documented Class Library
### IMediaObjectImpl Members

#### IMediaObjectImpl overview

#### Protected Static Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_INVALIDARG</td>
<td>COM return code indicating invalid argument specified</td>
</tr>
<tr>
<td>E_NOTIMPL</td>
<td>COM return code indicating method not supported</td>
</tr>
<tr>
<td>E_POINTER</td>
<td>COM return code indicating invalid pointer provided</td>
</tr>
<tr>
<td>E_UNEXPECTED</td>
<td>COM return code indicating a method called at an unexpected time</td>
</tr>
<tr>
<td>S_FALSE</td>
<td>COM return code indicating partial success</td>
</tr>
<tr>
<td>S_OK</td>
<td>COM return code indicating success</td>
</tr>
</tbody>
</table>

#### Protected Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MoCloneMediaType</td>
<td>Make a clone of a media type</td>
</tr>
<tr>
<td>TypesMatch</td>
<td>Check to see if two Media Types are exactly the same</td>
</tr>
</tbody>
</table>

#### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddEnvelope</td>
<td>COM entry point for IMediaParams.AddEnvelope</td>
</tr>
<tr>
<td>AllocateStreamingResources</td>
<td>COM entry point for IMediaObject.AllocateStreamingResources</td>
</tr>
<tr>
<td>Discontinuity</td>
<td>COM entry point for IMediaObject.Discontinuity</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>(inherited from Object) Determines whether the specified <strong>Object</strong> is equal to the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>Flush</strong></td>
<td>COM entry point for IMediaObject.Flush</td>
</tr>
<tr>
<td><strong>FlushEnvelope</strong></td>
<td>COM entry point for IMediaParams.FlushEnvelope</td>
</tr>
<tr>
<td><strong>FreeStreamingResources</strong></td>
<td>COM entry point for IMediaObject.FreeStreamingResources</td>
</tr>
<tr>
<td><strong>GetCurrentTimeFormat</strong></td>
<td>COM entry point for IMediaParamInfo.GetCurrentTimeFormat</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>(inherited from Object) Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td><strong>GetInputCurrentType</strong></td>
<td>COM entry point for IMediaObject.GetInputCurrentType</td>
</tr>
<tr>
<td><strong>GetInputMaxLatency</strong></td>
<td>COM entry point for IMediaObject.GetInputMaxLatency</td>
</tr>
<tr>
<td><strong>GetInputSizeInfo</strong></td>
<td>COM entry point for IMediaObject.GetInputSizeInfo</td>
</tr>
<tr>
<td><strong>GetInputStatus</strong></td>
<td>COM entry point for IMediaObject.GetInputStatus</td>
</tr>
<tr>
<td><strong>GetInputStreamInfo</strong></td>
<td>COM entry point for IMediaObject.GetInputStreamInfo</td>
</tr>
<tr>
<td><strong>GetInputType</strong></td>
<td>COM entry point for IMediaObject.GetInputType</td>
</tr>
<tr>
<td><strong>GetNumTimeFormats</strong></td>
<td>COM entry point for IMediaParamInfo.GetNumTimeFormats</td>
</tr>
<tr>
<td><strong>GetOutputCurrentType</strong></td>
<td>COM entry point for IMediaObject.GetOutputCurrentType</td>
</tr>
<tr>
<td><strong>GetOutputSizeInfo</strong></td>
<td>COM entry point for IMediaObject.GetOutputSizeInfo</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetOutputStreamInfo</td>
<td>COM entry point for IMediaObject.GetOutputStreamInfo</td>
</tr>
<tr>
<td>GetOutputType</td>
<td>COM entry point for IMediaObject.GetOutputType</td>
</tr>
<tr>
<td>GetParam</td>
<td>COM entry point for IMediaParams.GetParam</td>
</tr>
<tr>
<td>GetParamCount</td>
<td>COM entry point for IMediaParamInfo.GetParamCount</td>
</tr>
<tr>
<td>GetParamInfo</td>
<td>COM entry point for IMediaParamInfo.GetParamInfo</td>
</tr>
<tr>
<td>GetParamText</td>
<td>COM entry point for IMediaParamInfo.GetParamText</td>
</tr>
<tr>
<td>GetStreamCount</td>
<td>COM entry point for IMediaObject.GetStreamCount</td>
</tr>
<tr>
<td>GetSupportedTimeFormat</td>
<td>COM entry point for IMediaParamInfo.GetSupportedTimeFormat</td>
</tr>
<tr>
<td>GetType (inherited from Object)</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td>Lock</td>
<td>COM entry point for IMediaObject.Lock</td>
</tr>
<tr>
<td>ProcessInput</td>
<td>COM entry point for IMediaObject.ProcessInput</td>
</tr>
<tr>
<td>ProcessOutput</td>
<td>COM entry point for IMediaObject.ProcessOutput</td>
</tr>
<tr>
<td>SetInputMaxLatency</td>
<td>COM entry point for IMediaObject.SetInputMaxLatency</td>
</tr>
<tr>
<td>SetInputType</td>
<td>COM entry point for IMediaObject.SetInputType</td>
</tr>
<tr>
<td>SetOutputType</td>
<td>COM entry point for IMediaObject.SetOutputType</td>
</tr>
<tr>
<td>SetParam</td>
<td>COM entry point for IMediaObject.SetParam</td>
</tr>
</tbody>
</table>
**IMediaParams.SetParam**

| COM entry point for IMediaParams.SetTimeFormat |

**SetTimeFormat**

| (inherited from Object) |

| Returns a String that represents the current Object |

## Protected Instance Constructors

| IMediaObjectImpl Constructor | Constructor |

## Protected Instance Methods

| InputType | Get the AMMediaType for the specified Input stream |

| InputTypeSet | Check whether the media type is set for the specified input stream |

| InternalAcceptingInput | (Abstract) Report whether more input buffers can be accepted |

| InternalAllocateStreamingResources | (Virtual) Allows stream resources to be allocated |

| InternalCheckInputType | (Abstract) Determine whether the input stream supports a specific media type |

| InternalCheckOutputType | (Abstract) Determine whether the output stream supports a specific media type |

