

Capture Manager SDK - new solution for capturing live video and audio from web cameras on Windows OS

Capture Manager SDK allows you include into your software solutions easy supporting of working with web cameras. This SDK is based on Microsoft Media Foundation which is included into the regular Windows OS editions.

Microsoft Media Foundation is new technology of Microsoft which was introduced since Windows Vista, but only since Windows 7 it has got supporting of interfaces for capturing video and audio from web cameras.

Capture Manager SDK is presented in two versions:

1. [Freeware](#)
2. [Commercial](#)

ILogPrintOut

ILogPrintOut – it manages writing of log information by the next methods:

HRESULT setVerbose(DWORD aLevelType, BSTR aFilePath, BOOL aState) - state of print out of log, where **aLevelType** is constant of [LogLevel](#), **aFilePath** is the full file path to log file, **aState** is state of enabling or disabling write info to log file;

HRESULT addPrintOutDestination(DWORD aLevelType, BSTR aFilePath) - add print out destination, where **aLevelType** is constant of [LogLevel](#), **aFilePath** is the full file path to log file;

HRESULT removePrintOutDestination(DWORD aLevelType, BSTR aFilePath) – remove print out destination, where **aLevelType** is constant of [LogLevel](#), **aFilePath** is the full file path to log file;

Constants for logging.

LogLevel – it is list of constants which are define level of type information which must be written in log output.

INFO_LEVEL = 0 - log general information level;

ERROR_LEVEL = **INFO_LEVEL** + 1 - log error information level.

ICaptureManagerControl

ICaptureManagerControl – it manages creating of all capturing control objects and miscellaneous objects by the next methods:

HRESULT createControl(REFIID aREFIID, IUnknown aPtrPtrControl)**
– create a control, where **aREFIID** is IID of the interface of the needed control object ([**ISourceControl**](#), [**ISinkControl**](#), [**IEncoderControl**](#), [**ISessionControl**](#), [**IStreamControl**](#)), **aPtrPtrControl** is pointer on the created object;

HRESULT createMisc(REFIID aREFIID, IUnknown aPtrPtrMisc)** -
create a miscellaneous object, where **aREFIID** is IID of the interface of the needed miscellaneous object ([**IMediaTypeParser**](#), [**IStrideForBitmap**](#), [**IVersionControl**](#), [**IEVRStreamControl**](#), [**IRenderingControl**](#)), **aPtrPtrMisc** is pointer on the created object;

ISourceControl

ISourceControl – it manages and creates sources of video and audio data by the next methods:

HRESULT getCollectionOfSources(BSTR* aPtrPtrXMLstring) - get collection of sources in XML form ([SourceXMLDocument](#)), where **aPtrPtrXMLstring** is a pointer on the create string;

HRESULT getSourceOutputMediaType(BSTR aSymbolicLink, DWORD aIndexStream, DWORD aIndexMediaType, IUnknown aPtrPtrOutputMediaType)** - get pointer on output media type of the source, where **aSymbolicLink** is a pointer on the unique symbolic link string of CaptureManager source, **aIndexStream** is index of stream in selected source, **aIndexMediaType** is index of media type in the selected stream in the selected source, **aPtrPtrOutputMediaType** is a pointer on the selected output media type of the selected source;

HRESULT createSourceNode(BSTR aSymbolicLink, DWORD aIndexStream, DWORD aIndexMediaType, IUnknown aPtrPtrTopologyNode)** - create source node from defined source settings, where **aSymbolicLink** is a pointer on the unique symbolic link string of CaptureManager source, **aIndexStream** is index of stream in selected source, **aIndexMediaType** is index of media type in the selected stream in the selected source, **aPtrPtrTopologyNode** is a pointer on the topology node which contains the selected source;

HRESULT createSourceNodeWithDownStreamConnection(BSTR aSymbolicLink, DWORD aIndexStream, DWORD aIndexMediaType, IUnknown* aPtrDownStreamTopologyNode, IUnknown aPtrPtrTopologyNode)** - create source node from defined source settings with down stream connection, where **aSymbolicLink** is a pointer on the unique symbolic link string of CaptureManager source, **aIndexStream** is index of stream in selected source, **aIndexMediaType** is index of media type in the

*selected stream in the selected source, **aPtrDownStreamTopologyNode** is a pointer on the topology node which is connected with the selected source, **aPtrPtrTopologyNode** is a pointer on the topology node which contains the selected source;*

HRESULT createSource(BSTR aSymbolicLink, IUnknown aPtrPtrMediaSource)** - *get MediaSource by Symbolic Link, where **aSymbolicLink** is a pointer on the unique symbolic link string of CaptureManager source, **aPtrPtrMediaSource** is a pointer on the selected source;*

HRESULT createSourceFromCaptureProcessor(IUnknown* aPtrCaptureProcessor, IUnknown aPtrPtrMediaSource)** - *get MediaSource by defined customized object, where **aPtrCaptureProcessor** is pointer on the customized object with [ICaptureProcessor](#) interface, **aPtrPtrMediaSource** is a pointer on the selected source;*

HRESULT getSourceOutputMediaTypeFromMediaSource(IUnknown* aPtrMediaSource, DWORD aIndexStream, DWORD aIndexMediaType, IUnknown aPtrPtrOutputMediaType)** - *get pointer on output media type from defined MediaSource and source settings, where **aPtrMediaSource** is the pointer on the selected source, **aIndexStream** is index of stream in selected source, **aIndexMediaType** is index of media type in the selected stream in the selected source, **aPtrPtrOutputMediaType** is a pointer on the selected output media type of the selected source;*