| InternalDiscontinuity | (Virtual) Called to notify of a stream discontinuity |

<p>| InternalFlush | (Abstract) Called to flush all pending processing |</p>
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InternalFreeStreamingResources</td>
<td>(Virtual) Allows stream resources to be released</td>
</tr>
<tr>
<td>InternalGetCurrentTime</td>
<td>(Virtual) Returns the current time in the media stream</td>
</tr>
<tr>
<td>InternalGetInputMaxLatency</td>
<td>(Virtual) Retrieves the maximum latency on a specified input stream</td>
</tr>
<tr>
<td>InternalGetInputSizeInfo</td>
<td>(Virtual) Retrieves the buffer requirements for a specified input stream</td>
</tr>
<tr>
<td>InternalGetInputStreamInfo</td>
<td>(Virtual) Controls information about how input buffers are formatted</td>
</tr>
<tr>
<td>InternalGetInputType</td>
<td>(Virtual) Retrieves a preferred media type for a specified input stream</td>
</tr>
<tr>
<td>InternalGetOutputSizeInfo</td>
<td>(Abstract) Determine the requirements for the output stream</td>
</tr>
<tr>
<td>InternalGetOutputStreamInfo</td>
<td>(Virtual) Controls information about how output buffers are formatted</td>
</tr>
<tr>
<td>InternalGetOutputType</td>
<td>(Virtual) Retrieves a preferred media type for a specified output stream</td>
</tr>
<tr>
<td>InternalProcessInput</td>
<td>(Abstract) Accept input buffers to be processed</td>
</tr>
<tr>
<td>InternalProcessOutput</td>
<td>(Abstract) Process the input buffers from a previous call to InternalProcessInput into the provided output buffers</td>
</tr>
<tr>
<td>InternalSetInputMaxLatency</td>
<td>(Virtual) Set the maximum</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong> (inherited from Object)</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
<tr>
<td><strong>OutputType</strong></td>
<td>Get the AMMediaType for the specified Output stream.</td>
</tr>
<tr>
<td><strong>OutputTypeSet</strong></td>
<td>Check whether the media type is set for the specified output stream.</td>
</tr>
<tr>
<td><strong>ParamCalcValueForTime</strong></td>
<td>Given a parameter number and a time, return the parameter value at that time.</td>
</tr>
<tr>
<td><strong>ParamDefine</strong></td>
<td>Create a definition for a parameter that is accessible thru IMediaParamInfo and IMediaParams.</td>
</tr>
</tbody>
</table>

**See Also**

[IMediaObjectImpl Class] | [MediaObjectTemplate Namespace]
An NDoc Documented Class Library
IMediaObjectImpl Constructor

Constructor

```java
protected IMediaObjectImpl(
    int iInputs,
    int iOutputs,
    int iParams,
    TimeFormatFlags iT imeFormats
);
```

Parameters

*iInputs*  
Number of input streams

*iOutputs*  
Number of output streams

*iParams*  
Number of parameters

*iTimeFormats*  
What time formats the parameters support

Remarks

This constructor will be called from the constructor of the class that implements IMediaObjectImpl. See [IMediaObjectImpl](#) for a step by step description of the process.

See Also

[IMediaObjectImpl Class](#) | [MediaObjectTemplate Namespace](#)
An NDoc Documented Class Library
IMediaObjectImpl Fields

The fields of the IMediaObjectImpl class are listed below. For a complete list of IMediaObjectImpl class members, see the IMediaObjectImpl Members topic.

Protected Static Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_INVALIDARG</td>
<td>COM return code indicating invalid argument specified</td>
</tr>
<tr>
<td>E_NOTIMPL</td>
<td>COM return code indicating method not supported</td>
</tr>
<tr>
<td>E_POINTER</td>
<td>COM return code indicating invalid pointer provided</td>
</tr>
<tr>
<td>E_UNEXPECTED</td>
<td>COM return code indicating a method called at an unexpected time</td>
</tr>
<tr>
<td>S_FALSE</td>
<td>COM return code indicating partial success</td>
</tr>
<tr>
<td>S_OK</td>
<td>COM return code indicating success</td>
</tr>
</tbody>
</table>

See Also

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
IMediaObjectImpl.E_INVALIDARG Field

COM return code indicating invalid argument specified

```cpp
protected const int E_INVALIDARG = -2147024809;
```

See Also

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
**IMediaObjectImpl.E_NOTIMPL Field**

COM return code indicating method not supported

```cpp
protected const int E_NOTIMPL = -2147467263;
```

See Also

[IMediaObjectImpl Class](#) | [MediaObjectTemplate Namespace](#)
An NDoc Documented Class Library
**IMediaObjectImpl.E_POINTER Field**

COM return code indicating invalid pointer provided

```c
protected const int E_POINTER = -2147467261;
```

See Also

- [IMediaObjectImpl Class](#) | [MediaObjectTemplate Namespace](#)
An NDoc Documented Class Library
COM return code indicating a method called at an unexpected time

protected const int E_UNEXPECTED = -2147418113;

See Also

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
IMediaObjectImpl.S_FALSE Field

COM return code indicating partial success

```cpp
protected const int S_FALSE = 1;
```

See Also

[IMediaObjectImpl Class](#) | [MediaObjectTemplate Namespace](#)
An NDoc Documented Class Library
**IMediaObjectImpl.S_OK Field**

COM return code indicating success

```c
protected const int S_OK = 0;
```

See Also

[IMediaObjectImpl Class](#) | [MediaObjectTemplate Namespace](#)
An NDoc Documented Class Library
### IMediaObjectImpl Methods

The methods of the **IMediaObjectImpl** class are listed below. For a complete list of **IMediaObjectImpl** class members, see the **IMediaObjectImpl Members** topic.

#### Protected Static Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>s MoCloneMediaType</td>
<td>Make a clone of a media type</td>
</tr>
<tr>
<td>s TypesMatch</td>
<td>Check to see if two Media Types are exactly the same</td>
</tr>
</tbody>
</table>

#### Public Instance Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddEnvelope</td>
<td>COM entry point for IMediaParams.AddEnvelope</td>
</tr>
<tr>
<td>AllocateStreamingResources</td>
<td>COM entry point for IMediaObject.AllocateStreamingResources</td>
</tr>
<tr>
<td>Discontinuity</td>
<td>COM entry point for IMediaObject.Discontinuity</td>
</tr>
<tr>
<td>Equals (inherited from Object)</td>
<td>Determines whether the specified Object equal to the current Object.</td>
</tr>
<tr>
<td>Flush</td>
<td>COM entry point for IMediaObject.Flush</td>
</tr>
<tr>
<td>FlushEnvelope</td>
<td>COM entry point for IMediaParams.FlushEnvelope</td>
</tr>
<tr>
<td>FreeStreamingResources</td>
<td>COM entry point for IMediaObject.FreeStreamingResources</td>
</tr>
<tr>
<td>GetCurrentTimeFormat</td>
<td>COM entry point for IMediaParamInfo.GetCurrentTimeFormat</td>
</tr>
<tr>
<td>GetHashCode (inherited from Object)</td>
<td>Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.</td>
</tr>
<tr>
<td>GetInputCurrentType</td>
<td>COM entry point for IMediaObject.GetInputCurrentType</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetInputMaxLatency</td>
<td>COM entry point for IMediaObject.GetInputMaxLatency</td>
</tr>
<tr>
<td>GetInputSizeInfo</td>
<td>COM entry point for IMediaObject.GetInputSizeInfo</td>
</tr>
<tr>
<td>GetInputStatus</td>
<td>COM entry point for IMediaObject.GetInputStatus</td>
</tr>
<tr>
<td>GetInputStreamInfo</td>
<td>COM entry point for IMediaObject.GetInputStreamInfo</td>
</tr>
<tr>
<td>GetInputType</td>
<td>COM entry point for IMediaObject.GetInputType</td>
</tr>
<tr>
<td>GetNumTimeFormats</td>
<td>COM entry point for IMediaParamInfo.GetNumTimeFormats</td>
</tr>
<tr>
<td>GetOutputCurrentType</td>
<td>COM entry point for IMediaObject.GetOutputCurrentType</td>
</tr>
<tr>
<td>GetOutputSizeInfo</td>
<td>COM entry point for IMediaObject.GetOutputSizeInfo</td>
</tr>
<tr>
<td>GetOutputStreamInfo</td>
<td>COM entry point for IMediaObject.GetOutputStreamInfo</td>
</tr>
<tr>
<td>GetOutputType</td>
<td>COM entry point for IMediaObject.GetOutputType</td>
</tr>
<tr>
<td>GetParam</td>
<td>COM entry point for IMediaParams.GetParam</td>
</tr>
<tr>
<td>GetParamCount</td>
<td>COM entry point for IMediaParamInfo.GetParamCount</td>
</tr>
<tr>
<td>GetParamInfo</td>
<td>COM entry point for IMediaParamInfo.GetParamInfo</td>
</tr>
<tr>
<td>GetParamText</td>
<td>COM entry point for IMediaParamInfo.GetParamText</td>
</tr>
<tr>
<td>GetStreamCount</td>
<td>COM entry point for IMediaObject.GetStreamCount</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>GetSupportedTimeFormat</td>
<td>COM entry point for IMediaParamInfo.GetSupportedTimeFormat</td>
</tr>
<tr>
<td>GetType (inherited from Object)</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>Lock</td>
<td>COM entry point for IMediaObject.Lock</td>
</tr>
<tr>
<td>ProcessInput</td>
<td>COM entry point for IMediaObject.ProcessInput</td>
</tr>
<tr>
<td>ProcessOutput</td>
<td>COM entry point for IMediaObject.ProcessOutput</td>
</tr>
<tr>
<td>SetInputMaxLatency</td>
<td>COM entry point for IMediaObject.SetInputMaxLatency</td>
</tr>
<tr>
<td>SetInputType</td>
<td>COM entry point for IMediaObject.SetInputType</td>
</tr>
<tr>
<td>SetOutputType</td>
<td>COM entry point for IMediaObject.SetOutputType</td>
</tr>
<tr>
<td>SetParam</td>
<td>COM entry point for IMediaParams.SetParam</td>
</tr>
<tr>
<td>SetTimeFormat</td>
<td>COM entry point for IMediaParams.SetTimeFormat</td>
</tr>
<tr>
<td>ToString (inherited from Object)</td>
<td>Returns a String that represents the current Object.</td>
</tr>
</tbody>
</table>

**Protected Instance Methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InputType</td>
<td>Get the AMMediaType for the specified Input stream</td>
</tr>
<tr>
<td>InputTypeSet</td>
<td>Check whether the media type is set for the specified input stream</td>
</tr>
<tr>
<td>InternalAcceptingInput</td>
<td>(Abstract) Report whether more input buffers can be accepted</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>InternalAllocateStreamingResources</td>
<td>(Virtual) Allows stream resources to be allocated</td>
</tr>
<tr>
<td>InternalCheckInputType</td>
<td>(Abstract) Determine whether the input stream supports a specific media type</td>
</tr>
<tr>
<td>InternalCheckOutputType</td>
<td>(Abstract) Determine whether the output stream supports a specific media type</td>
</tr>
<tr>
<td>InternalDiscontinuity</td>
<td>(Virtual) Called to notify of a stream discontinuity</td>
</tr>
<tr>
<td>InternalFlush</td>
<td>(Abstract) Called to flush all pending processing</td>
</tr>
<tr>
<td>InternalFreeStreamingResources</td>
<td>(Virtual) Allows stream resources to be released</td>
</tr>
<tr>
<td>InternalGetCurrentTime</td>
<td>(Virtual) Returns the current time in the media stream</td>
</tr>
<tr>
<td>InternalGetInputMaxLatency</td>
<td>(Virtual) Retrieves the maximum latency on a specified input stream</td>
</tr>
<tr>
<td>InternalGetInputSizeInfo</td>
<td>(Virtual) Retrieves the buffer requirements for a specified input stream</td>
</tr>
<tr>
<td>InternalGetInputStreamInfo</td>
<td>(Virtual) Controls information about how input buffers are formatted</td>
</tr>
<tr>
<td>InternalGetInputType</td>
<td>(Virtual) Retrieves a preferred media type for a specified input stream</td>
</tr>
<tr>
<td>InternalGetOutputSizeInfo</td>
<td>(Abstract) Determine the requirements for the output</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>InternalGetOutputStreamInfo</code></td>
<td>(Virtual) Controls information about how output buffers are formatted</td>
</tr>
<tr>
<td><code>InternalGetOutputType</code></td>
<td>(Virtual) Retrieves a preferred media type for a specified output stream</td>
</tr>
<tr>
<td><code>InternalProcessInput</code></td>
<td>(Abstract) Accept input buffers to be processed</td>
</tr>
<tr>
<td><code>InternalProcessOutput</code></td>
<td>(Abstract) Process the input buffers from a previous call to <code>InternalProcessInput</code> into the provided output buffers</td>
</tr>
<tr>
<td><code>InternalSetInputMaxLatency</code></td>
<td>(Virtual) Set the maximum latency on a specified input stream</td>
</tr>
<tr>
<td><code>MemberwiseClone</code> (inherited from <code>Object</code>)</td>
<td>Creates a shallow copy of the current <code>Object</code>.</td>
</tr>
<tr>
<td><code>OutputType</code></td>
<td>Get the AMMediaType for the specified Output stream</td>
</tr>
<tr>
<td><code>OutputTypeSet</code></td>
<td>Check whether the media type is set for the specified output stream</td>
</tr>
<tr>
<td><code>ParamCalcValueForTime</code></td>
<td>Given a parameter number and a time, return the parameter value at that time.</td>
</tr>
<tr>
<td><code>ParamDefine</code></td>
<td>Create a definition for a parameter that is accessible thru <code>IMediaParamInfo</code> and <code>IMediaParams</code>.</td>
</tr>
</tbody>
</table>
See Also