HRESULT createSourceNodeFromExternalSource(IUnknown* aPtrMediaSource, DWORD aIndexStream, DWORD aIndexMediaType, IUnknown aPtrPtrTopologyNode)** - *create source node from defined MediaSource and source settings, where **aPtrMediaSource** is the pointer on the selected source, **aIndexStream** is index of stream in selected source, **aIndexMediaType** is index of media type in the selected stream in the selected source, **aPtrPtrTopologyNode** is a pointer on the topology node which contains the selected source;*

HRESULT createSourceNodeFromExternalSourceWithDownStreamConnection(

IUnknown* aPtrMediaSource, DWORD aIndexStream, DWORD aIndexMediaType, IUnknown* aPtrDownStreamTopologyNode, IUnknown aPtrPtrTopologyNode)** - create source node from defined *MediaSource* and source settings with down stream connection, where **aPtrMediaSource** is the pointer on the selected source, **aIndexStream** is index of stream in selected source, **aIndexMediaType** is index of media type in the selected stream in the selected source, **aPtrDownStreamTopologyNode** is a pointer on the topology node which is connected with the selected source, **aPtrPtrTopologyNode** is a pointer on the topology node which contains the selected source;

HRESULT createSourceControl(BSTR aSymbolicLink, REFIID aREFIID, IUnknown aPtrPtrControl)** - create control for the source, where **aSymbolicLink** is a pointer on the unique symbolic link string of *CaptureManager* source, **aREFIID** is IID of interface of the needed source control ([**IWebCamControl**](#)), **PtrPtrControl** is a pointer on the control object;

SourceXMLDocument

SourceXMLDocument – it is XML document which define the current properties of CaptureManager media sources.

Node "**Sources**" is declaration of the root XML node.

Node "**Source**" is declaration of one node of media source which contains information about the current source. Node "**Source.Attributes**" declares list of "**Attribute**" nodes.

Node "**Attribute**" presents information about source. It includes attribute "**Name**" for symbolic name of node, attribute "**GUID**" for unique id of node, attribute "**Title**" for friendly name of node, attribute "**Description**" for more informative description of node.

Node "**PresentationDescriptor**" includes information about audio and video streams. Node "PresentationDescriptor.Attributes" declares list of "**Attribute**" nodes.

Node "**StreamDescriptor**" includes information about one media stream. It includes attribute "**Index**" for unique id of the media stream, attribute "**MajorType**" for symbolic name of media type, attribute "**MajorTypeGUID**" for unique id of media type. Node "**StreamDescriptor.Attributes**" declares list of "**Attribute**" nodes.

Node "**MediaTypes**" includes information about media types of the one media stream. It includes attribute "**TypeCount**" for amount of media types, which are supported by this stream.

Node "**MediaType**" includes information about one media type of the one media stream. It includes attribute "**Index**" which is unique index of current media type.

Node "**MediaTypeItem**" includes information about one media property of one media type. It includes attribute "**Name**" which is unique symbolic name of

current property, attribute "**GUID**" which is unique id of current property, attribute "**Title**" which is friendly name of current property, attribute "**Description**" which is friendly description of current property.

Node "**SingleValue**" includes information about single value of media type property in attribute "**Value**".

Node "**Value.ValueParts**" includes information about one or more values which define media type property. Each value is defined by node "**ValuePart**" with attribute "**Title**" - is friendly title of value part and attribute "**Value**"

Node "**RatioValue**" includes information about one rational value which define media type property, which is presented in attribute "**Value**" in float format. This node includes two "**ValuePart**" nodes for for presenting rational value by ration of "**Numerator**" and "**Denominator**".

```
<?xml version="1.0" ?>
<!--XML Document of sources-->
<Sources>
  <Source>
    <Source.Attributes>
    </Source.Attributes>
    <PresentationDescriptor StreamCount="">
    <PresentationDescriptor.Attributes>
    </PresentationDescriptor.Attributes>
    <StreamDescriptor Index="" MajorType="" MajorTypeGUID="">
    <StreamDescriptor.Attributes>
    </StreamDescriptor.Attributes>
    </StreamDescriptor>
    <MediaTypes TypeCount="">
    <MediaType Index="">
    <MediaTypeItem Name="" GUID="" Title="" Description="">
    </MediaTypeItem>
    </MediaType>
    </MediaTypes>
    </PresentationDescriptor>
  </Source>
</Sources>
```

```
<Attribute Name="" GUID="" Title="" Description="">
</Attribute>
<SingleValue Value=""/>
<Value.ValueParts>
<ValuePart Title="" Value=""/>
</Value.ValueParts>
<RatioValue Value="">
<Value.ValueParts>
<ValuePart Title="Numerator" Value=""/>
<ValuePart Title="Denominator" Value=""/>
</Value.ValueParts>
</RatioValue>
<Value.ValueParts>
<ValuePart Title="Width" Value=""/>
<ValuePart Title="Height" Value=""/>
</Value.ValueParts>
<Value.ValueParts>
<ValuePart Title="Width" Value=""/>
<ValuePart Title="Height" Value=""/>
<ValuePart Title="OffsetX" Value=""/>
<ValuePart Title="OffsetY" Value=""/>
</Value.ValueParts>
```

IWebCamControl

IWebCamControl – it manages options of web camera by the next methods:

HRESULT getCamParametr(BSTR* aXMLstring) – get web camera parametr in XML format ([WebCamParametrXMLDocument](#)), where **aXMLstring** is a pointer on the created string;

HRESULT getCamParametr(DWORD aParametrIndex, LONG* aCurrentValue, LONG* aMin, LONG* aMax, LONG* aStep, LONG* aDefault, LONG* aFlag) - get parametr of the source, where [aParametrIndex](#) is index of the current parametr, **aCurrentValue** is a pointer on the current value, **aMin** is a pointer on the minimum accessible value, **aMax** is a pointer on the maximum accessible value, **aStep** is a pointer on the accessible change step value, **aDefault** is a pointer on the default value of parametr from device firmware, [aFlag](#) is a pointer on the flag value of the parametr;

HRESULT setCamParametr(DWORD aParametrIndex, LONG aNewValue, LONG aFlag) - set parametr to the source, where [aParametrIndex](#) is index of the current parametr, **aNewValue** is the new value, [aFlag](#) is the new flag value of the parametr;

WebCamParametrXMLDocument

WebCamParametrXMLDocument – it is XML document which define the current properties of web camera parametr.