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
IMediaObjectImpl/AddEnvelope Method

COM entry point for IMediaParams/AddEnvelope

```java
public int AddEnvelope(
    int dwParamIndex,
    int cSegments,
    MPEnvelopeSegment[] pEnvelopeSegments);
```

Implements

IMediaParams/AddEnvelope

Remarks

There should be no need to modify or override this method. See ParamDefine and ParamCalcValueForTime for details about how this works.

See Also

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
**IMediaObjectImpl.AllocateStreamingResources Method**

COM entry point for IMediaObject.AllocateStreamingResources

```java
public int AllocateStreamingResources();
```

**Implements**

IMediaObject.AllocateStreamingResources

**Remarks**

There should be no need to modify or override this method. It will call the abstract and virtual methods to perform its work.

**See Also**

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
**IMediaObjectImpl.Discontinuity Method**

COM entry point for IMediaObject.Discontinuity

```java
public int Discontinuity(int ulStreamIndex);
```

**Implements**

*IMediaObject.Discontinuity*

**Remarks**

There should be no need to modify or override this method. It will call the abstract and virtual methods to perform its work.

**See Also**

*IMediaObjectImpl Class* | *MediaObjectTemplate Namespace*
An NDoc Documented Class Library
**IMediaObjectImpl.Flush Method**

COM entry point for IMediaObject.Flush

```csharp
public int Flush();
```

**Implements**

IMediaObject.Flush

**Remarks**

There should be no need to modify or override this method. It will call the abstract and virtual methods to perform its work.

**See Also**

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
**IMediaObjectImpl.FlushEnvelope Method**

COM entry point for IMediaParams.FlushEnvelope

```csharp
public int FlushEnvelope(
    int dwParamIndex,
    long refTimeStart,
    long refTimeEnd
);
```

**Implements**

IMediaParams.FlushEnvelope

**Remarks**

There should be no need to modify or override this method. See ParamDefine and ParamCalcValueForTime for details about how this works.

**See Also**

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
IMediaObjectImpl.FreeStreamingResources Method

COM entry point for IMediaObjectFreeStreamingResources

public int FreeStreamingResources();

Implements

IMediaObject.FreeStreamingResources

Remarks

There should be no need to modify or override this method. It will call the abstract and virtual methods to perform its work.

See Also

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
IMediaObjectImpl.GetCurrentTimeFormat Method

COM entry point for IMediaParamInfo.GetCurrentTimeFormat

```csharp
public int GetCurrentTimeFormat(
    out Guid pguidTimeFormat,
    out int pTimeData
);
```

Implements

IMediaParamInfo.GetCurrentTimeFormat

Remarks

There should be no need to modify or override this method. See ParamDefine and ParamCalcValueForTime for details about how this works.

See Also

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
IMediaObjectImpl.GetInputCurrentType Method

COM entry point for IMediaObject.GetInputCurrentType

```csharp
public int GetInputCurrentType(
    int ulStreamIndex,
    AMMediaType pmt
);
```

Implements

IMediaObject.GetInputCurrentType

Remarks

There should be no need to modify or override this method. It will call the abstract and virtual methods to perform its work.

See Also

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
**IMediaObjectImpl.GetInputMaxLatency Method**

COM entry point for IMediaObject.GetInputMaxLatency

```csharp
public int GetInputMaxLatency(
    int ulStreamIndex,
    out long prtLatency
);
```

**Implements**

- IMediaObject.GetInputMaxLatency

**Remarks**

There should be no need to modify or override this method. It will call the abstract and virtual methods to perform its work.

**See Also**

- IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
**Implements**

IMediaObject.GetInputSizeInfo

**Remarks**

There should be no need to modify or override this method. It will call the abstract and virtual methods to perform its work.

**See Also**

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
**IMediaObjectImpl.GetInputStatus Method**

COM entry point for IMediaObject.GetInputStatus

```csharp
public int GetInputStatus(
    int uiStreamIndex,
    out DMOInputStatusFlags pdwStatus
);
```

**Implements**

IMediaObject.GetInputStatus

**Remarks**

There should be no need to modify or override this method. It will call the abstract and virtual methods to perform its work.

**See Also**

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
**IMediaObjectImpl.GetInputStreamInfo Method**

COM entry point for IMediaObject.GetInputStreamInfo

```csharp
public int GetInputStreamInfo(
    int ulStreamIndex,
    out DMOInputStreamInfo pdwFlags
);
```

**Implements**

IMediaObject.GetInputStreamInfo

**Remarks**

There should be no need to modify or override this method. It will call the abstract and virtual methods to perform its work.

**See Also**

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
**IMediaObjectImpl.GetInputType Method**

COM entry point for IMediaObject.GetInputType

```java
public int GetInputType(
    int ulStreamIndex,
    int ulTypeIndex,
    AMMediaType pmt
);
```

**Implements**

IMediaObject.GetInputType

**Remarks**

There should be no need to modify or override this method. It will call the abstract and virtual methods to perform its work.

**See Also**

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
IMediaObjectImpl.GetNumTimeFormats Method

COM entry point for IMediaParamInfo.GetNumTimeFormats

```csharp
public int GetNumTimeFormats(
    out int pdwNumTimeFormats
);
```

Implements

IMediaParamInfo.GetNumTimeFormats

Remarks

There should be no need to modify or override this method. See ParamDefine and ParamCalcValueForTime for details about how this works.

See Also

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
COM entry point for IMediaObject.GetOutputCurrentType

```csharp
public int GetOutputCurrentType(
    int uiStreamIndex,
    AMMediaType pmt
);
```

**Implements**

IMediaObject.GetOutputCurrentType

**Remarks**

There should be no need to modify or override this method. It will call the abstract and virtual methods to perform its work.

**See Also**

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
IMediaObjectImpl.GetOutputSizeInfo Method

COM entry point for IMediaObject.GetOutputSizeInfo

```csharp
public int GetOutputSizeInfo(
    int ulStreamIndex,
    out int pulSize,
    out int pulAlignment
);
```

Implements
- IMediaObject.GetOutputSizeInfo

Remarks
There should be no need to modify or override this method. It will call the abstract and virtual methods to perform its work.

See Also
- IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
**IMediaObjectImpl.GetOutputStreamInfo Method**

COM entry point for IMediaObject.GetOutputStreamInfo

```java
public int GetOutputStreamInfo(
    int ulStreamIndex,
    out DMOOutputStreamInfo pdwFlags)
```

**Implements**

IMediaObject.GetOutputStreamInfo

**Remarks**

There should be no need to modify or override this method. It will call the abstract and virtual methods to perform its work.

**See Also**

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
COM entry point for IMediaObject.GetOutputType

```csharp
public int GetOutputType(
    int uiStreamIndex,
    int ulTypeIndex,
    AMMediaType pmt
);
```

**Implements**

IMediaObject.GetOutputType

**Remarks**

There should be no need to modify or override this method. It will call the abstract and virtual methods to perform its work.

**See Also**

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
**IMediaObjectImpl.GetParam Method**

COM entry point for IMediaParams.GetParam

```csharp
public int GetParam(
    int dwParamIndex,
    out MPData pValue
);
```

**Implements**

IMediaParams.GetParam

**Remarks**

There should be no need to modify or override this method. See ParamDefine and ParamCalcValueForTime for details about how this works.

**See Also**

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
IMediaObjectImpl.GetParamCount Method

COM entry point for IMediaParamInfo.GetParamCount

```csharp
public int GetParamCount(
    out int pdwParams
);
```

Implements

IMediaParamInfo.GetParamCount

Remarks

There should be no need to modify or override this method. See ParamDefine and ParamCalcValueForTime for details about how this works.

See Also

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
COM entry point for IMediaParamInfo.GetParamInfo

```csharp
public int GetParamInfo(
    int dwParamIndex,
    out ParamInfo pInfo
);
```

**Implements**

IMediaParamInfo.GetParamInfo

**Remarks**

There should be no need to modify or override this method. See ParamDefine and ParamCalcValueForTime for details about how this works.

**See Also**

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
IMediaObjectImpl.GetParamText Method

COM entry point for IMediaParamInfo.GetParamText

```csharp
public int GetParamText(
    int dwParamIndex,
    out IntPtr ppwchText
);
```

Implements

IMediaParamInfo.GetParamText

Remarks

There should be no need to modify or override this method. See ParamDefine and ParamCalcValueForTime for details about how this works.

See Also

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
COM entry point for IMediaObject.GetStreamCount

```csharp
public int GetStreamCount(
    out int pulNumberOfInputStreams,
    out int pulNumberOfOutputStreams
);```

**Implements**

IMediaObject.GetStreamCount

**Remarks**

There should be no need to modify or override this method. It will call the abstract and virtual methods to perform its work.

**See Also**

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
**IMediaObjectImpl.GetSupportedTimeFormat Method**

COM entry point for IMediaParamInfo.GetSupportedTimeFormat

```csharp
public int GetSupportedTimeFormat(
    int dwFormatIndex,
    out Guid pguidTimeFormat
);
```

**Implements**

IMediaParamInfo.GetSupportedTimeFormat

**Remarks**

There should be no need to modify or override this method. See ParamDefine and ParamCalcValueForTime for details about how this works.

**See Also**

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
**IMediaObjectImpl.InputType Method**

Get the AMMediaType for the specified Input stream

```java
protected AMMediaType InputType(
    int ulInputStreamIndex
);
```

**Parameters**

*ulInputStreamIndex*

The stream to get the media type for

**Return Value**

The media type for the stream, or null if not set

**Remarks**

The abstract class will call `InternalCheckInputType` to see whether a given media type is supported. To see what media type was actually set, the derived class can call this method.

**See Also**

[IMediaObjectImpl Class](#) | [MediaObjectTemplate Namespace](#)
An NDoc Documented Class Library
IMediaObjectImpl.InputTypeSet Method

Check whether the media type is set for the specified input stream

```c++
protected bool InputTypeSet(
    int ulInputStreamIndex
);
```

Parameters

`ulInputStreamIndex`
Zero based stream number to check

Return Value

true if the stream type is set

See Also

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
IMediaObjectImpl.InternalAcceptingInput Method

(Abstract) Report whether more input buffers can be accepted

```java
protected abstract int InternalAcceptingInput(int dwInputStreamIndex)
```

Parameters

- `dwInputStreamIndex`
  - Input stream number

Return Value

- S_OK if the implementor is ready to accept an input buffer, else S_FALSE

Remarks

This method is called by the abstract class. If the implementor has room for another input buffer, it should return S_OK. It is perfectly acceptable for a DMO to only accept one input buffer at a time, and to return S_FALSE until InternalProcessOutput has been called to process the buffer.

See Also

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
(Virtual) Allows stream resources to be allocated

protected virtual int InternalAllocateStreamingResources();

Return Value

S_OK for successful operation

Remarks

Allows the implementor to allocate any resources necessary for performing the processing. The default implementation assumes we don't need to allocate any addition resources to perform the processing.

See Also

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
IMediaObjectImpl.InternalCheckInputType Method

(Abstract) Determine whether the input stream supports a specific media type

```c++
protected abstract int InternalCheckInputType(
    int dwInputStreamIndex,
    AMMediaType pmt
);
```

Parameters

- `dwInputStreamIndex`  
  Input stream number

- `pmt`  
  The media type to check

Return Value

- S_OK if the specified stream supports the specified media type, else DMOResults.E_InvalidType

Remarks

This method is called by the abstract class. The implementor should check the properties of the AMMediaType to ensure that if a sample of the specified type is sent, it will be able to process it.

See Also

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
IMediaObjectImpl.InternalCheckOutputType Method

(Abstract) Determine whether the output stream supports a specific media type