Node "**Parameters**" is declaration of the root XML node.

Node "**Group**" is declaration of parametr which is grouped by type, which is defined by attribute "**Title**":

- **Video processor** - video processor parametr;
- **Camera control** - camera driver parametr.

Node "**Parametr**" is declaration of the parametr node.

Attribute "**Index**" is unique index of the parametr, attribute "**Title**" defines friendly name of the parametr, attribute "**CurrentValue**" defines current value of the parametr, attribute "**Default**" defines default value of the parametr, attribute "**Min**" defines minimum acceptable value of the parametr, attribute "**Max**" defines maximum acceptable value of the parametr, attribute "**Step**" defines value of changing between minimum and maximum values of the parametr, attribute "**Flag**" defines current flag type of control of the parametr.

```
<?xml version="1.0" ?>
<!--XML Document of web cam parametr-->
<Parameters>
  <Group Title="Video processor" GUID="">
    <Parametr Index="" Title="" CurrentValue="" Default="" Min=""
Max="" Step="" Flag=""/>
  </Group>
  <Group Title="Camera control" GUID="">
    <Parametr Index="" Title="" CurrentValue="" Default="" Min=""
Max="" Step="" Flag=""/>
  </Group>
</Parameters>
```

Constants for flags of web camera parameters

WebCamParametrFlag – it is list of constants which define enabling of the parametr control.

Auto = 1 - enable control of parametr by device;

Manual = 2 - enable control of parametr by user.

Constants for reference of web camera parametr

WebCamParametr – it is list of constants which define web camera parametr.

BRIGHTNESS = 0;
CONTRAST = BRIGHTNESS + 1;
HUE = CONTRAST + 1;
SATURATION = HUE + 1;
SHARPNESS = SATURATION + 1;
GAMMA = SHARPNESS + 1;
COLORENABLE = GAMMA + 1;
WHITRBALANCE = COLORENABLE + 1;
BACKLIGHTCOMPENSATION = WHITRBALANCE + 1;
GAIN = BACKLIGHTCOMPENSATION + 1;
PAN = GAIN + 1;
TILT = PAN + 1;
ROLL = TILT + 1;
ZOOM = ROLL + 1;
EXPOSURE = ZOOM + 1;
IRIS = EXPOSURE + 1 ;
FOCUS = IRIS + 1.

ICaptureProcessor

ICaptureProcessor – it is a callback interface for customized code which injects data into the recording pipeline by the next methods:

HRESULT STDMETHODCALLTYPE start(LONGLONG aStartPositionInHundredNanosecondUnits, REFGUID aGUIDTimeFormat) - *callback method for starting of the capture processor, where **aStartPositionInHundredNanosecondUnits** is time offset, **aGUIDTimeFormat** is type of time format (**GUID_NULL**);*
HRESULT STDMETHODCALLTYPE stop() - *callback method for stopping of the capture processor;*

HRESULT STDMETHODCALLTYPE pause() - *callback method for pausing of the capture processor;*

HRESULT STDMETHODCALLTYPE shutdown() - *callback method for releasing of the resource of the capture processor;*

HRESULT STDMETHODCALLTYPE initilaize(IUnknown *aPtrIInitilaizeCaptureSource) - *callback method for initilaizing of the resource of the capture processor, where **aPtrIInitilaizeCaptureSource** is pointer on object with [IInitilaizeCaptureSource](#) interface;*

HRESULT STDMETHODCALLTYPE setCurrentMediaType(IUnknown *aPtrICurrentMediaType) - *callback method for the selected media type of the capture processor, where **aPtrICurrentMediaType** is pointer on object with [ICurrentMediaType](#) interface;*

HRESULT STDMETHODCALLTYPE sourceRequest(IUnknown *aPtrISourceRequestResult) - *callback method for injection data to the capture processor, where **aPtrISourceRequestResult** is pointer on object with [ISourceRequestResult](#) interface.*

IInitilaizeCaptureSource

IInitilaizeCaptureSource – it sets source configuration by the next method:

HRESULT STDMETHODCALLTYPE setPresentationDescriptor(BSTR aXMLPresentationDescriptor) - set XML format presentation descriptor ([PresentationDescriptorXMLDocument](#)), where **aXMLPresentationDescriptor** is a pointer on the created string.

PresentationDescriptorXMLDocument

PresentationDescriptorXMLDocument – it is XML document which describes the source's configuration.