```c
protected abstract int InternalCheckOutputType(
    int dwOutputStreamIndex,
    AMMediaType pmt
);
```

**Parameters**

- `dwOutputStreamIndex`
- `pmt`
  The media type to check

**Return Value**

- S_OK if the specified stream supports the specified media type, else DMOResults.E_InvalidType

**Remarks**

This method is called by the abstract class. The implementor should check the properties of the AMMediaType to ensure that if requested, it can produce an output sample of the specified type.

**See Also**

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
**IMediaObjectImpl.InternalDiscontinuity Method**

(Virtual) Called to notify of a stream discontinuity

```csharp
protected virtual int InternalDiscontinuity(
    int dwInputStreamIndex
);
```

**Parameters**

*dwInputStreamIndex*
- Input stream number

**Return Value**

S_OK for successful operation

**Remarks**

The default implementation assumes no special processing is required for discontinuities.

**See Also**

[IMediaObjectImpl Class] | [MediaObjectTemplate Namespace]
An NDoc Documented Class Library
**IMediaObjectImpl.InternalFlush Method**

(Abstract) Called to flush all pending processing

```csharp
protected abstract int InternalFlush();
```

**Return Value**

S_OK to indicate successful operation

**Remarks**

This method is called by the abstract class. In response, the implementor should discard any pending input buffers.

**See Also**

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
**IMediaObjectImpl.InternalFreeStreamingResources Method**

(Virtual) Allows stream resources to be released

```csharp
protected virtual int InternalFreeStreamingResources();
```

**Return Value**

S_OK for successful operation

**Remarks**

Allows the implementor to release any resources used for performing the processing. The default implementation assumes we don't need to release any addition resources.

**See Also**

[IMediaObjectImpl Class](#) | [MediaObjectTemplate Namespace](#)
An NDoc Documented Class Library
**IMediaObjectImpl.InternalGetCurrentTime Method**

(Virtual) Returns the current time in the media stream

```csharp
protected virtual long InternalGetCurrentTime();
```

**Return Value**

The current time in the media stream

**Remarks**

Typically, this function should be overridden to return the most recent timestamp from the last call to `InternalProcessInput`. It is used to support `IMediaParams.GetParam`. The default implementation assumes the stream has no time stamps or that the stream is stopped.

**See Also**

[IMediaObjectImpl Class](#) | [MediaObjectTemplate Namespace](#)
An NDoc Documented Class Library
**IMediaObjectImpl.InternalGetInputMaxLatency Method**

(Virtual) Retrieves the maximum latency on a specified input stream

```csharp
protected virtual int InternalGetInputMaxLatency(
    int dwInputStreamIndex,
    out long prtMaxLatency
);
```

**Parameters**

- `dwInputStreamIndex`: Input stream number
- `prtMaxLatency`: Latency

**Return Value**

- S_OK for successful completion

**Remarks**

The latency is the difference between a time stamp on the input stream and the corresponding time stamp on the output stream. The maximum latency is the largest possible difference in the time stamps. The default implementation returns E_NOTIMPL indicating the DMO doesn't support reporting latency.

**See Also**

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
IMediaObjectImpl.InternalGetInputSizeInfo Method

(Virtual) Retrieves the buffer requirements for a specified input stream

```csharp
protected virtual int InternalGetInputSizeInfo(
    int dwInputStreamIndex,
    out int pcbSize,
    out int pcbMaxLookahead,
    out int pcbAlignment
);
```

Parameters

`dwInputStreamIndex`
Input stream number

`pcbSize`
Minimum size of an input buffer for this stream, in bytes

`pcbMaxLookahead`
Maximum amount of data that the DMO will hold for lookahead, in bytes

`pcbAlignment`
the required buffer alignment, in bytes. If the input stream has no alignment requirement, the value is 1.

Return Value

S_OK for successful operation

Remarks

The implementator could override this method to specify different values. The default implementation reports that we can accept any alignment, hold no lookahead buffers, and the input buffer must be at least 1 byte long.

See Also

IMediaObjectImpl Class | MediaObjectTemplate Namespace
IMediaObjectImpl.InternalGetInputStreamInfo Method

(Virtual) Controls information about how input buffers are formatted

```csharp
protected virtual int InternalGetInputStreamInfo(int dwInputStreamIndex, out DMOInputStreamInfo pdwFlags);
```

Parameters

dwInputStreamIndex  
Input stream number

pdwFlags  
Flags specifying how input buffers need to be formatted

Return Value

S_OK for successful completion

Remarks

Allows the implementor to specify flags controlling the format of input buffers. The default implementation return FixedSampleSize | SingleSamplePerBuffer | WholeSamples

See Also

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
IMediaObjectImpl.InternalGetInputType Method

(Virtual) Retrieves a preferred media type for a specified input stream

```cpp
protected virtual int InternalGetInputType(
    int dwInputStreamIndex,
    int dwTypeIndex,
    out AMMediaType pmt
);
```

Parameters

- **dwInputStreamIndex**
  - Input stream number

- **dwTypeIndex**
  - Index into the array of supported media types

- **pmt**
  - The media type

Return Value

DMOResults.E_NoMoreItems if out of range or S_OK for successful completion

Remarks

If the implementor supports returning a collection of supported media types, it should override this method. The default implementation assumes we don't enumerate our supported types. The app calling this DMO should just try setting something and see if it works.

See Also

IMediaObjectImpl Class  |  MediaObjectTemplate Namespace
An NDoc Documented Class Library
I.MediaObjectImpl.InternalGetOutputStreamSizeInfo Method

(Abstract) Determine the requirements for the output stream

```csharp
protected abstract int InternalGetOutputStreamSizeInfo
    (int dwOutputStreamIndex,
     out int pcbSize,
     out int pcbAlignment);
```

**Parameters**

- `dwOutputStreamIndex`  
  Output stream number

- `pcbSize`  
  The minimum size of an output buffer for this stream, in bytes

- `pcbAlignment`  
  The required buffer alignment, in bytes. If the output stream has no alignment requirement, the value is 1.

**Return Value**

- S_OK to indicate successful operation.

**Remarks**

This method is called by the abstract class. You should never return zero for the alignment.

**See Also**

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
IMediaObjectImpl::InternalGetOutputStreamInfo Method

(Virtual) Controls information about how output buffers are formatted

```csharp
protected virtual int InternalGetOutputStreamInfo(
    int dwOutputStreamIndex,
    out DM00OutputStreamInfo pdwFlags
);
```

Parameters

- `dwOutputStreamIndex`
- `pdwFlags`
  Flags specifying how output buffers need to be formatted

Return Value

S_OK for successful completion

Remarks

Allows the implementor to specify flags controlling the format of output buffers. The default implementation returns WholeSamples | SingleSamplePerBuffer | FixedSampleSize

See Also

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
**IMediaObjectImpl/InternalGetOutputType Method**

(Virtual) Retrieves a preferred media type for a specified output stream

```csharp
protected virtual int InternalGetOutputType(
    int dwOutputStreamIndex,
    int dwTypeIndex,
    out AMMediaType pmt
);
```

**Parameters**

- **dwOutputStreamIndex**
  Output stream number

- **dwTypeIndex**
  Index into the array of supported media types

- **pmt**
  The media type

**Return Value**

DMOResults.E_NoMoreItems if out of range or S_OK for successful completion

**Remarks**

If the implementor supports returning a collection of supported media types, it should override this method. The default implementation assumes our output type is the same as our input type.

**See Also**

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
(Abstract) Accept input buffers to be processed

```java
protected abstract int InternalProcessInput(
    int dwInputStreamIndex,
    IMediaBuffer pBuffer,
    DMOInputDataBuffer dwFlags,
    long rtTimestamp,
    long rtTimelength
);
```

**Parameters**

- `dwInputStreamIndex`
  Input stream number

- `pBuffer`
  Input buffer to process

- `dwFlags`
  Processing flags

- `rtTimestamp`
  Timestamp of sample(s)

- `rtTimelength`
  Length of sample(s)

**Return Value**

S_OK if the operation completes successfully, S_FALSE if there is no input to process.

**Remarks**

This method is called by the abstract class. It passes the actual data to be process to the implementor. Commonly, the implementor stores these values, waiting for the call to `InternalProcessOutput` (which contains the buffers into which the results are to be stored), at which point they are released.
See Also

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
IMediaObjectImpl.InternalProcessOutput Method

(Abstract) Process the input buffers from a previous call to InternalProcessInput into the provided output buffers

```csharp
protected abstract int InternalProcessOutput(DMOProcessOutput dwFlags,
                                           int cOutputBufferCount,
                                           DMOOutputDataBuffer[] pOutputBuffers,
                                           out int pdwStatus);
```

Parameters

- **dwFlags**
  
  Flags controlling the operation

- **cOutputBufferCount**
  
  The number of buffers provided (one per output stream)

- **pOutputBuffers**
  
  The output buffer into which the data is processed

- **pdwStatus**
  
  Zero

Return Value

  S_FALSE if there is no output, S_OK for successful operation.

Remarks

  This method is called by the abstract class. It passes the output buffers to the implementor. Typically, this is when the actual work is done, processing the input buffers into the output buffers.

See Also

  IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
**IMediaObjectImpl.InternalSetInputMaxLatency Method**

(Virtual) Set the maximum latency on a specified input stream

```cpp
protected virtual int InternalSetInputMaxLatency(
    int dwInputStreamIndex,
    long rtMaxLatency
);
```

**Parameters**

- `dwInputStreamIndex` - Input stream number
- `rtMaxLatency` - Maximum latency

**Return Value**

- S_OK for successful operation

**Remarks**

The default implementation returns E_NOTIMPL indicating the DMO doesn't support reporting latency.

**See Also**

- [IMediaObjectImpl Class](#) | [MediaObjectTemplate Namespace](#)
An NDoc Documented Class Library
**IMediaObjectImpl.Lock Method**

COM entry point for IMediaObject.Lock

```csharp
public int Lock(
    bool lLock
);
```

**Implements**

IMediaObject.Lock

**Remarks**

There should be no need to modify or override this method. It will call the abstract and virtual methods to perform its work.

**See Also**

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
**IMediaObjectImpl.MoCloneMediaType Method**

Make a clone of a media type

```c
protected static AMMediaType MoCloneMediaType,
    AMMediaType pmt1
);
```

**Parameters**

*pmt1*

The AMMediaType to clone

**Return Value**

Returns the clone

**Remarks**

Note that like all AMMediaTypes, the clone must be released with DsUtils.FreeAMMediaType when it is no longer needed.

**See Also**

[IMediaObjectImpl Class](#) | [MediaObjectTemplate Namespace](#)
An NDoc Documented Class Library
IMediaObjectImpl.OutputType Method

Get the AMMediaType for the specified Output stream

```c
protected AMMediaType OutputType(
    int ulOutputStreamIndex
);
```

**Parameters**

*ulOutputStreamIndex*

The stream to get the media type for

**Return Value**

The media type for the stream, or null if not set

**Remarks**

The abstract class will call InternalCheckOutputType to see whether a given media type is supported. To see what media type was actually set, the derived class can call this method.

**See Also**

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
**IMediaObjectImpl.OutputTypeSet Method**

Check whether the media type is set for the specified output stream

```cpp
protected bool OutputTypeSet(
    int ulOutputStreamIndex
);
```

**Parameters**

*ulOutputStreamIndex*
Zero based stream number to check

**Return Value**
true if the stream type is set

**See Also**

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
**IMediaObjectImpl.ParamCalcValueForTime Method**

Given a parameter number and a time, return the parameter value at that time.

```cpp
protected MPData ParamCalcValueForTime(
    int dwParam,
    long rtTimeStamp
);
```

**Parameters**

- **dwParam**
  - Zero based parameter number

- **rtTimeStamp**
  - Time

**Return Value**

Calculated value for the specified time

**Remarks**

Parameters for DMO can be set in one of two ways. IMediaParams.SetParam can be used to set a parameter to a specific value. It is useful for setting values that aren't intended to change over time. There is also IMediaParams.AddEnvelope. This method can be use for things that change over time. For example, consider a parameter for adjusting the audio volume. You might want to be able to have the volume go from 0% to 150% over the first x seconds.

You can easily support both by using this method. As you prepare to process buffers, take the timestamp that applies to that buffer, and call this method to get the desired value for that parameter at that time.

**See Also**

[IMediaObjectImpl Class](#) | [MediaObjectTemplate Namespace](#)
An *NDoc Documented Class Library*
**IMediaObjectImpl.ParamDefine Method**

Create a definition for a parameter that is accessible thru IMediaParamInfo and IMediaParams.