Node "**PresentationDescriptor**" is declaration of the root XML node, it has attribute "**StreamCount**" which defines amount of media streams - it supports one video stream. The next document presents template document for video. It has one media type with the next templates: "**Temp_Width**" - for replacement width of video frame in pixels, "**Temp_Height**" - for replacement height of video frame in pixels, "**Temp_Stride**" - for replacement width of video frame in bytes (can be computed by [IStrideForBitmap](#)), "**Temp_SampleSize**" - for replacement size of video frame in bytes, "**Temp_SubTypeGUID**" - for replacement GUID of video frame.

```
<?xml version='1.0' encoding='UTF-8'?>
<PresentationDescriptor StreamCount='1'>
  <PresentationDescriptor.Attributes Title='Attributes of Presentation'>
    <Attribute
Name='MF_DEVSOURCE_ATTRIBUTE_SOURCE_TYPE_VIDCAP_SYMBOLIC
GUID='{58F0AAD8-22BF-4F8A-BB3D-D2C4978C6E2F}' Title='The
symbolic link for a video capture driver.' Description='Contains the unique
symbolic link for a video capture driver.'>
      <SingleValue Value='ImageCaptureProcessor' />
    </Attribute>
    <Attribute
Name='MF_DEVSOURCE_ATTRIBUTE_FRIENDLY_NAME'
GUID='{60D0E559-52F8-4FA2-BBCE-ACDB34A8EC01}' Title='The display
name for a device.' Description='The display name is a human-readable string,
suitable for display in a user interface.'>
      <SingleValue Value='Image Capture Processor' />
    </Attribute>
  </PresentationDescriptor.Attributes>
  <StreamDescriptor Index='0' MajorType='MFMediaType_Video'
```

```
MajorTypeGUID='{73646976-0000-0010-8000-00AA00389B71}'>
  <MediaTypes TypeCount='1'>
    <MediaType Index='0'>
      <MediaTypeInfo Name='MF_MT_FRAME_SIZE'
GUID='{1652C33D-D6B2-4012-B834-72030849A37D}' Title='Width and
height of the video frame.' Description='Width and height of a video frame, in
pixels.'>
        <Value.ValueParts>
          <ValuePart Title='Width' Value='Temp_Width' />
          <ValuePart Title='Height' Value='Temp_Height' />
        </Value.ValueParts>
      </MediaTypeInfo>
      <MediaTypeInfo Name='MF_MT_AVG_BITRATE'
GUID='{20332624-FB0D-4D9E-BD0D-CBF6786C102E}' Title='Approximate
data rate of the video stream.' Description='Approximate data rate of the video
stream, in bits per second, for a video media type.'>
        <SingleValue Value='33570816' />
      </MediaTypeInfo>
      <MediaTypeInfo Name='MF_MT_MAJOR_TYPE'
GUID='{48EBA18E-F8C9-4687-BF11-0A74C9F96A8F}' Title='Major type
GUID for a media type.' Description='The major type defines the overall
category of the media data.'>
        <SingleValue Value='MFMedia_Type_Video'
GUID='{73646976-0000-0010-8000-00AA00389B71}' />
      </MediaTypeInfo>
      <MediaTypeInfo Name='MF_MT_DEFAULT_STRIDE'
GUID='{644B4E48-1E02-4516-B0EB-C01CA9D49AC6}' Title='Default
surface stride.' Description='Default surface stride, for an uncompressed video
media type. Stride is the number of bytes needed to go from one row of pixels to
the next.'>
        <SingleValue Value='Temp_Stride' />
      </MediaTypeInfo>
      <MediaTypeInfo Name='MF_MT_FIXED_SIZE_SAMPLES'
GUID='{B8EBEFAF-B718-4E04-B0A9-116775E3321B}' Title='The fixed size
of samples in stream.' Description='Specifies for a media type whether the
samples have a fixed size.'>
        <SingleValue Value='True' />
      </MediaTypeInfo>
    </MediaType>
  </MediaTypes>
</MajorType>
```

```

</MediaTypeItem>
  <MediaTypeItem Name='MF_MT_FRAME_RATE'
GUID='{C459A2E8-3D2C-4E44-B132-FEE5156C7BB0}' Title='Frame rate.'
Description='Frame rate of a video media type, in frames per second.'>
  <RatioValue Value='10.0'>
    <Value.ValueParts>
      <ValuePart Title='Numerator' Value='10' />
      <ValuePart Title='Denominator' Value='1' />
    </Value.ValueParts>
  </RatioValue>
</MediaTypeItem>
  <MediaTypeItem Name='MF_MT_PIXEL_ASPECT_RATIO'
GUID='{C6376A1E-8D0A-4027-BE45-6D9A0AD39BB6}' Title='Pixel aspect
ratio.' Description='Pixel aspect ratio for a video media type.'>
  <RatioValue Value='1'>
    <Value.ValueParts>
      <ValuePart Title='Numerator' Value='1' />
      <ValuePart Title='Denominator' Value='1' />
    </Value.ValueParts>
  </RatioValue>
</MediaTypeItem>
  <MediaTypeItem
Name='MF_MT_ALL_SAMPLES_INDEPENDENT' GUID='{C9173739-
5E56-461C-B713-46FB995CB95F}' Title='Independent of samples.'
Description='Specifies for a media type whether each sample is independent of
the other samples in the stream.'>
  <SingleValue Value='True' />
</MediaTypeItem>
  <MediaTypeItem Name='MF_MT_SAMPLE_SIZE'
GUID='{DAD3AB78-1990-408B-BCE2-EBA673DACC10}' Title='The fixed
size of each sample in stream.' Description='Specifies the size of each sample, in
bytes, in a media type.'>
  <SingleValue Value='Temp_SampleSize' />
</MediaTypeItem>
  <MediaTypeItem Name='MF_MT_INTERLACE_MODE'
GUID='{E2724BB8-E676-4806-B4B2-A8D6EFB44CCD}' Title='Describes
how the frames are interlaced.' Description='Describes how the frames in a video

```

media type are interlaced.'>

<SingleValue Value='MFVideoInterlace_Progressive' />

</MediaTypeInfoItem>

<MediaTypeInfoItem Name='MF_MT_SUBTYPE'

GUID='{F7E34C9A-42E8-4714-B74B-CB29D72C35E5}' Title='Subtype GUID for a media type.' Description='The subtype GUID defines a specific media format type within a major type.'>

<SingleValue GUID='{Temp_SubTypeGUID}' />

</MediaTypeInfoItem>

</MediaType>

</MediaTypes>

</StreamDescriptor>

</PresentationDescriptor>

ICurrentMediaType

ICurrentMediaType – it allows to get the current media type by the next method:

HRESULT STDMETHODCALLTYPE getMediaTypeIndex(DWORD *aPtrMediaTypeIndex) - get the selected media type index, where *aPtrMediaTypeIndex* is pointer on the index of the selected media type;

HRESULT STDMETHODCALLTYPE getStreamIndex(DWORD *aPtrStreamIndex) - get the selected stream index, where *aPtrStreamIndex* is pointer on the index of the selected stream;

HRESULT STDMETHODCALLTYPE getMediaType(IUnknown **aPtrMediaType) - get the current media type, where *aPtrMediaType* is pointer on the selected media type.