```csharp
protected void ParamDefine(
    int iParamNum,
    ParamInfo p,
    string sText
);
```

**Parameters**

- **iParamNum**
  - Zero based parameter number to set the definition for

- **p**
  - Populated ParamInfo struct

- **sText**
  - Format string (described in MSDN under IMediaParamInfo::GetParamText)

**Remarks**

This method should be called from the constructor of the class that implements IMediaObjectImpl. It defines a single parameter that can be set on the DMO. You must call it once for each of the parameters defined in the call to the IMediaObjectImpl constructor. This allows for automatic support of the IMediaParamInfo and IMediaParams methods. See the [ParamCalcValueForTime](#) for additional details.

**See Also**

- IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
**IMediaObjectImpl.ProcessInput Method**

COM entry point for IMediaObject.ProcessInput

```java
public int ProcessInput(
    int ulStreamIndex,
    IMediaBuffer pBuffer,
    DMOInputDataBuffer dwFlags,
    long rtTimestamp,
    long rtTimelength
);
```

**Implements**

IMediaObject.ProcessInput

**Remarks**

There should be no need to modify or override this method. It will call the abstract and virtual methods to perform its work.

**See Also**

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
**IMediaObjectImpl.ProcessOutput Method**

COM entry point for IMediaObject.ProcessOutput

```java
public int ProcessOutput(
    DMOProcessOutput dwFlags,
    int ulOutputBufferCount,
    DMOOutputDataBuffer[] pOutputBuffers,
    out int pdwStatus
);
```

**Implements**

IMediaObject.ProcessOutput

**Remarks**

There should be no need to modify or override this method. It will call the abstract and virtual methods to perform its work.

**See Also**

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An *NDoc Documented Class Library*
**IMediaObjectImpl.SetInputMaxLatency Method**

COM entry point for IMediaObject.SetInputMaxLatency

```csharp
public int SetInputMaxLatency(
    int ulStreamIndex,
    long rtLatency
);
```

**Implements**

IMediaObject.SetInputMaxLatency

**Remarks**

There should be no need to modify or override this method. It will call the abstract and virtual methods to perform its work.

**See Also**

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
**IMediaObjectImpl.SetInputType Method**

COM entry point for IMediaObject.SetInputType

```csharp
public int SetInputType(
    int uiStreamIndex,
    AMMediaType pmt,
    DMOSetType dwFlags
);
```

**Implements**

[IMediaObject.SetInputType](#)

**Remarks**

There should be no need to modify or override this method. It will call the abstract and virtual methods to perform its work.

**See Also**

[IMediaObjectImpl Class](#) | [MediaObjectTemplate Namespace](#)
An NDoc Documented Class Library
**IMediaObjectImpl.SetOutputType Method**

COM entry point for IMediaObject.SetOutputType

```csharp
public int SetOutputType(
    int ulStreamIndex,
    AMMediaType pmt,
    DMOSetType dwFlags
);
```

**Implements**

[IMediaObject.SetOutputType](#)

**Remarks**

There should be no need to modify or override this method. It will call the abstract and virtual methods to perform its work.

**See Also**

[IMediaObjectImpl Class](#) | [MediaObjectTemplate Namespace](#)
An NDoc Documented Class Library
**IMediaObjectImpl.SetParam Method**

COM entry point for IMediaParams.SetParam

```csharp
public int SetParam(
    int dwParamIndex,
    MPData value
);
```

**Implements**

IMediaParams.SetParam

**Remarks**

There should be no need to modify or override this method. See ParamDefine and ParamCalcValueForTime for details about how this works.

**See Also**

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
**IMediaObjectImpl.SetTimeFormat Method**

COM entry point for IMediaParams.SetTimeFormat

```csharp
public int SetTimeFormat(
    Guid guidTimeFormat,
    int mpTimeData
);
```

**Implements**

IMediaParams.SetTimeFormat

**Remarks**

There should be no need to modify or override this method. See ParamDefine and ParamCalcValueForTime for details about how this works.

**See Also**

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
Check to see if two Media Types are exactly the same

```csharp
protected static bool TypesMatch(
    AMMediaType pmt1,
    AMMediaType pmt2
);
```

**Parameters**

- `pmt1`  
  Media type to compare

- `pmt2`  
  Media type to compare

**Return Value**

true if types are identical, else false

**See Also**

IMediaObjectImpl Class | MediaObjectTemplate Namespace
An NDoc Documented Class Library
IMediaObjectImpl.TimeFormatFlags Enumeration

Used by the IMediaObjectImpl constructor to specify which timeformats are supported

This enumeration has a FlagsAttribute attribute that allows a bitwise combination of its member values.

```csharp
protected enum IMediaObjectImpl.TimeFormatFlags
```

Members

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Used only when the DMO has no parameters</td>
<td>0</td>
</tr>
<tr>
<td>Reference</td>
<td>Reference time, in 100-nanosecond units. All DMOs should support this format.</td>
<td>1</td>
</tr>
<tr>
<td>Music</td>
<td>Music time, in parts per quarter note.</td>
<td>2</td>
</tr>
<tr>
<td>Samples</td>
<td>Samples per second.</td>
<td>4</td>
</tr>
</tbody>
</table>

Requirements

Namespace: MediaObjectTemplate

Assembly: DmoFlip (in DmoFlip.dll)

See Also

MediaObjectTemplate Namespace
An NDoc Documented Class Library
**MediaObjectTemplate Hierarchy**

**System.Object**

  **MediaObjectTemplate.IMediaObjectImpl**
  **DirectShowLib.DMO.IMediaObject**,  
  **DirectShowLib.DMO.IMediaParamInfo**,  
  **DirectShowLib.DMO.IMediaParams**

**System.ValueType**

  **System.Enum**
  **System.IComparable**,  
  **System.IConvertible**,  
  **System.IFormattable**

  **MediaObjectTemplate.IMediaObjectImpl.TimeFormatFlags**

**See Also**

  **MediaObjectTemplate Namespace**