ISourceRequestResult

ISourceRequestResult – it allows inject data into the capture processor by the next method:

HRESULT STDMETHODCALLTYPE setData(LPVOID aPtrData, DWORD aByteSize, BOOL aIsKeyFrame) - send data, where **aPtrData** is pointer on raw data, **aByteSize** is size of raw data, **aIsKeyFrame** is a state of uncompressed data;

HRESULT STDMETHODCALLTYPE getStreamIndex(DWORD *aPtrStreamIndex) - get the current media stream index, where **aPtrStreamIndex** is index of the current stream.

ISinkControl

ISinkControl – it manages and creates sinks of video and audio streams signals by the next methods:

HRESULT getCollectionOfSinks(BSTR* aPtrPtrXMLstring) - get
collection of sinks in XML form ([SinkXMLDocument](#)),

where **aPtrPtrXMLstring** is a pointer on the created string;

**HRESULT createSinkFactory(REFGUID aRefContainerTypeGUID,
REFIID aREFIID, IUnknown** aPtrPtrSink) - create sink factory,**

where **aRefContainerTypeGUID** is IID of type sink, **aREFIID** is IID of one of
the sink factory interfaces ([IFileSinkFactory](#),

[ISampleGrabberCallSinkFactory](#), [ISampleGrabberCallbackSinkFactory](#),

[IEVRSinkFactory](#), [IByteStreamSinkFactory](#), [IEVRMultiSinkFactory](#)), **aPtrPtr**
is a pointer on created object.

SinkFactoryXMLDocument

SinkFactoryXMLDocument – it is XML document which defines CaptureManager output sink factories.

Node "**SinkFactories**" is declaration of the root XML node.

Node "**SinkFactory**" is declaration of one node of sink factory which defines type of sink: file sink, sample grabber, byte stream sink, video renderer.

information about the current source. Attribute "**Name**" defines symbolic name of sink type and attribute "**GUID**" defines unique GUID of sink factory, attribute "**Title**" defines friendly description of type sink factory:

- **SampleGrabberCallbackSinkFactory**
- {3D64C48E-EDA4-4EE1-8436-58B64DD7CF13} - *Sample grabber callback sink factory*
- **SampleGrabberCallSinkFactory**
- {759D24FF-C5D6-4B65-8DDF-8A2B2BECDE39} - *Sample grabber call sink factory*
- **CMVRMultiSinkFactory**
- {A2224D8D-C3C1-4593-8AC9-C0FCF318FF05} - *CaptureManager Video Renderer multi sink factory*
- **FileSinkFactory**
- {D6E342E3-7DDD-4858-AB91-4253643864C2} - *File sink factory*
- **EVRSinkFactory**
- {2F34AF87-D349-45AA-A5F1-E4104D5C458E} - *Enhanced Video Renderer sink factory*
- **ByteStreamSinkFactory**
- {2E891049-964A-4D08-8F36-95CE8CB0DE9B} - *Byte stream sink factory*
- **EVRMultiSinkFactory**
- {10E52132-A73F-4A9E-A91B-FE18C91D6837} - *Enhanced Video Renderer multi sink factory*

Node "**Value.ValueParts**" is root of sink format list. Node "**ValuePart**" is node of sink factory supported format which has attribute "**Title**" - friendly description of format, attribute "**Value**" - symbolic link of format, attribute "**MIME**" - an Internet standard that extends the format, attribute "**Description**" - friendly description of format, attribute "**MaxPortCount**" - maximum incoming streams, attribute "**GUID**" - defines unique GUID of format.

```
<?xml version="1.0" ?>
<!--XML Document of sink factories-->
<SinkFactories>
```



```
<SinkFactory Name="" GUID="" Title="">
  <Value.ValueParts>
    <ValuePart Title="" Value="" MIME="" Description=""
MaxPortCount="" GUID=""/>
  </Value.ValueParts>
</SinkFactory>
</SinkFactories>
```

IFileSinkFactory

IFileSinkFactory – it creates media sink nodes which are linked with media file by the next methods:

**HRESULT createOutputNodes(VARIANT
aArrayPtrCompressedMediaTypes, BSTR aPtrFileName, VARIANT*
aPtrArrayPtrTopologyOutputNodes)** - create output nodes of container by file name, where **aArrayPtrCompressedMediaTypes** is an array of media types of the written formats, **aPtrFileName** is full path to output media file, **aPtrArrayPtrTopologyOutputNodes** is an array of topology nodes for writing of encoded media data.

ISampleGrabberCallSinkFactory

ISampleGrabberCallSinkFactory – it creates media sink for grabbing of one sample by direct calling object with interface ISessionCallback by the next methods:

HRESULT createOutputNode(REFGUID aRefMajorType, REFGUID aRefSubType, DWORD aSampleByteSize, REFIID aREFIID, IUnknown aPtrPtrISampleGrabberCall)** - create output node, where **aRefMajorType** is type of data (**MFMediaType_Video**, **MFMediaType_Audio**), **aRefSubType** is format of data, **aSampleByteSize** is expected size of data, **aREFIID** is IID of interface of the sample grabber (**[ISampleGrabberCall](#)**), **aPtrPtrISampleGrabberCall** is a pointer on created object.

ISampleGrabberCall

ISampleGrabberCall – it manages grabbing of one sample by the next methods:

HRESULT readData(LPVOID aPtrData, DWORD* aByteSize) - get grabbed sample data, where **aPtrData** is a pointer on writing data, **aByteSize** is a pointer on wrote amount variable.

ISampleGrabberCallbackSinkFactory

ISampleGrabberCallbackSinkFactory – it creates media sink for grabbing of one sample by calling object with interface ISampleGrabberCallback from CaptureManager inner thread by the next methods:

HRESULT createOutputNode(REFGUID aRefMajorType, REFGUID aRefSubType, IUnknown* aPtrISampleGrabberCallback, IUnknown aPtrPtrTopologyNode)** - create output node, , where **aRefMajorType** is type of data (**MFMediaType_Video**, **MFMediaType_Audio**), **aRefSubType** is format of data, **aPtrISampleGrabberCallback** is a pointer on object with an interface of the sample grabber callback ([**ISampleGrabberCallback**](#)), **aPtrPtrTopologyNode** is a pointer on created topology node.

ISampleGrabberCallback

ISampleGrabberCallback – it manages grabbing of one sample from CaptureManager inner thread by the next methods:

HRESULT invoke(REFGUID aGUIDMajorMediaType, DWORD aSampleFlags, LONGLONG aSampleTime, LONGLONG aSampleDuration, LPVOID aPtrSampleBuffer, DWORD aSampleSize) - method for callback invoking, where **aGUIDMajorMediaType** is data type (**MFMediaType_Video**, **MFMediaType_Audio**), **aSampleFlags** is system sample flags, **aSampleTime** is current sample time (100 nsec), **aSampleDuration** is current sample duration time (100 nsec), **aPtrSampleBuffer** is a pointer on the sample raw data, **aSampleSize** is a size of the sample raw data.

IEVRSinkFactory

IEVRSinkFactory – it creates media sink for rendering of video stream by the next methods:

HRESULT createOutputNode(LPVOID aHWND, IUnknown **aPtrPtrTopologyNode) - create output node, where **aHWND** is a handler on target window, **aPtrPtrTopologyNode** is a pointer on created topology node.

IEVRMultiSinkFactory

IEVRMultiSinkFactory – it creates array of media sinks for rendering of video streams by the next methods:

HRESULT STDMETHODCALLTYPE createOutputNodes(LPVOID aHandle, IUnknown *aPtrUnkTarget, DWORD aOutputNodeAmount, VARIANT *aPtrArrayPtrTopologyOutputNodes) - create output nodes, where **aHandle** is handler, **aPtrUnkTarget** is a pointer on redering target surface, **aOutputNodeAmount** is an amount of output nodes, **aPtrArrayPtrTopologyOutputNodes** is an array of output rendering topology nodes.

IByteStreamSinkFactory

IByteStreamSinkFactory – it creates media sink nodes which are linked with customised byte stream object with interface IMFByteStream by the next methods:

HRESULT createOutputNodes(VARIANT aArrayPtrCompressedMediaTypes, IUnknown* aPtrByteStream, VARIANT* aPtrArrayPtrTopologyOutputNodes) - create output nodes of container by byte stream, where **aArrayPtrCompressedMediaTypes** is an array of media types of the written formats, **aPtrByteStream** is pointer on customized object with **IMFByteStream** interface, **aPtrArrayPtrTopologyOutputNodes** is an array of topology nodes for writing of encoded media data.

IEncoderControl

IEncoderControl – it manages and creates video and audio encoders by the next methods:

HRESULT getCollectionOfEncoders(BSTR* aPtrPtrXMLstring) - get GUID collection of encoders in XML form ([EncoderFactoryXMLDocument](#)), where **aPtrPtrXMLstring** is a pointer on created string;

HRESULT getMediaTypeCollectionOfEncoder(IUnknown *aPtrUncompressedMediaType, REFCLSID aRefEncoderCLSID, BSTR* aPtrPtrXMLstring) - get collection of compressed media types for the selected encoder in XML form ([EncoderMediaTypeXMLDocument](#)), where **aPtrUncompressedMediaType** is a pointer on media type object, **aRefEncoderCLSID** is an encoder IID, **aPtrPtrXMLstring** is a pointer on created string;

HRESULT createEncoderNodeFactory(REFCLSID aRefEncoderCLSID, REFIID aREFIID, IUnknown aPtrPtrEncoderNodeFactory)** - create an encoder node factory, where **aRefEncoderCLSID** is an encoder IID, **aREFIID** is IID of encoder node factory interface ([IEncoderNodeFactory](#)), **aPtrPtrEncoderNodeFactory** is a pointer on created object.

EncoderFactoryXMLDocument

EncoderFactoryXMLDocument – it is XML document which defines CaptureManager encoder factories.

Node "**EncoderFactories**" is declaration of the root XML node.

Node "**Group**" is root of encoder factories list which are sorted by type.

Attribute "**Title**" defines friendly description of group, attribute "GUID" defines unique GUID of encoder factories group:

- **Video** - {73646976-0000-0010-8000-00AA00389B71}
- **Audio** - {73647561-0000-0010-8000-00AA00389B71}

Node "**EncoderFactory**" is node of encoder factory which has attribute "**Name**" - symbolic link of encoder factory, attribute "**Title**" - friendly description of format, attribute "**CLSID**" - defines unique GUID of encoder:

```
<?xml version="1.0" ?>
<!--XML Document of encoders-->
<EncoderFactories>
  <Group Title="Video" GUID="">
    <EncoderFactory Name="" Title="" CLSID=""/>
  </Group>
  <Group Title="Audio" GUID="">
    <EncoderFactory Name="" Title="" CLSID=""/>
  </Group>
</EncoderFactories>
```

EncoderMediaTypeXMLDocument

EncoderMediaTypeXMLDocument – it is XML document which defines the supported encoder media types.

Node "**EncoderMediaTypes**" is declaration of the root XML node.

Node "**Group**" is root of encoder media types list which are sorted by type.

Attribute "**Title**" defines friendly description of group, attribute "**GUID**" defines unique GUID of encoder factories group:

- **Variable bit rate** - {EE8C3745-F45B-42B3-A8CC-C7A696440955}
- **Constant bit rate** - {CA37E2BE-BEC0-4B17-946D-44FBC1B3DF55}

Node "**MediaTypes**" is node of encoder media type list which has attribute "**TypeCount**" - amount media types in list.

Node "**MediaType**" is root node for all items of one media type. It has attribute "**Index**" which defines index of the media type in the list of media types.

Node "**MediaTypeItem**" defines one media type's property. It has attribute "**Name**" - symbolic link, attribute "**GUID**" - unique GUID of media type's property, attribute "**Title**" - friendly name of property, attribute "**Description**" - friendly description of property.

```
<?xml version="1.0" ?>
<!--XML Document of Encoder MediaTypes-->
<EncoderMediaTypes>
  <Group Title="VBR" GUID="">
    <MediaTypes TypeCount="">
      <MediaType Index="">
        <MediaTypeItem Name="" GUID="" Title="" Description="">
        </MediaTypeItem>
      </MediaType>
    </MediaTypes>
  </Group>
```

```
<Group Title="CBR" GUID="">
  <MediaTypes TypeCount="">
    <MediaType Index="">
      <MediaTypeInfo Name="" GUID="" Title="" Description="">
      </MediaTypeInfo>
    </MediaType>
  </MediaTypes>
</Group>
</EncoderMediaTypes>
```

IEncoderNodeFactory

IEncoderNodeFactory – it creates node of the selected encoder by the next methods:

HRESULT STDMETHODCALLTYPE createCompressedMediaType(IUnknown *aPtrUncompressedMediaType, REFGUID aRefEncodingModeGUID, DWORD aEncodingModeValue, DWORD aIndexCompressedMediaType, IUnknown

****aPtrPtrCompressedMediaType)** - create encoder output media type, where **aPtrUncompressedMediaType** is a pointer on uncompressed media type, **aRefEncodingModeGUID** is IID of encoding mode (CBR, VBR), **aEncodingModeValue** is quality of encoding, **aIndexCompressedMediaType** is an index of compressed media type from the list of the supported compressed media types, **aPtrPtrCompressedMediaType** is a pointer on created object;

HRESULT STDMETHODCALLTYPE createEncoderNode(IUnknown *aPtrUncompressedMediaType, REFGUID aRefEncodingModeGUID, DWORD aEncodingModeValue, DWORD aIndexCompressedMediaType, IUnknown *aPtrDownStreamNode, IUnknown **aPtrPtrEncoderNode) - create encoder node, where **aPtrUncompressedMediaType** is a pointer on uncompressed media type, **aRefEncodingModeGUID** is IID of encoding mode (CBR, VBR), **aEncodingModeValue** is quality of encoding, **aIndexCompressedMediaType** is an index of compressed media type from the list of the supported compressed media types, **aPtrDownStreamNode** is a pointer on connected topology node, **aPtrPtrEncoderNode** is a pointer on created object.

ISessionControl

ISessionControl – it creates object for management of recording session by the next methods:

HRESULT createSession(VARIANT aArrayPtrSourceNodesOfTopology, REFIID aREFIID, IUnknown aPtrPtrSession)** - create session of topology, where **aArrayPtrSourceNodesOfTopology** is array of source topology nodes which stream media data, **aREFIID** is IID of session interface (**ISession**), **aPtrPtrSession** is pointer on created object.

ISession

ISession – it manages the recording session by the next methods:

**HRESULT startSession(LONGLONG
aStartPositionInHundredNanosecondUnits, REFGUID
aGUIDTimeFormat)** - start session which is referenced by session descriptor,
where **aStartPositionInHundredNanosecondUnits** is starting offset
(0), **aGUIDTimeFormat** is IID of time format (**GUID_NULL**);
HRESULT stopSession() - stop session which is referenced by session
descriptor;

HRESULT pauseSession() - pause session which is referenced by session
descriptor;

HRESULT closeSession() - close session which is referenced by session
descriptor;

HRESULT getSessionDescriptor(DWORD* aPtrSessionDescriptor) - get
session descriptor which is referenced with this session,
where **aPtrSessionDescriptor** is pointer on descriptor of the current session;

HRESULT getIConnectionPointContainer(REFIID aREFIID, IUnknown
aPtrPtrControl)** - get [IConnectionPointContainer](#) interface, where **aREFIID**
is IID of **IConnectionPointContainer** interface, **aPtrPtrControl** is pointer on
object.

ISessionCallback

ISessionCallback – it manages the result state of current recording session from CaptureManager inner thread by the next methods:

HRESULT invoke(DWORD aCallbackEventCode, DWORD aSessionDescriptor) - method for callback invoking, where **aCallbackEventCode** is a constant of [SessionCallbackEventCode](#), **aSessionDescriptor** is unique session descriptor.

Constants for reference of session's events

SessionCallbackEventCode – it is list of constants which refer to events in session.

UnknownEvent = 0 - event cannot be recognized;

Error = **UnknownEvent** + 1 - general error event;

Status_Error = **Error** + 1 - inner state session error event;

Execution_Error = **Status_Error** + 1 - execution session error event;

ItIsReadyToStart = **Execution_Error** + 1 - session ready to start event;

ItIsStarted = **ItIsReadyToStart** + 1 - session is started event;

ItIsPaused = **ItIsStarted** + 1 - session is paused event;

ItIsStopped = **ItIsPaused** + 1 - session is stopped event;

ItIsEnded = **ItIsStopped** + 1 - session is ended event;

ItIsClosed = **ItIsEnded** + 1 - session is closed and resources are released event;

VideoCaptureDeviceRemoved = **ItIsClosed** + 1 - lose connection to video source event.

IStreamControl

IStreamControl – it creates object for controlling of media streams by the next methods:

HRESULT getCollectionOfStreamControlNodeFactories(BSTR* aPtrPtrXMLstring) - get collection of stream control nodes in XML form ([StreamControlXMLDocument](#)), where **aPtrPtrXMLstring** is pointer on created string.

HRESULT createStreamControlNodeFactory(REFIID aREFIID, IUnknown aPtrPtrStreamControlNodeFactory)** - create a stream control node factory, where **aREFIID** is IID of stream control factory interface ([ISpreaderNodeFactory](#)), **aPtrPtrStreamControlNodeFactory** is pointer on created object.

StreamControlXMLDocument

StreamControlXMLDocument – it is XML document which defines CaptureManager stream control factories.

Node "**StreamControlNodeFactories**" is declaration of the root XML node. Node "**StreamControlNodeFactory**" is declaration of one node of stream control factory. Attribute "**Name**" defines symbolic name of stream control factory, attribute "**GUID**" defines unique GUID of stream control factory, attribute "**Title**" defines friendly description of stream control factor, attribute "**OutputPortCountConstant**" defines flexibility changing of output port amount, attribute "**InputPortCountConstant**" defines flexibility changing of input port amount. Node "**Value.ValueParts**" is root node of value list. Node "**ValuePart**" defines ports of stream control. It has attribute "**Title**" defines friendly description of value, attribute "**Value**" defines value, attribute "**IsPortCountConstant**" defines is changable amount of ports or not.

```
<?xml version="1.0" ?>
<!--XML Document of stream control node factories-->
<StreamControlNodeFactories>
  <StreamControlNodeFactory Name="" GUID="" Title=""
OutputPortCountConstant="" InputPortCountConstant="">
    <Value.ValueParts>
      <ValuePart Title="Count of input ports" Value=""
IsPortCountConstant=""/>
      <ValuePart Title="Count of output ports" Value=""
IsPortCountConstant=""/>
    </Value.ValueParts>
  </StreamControlNodeFactory>
</StreamControlNodeFactories>
```

ISpreaderNodeFactory

ISpreaderNodeFactory – it creates node for controlling of distribution data to media nodes by the next methods:

**HRESULT STDMETHODCALLTYPE createSpreaderNode(VARIANT
aArrayPtrDownStreamTopologyNodes, IUnknown**

****aPtrPtrTopologySpreaderNode)** - *create spread topology node,
where **aArrayPtrDownStreamTopologyNodes** is an array of output topology
nodes, **aPtrPtrTopologySpreaderNode** is pointer on created topology node.*

IMediaTypeParser

IMediaTypeParser – it creates text representation of media type in XML format by the next methods:

HRESULT parse(IUnknown* aPtrMediaType, BSTR* aPtrPtrXMLstring)
- parse MediaType into XML form ([MediaTypeXMLDocument](#)),
where **aPtrPtrXMLstring** is pointer on created string.

MediaTypeXMLDocument

MediaTypeXMLDocument – it is XML document which defines the current media type.

Node "**MediaType**" is declaration of the root XML node.

Node "**MediaTypeItem**" defines one property of media type. It includes attribute "**Name**" for symbolic name of property, attribute "**GUID**" for unique id of property, attribute "**Title**" for friendly name of property, attribute "**Description**" for more informative description of property.

```
<?xml version="1.0" ?>
<!--XML Document of media type-->
<MediaType>
  <MediaTypeItem Name="" GUID="" Title="" Description="">
  </MediaTypeItem>
</MediaType>
```

IStrideForBitmap

IStrideForBitmap – it computes stride of memory for the specific bitmap format by the next methods:

HRESULT **getStrideForBitmap**(**REFGUID** **aMFVideoFormat**, **DWORD** **aWidthInPixels**, **LONG*** **aPtrStride**) - get widthInBytes - stride of imag, where **aMFVideoFormat** is IID of uncompressed video format, **aWidthInPixels** is width of image in pixels, **aPtrStride** is pointer to write computed value.

IVersionControl

IVersionControl – it manages information about current version of *CaptureManager* by the next methods:

HRESULT getVersion(DWORD* aPtrMAJOR, DWORD* aPtrMINOR, DWORD* aPtrPATCH, BSTR* aPtrPtrAdditionalLabel) – get version of library, where **aPtrMAJOR** is pointer to write major number of version, **aPtrMINOR** is pointer to write minor number of version, **aPtrPATCH** is pointer to write patch number of version, **aPtrPtrAdditionalLabel** is pointer to write additional label of version;

HRESULT getXMLStringVersion(BSTR* aPtrPtrXMLstring) - get XML string format of library version ([CaptureManagerVersionXMLDocument](#)), where **aPtrPtrXMLstring** is pointer on created string;

HRESULT checkVersion(DWORD aMAJOR, DWORD aMINOR, DWORD aPATCH, BOOL* aPtrResult) - check of version, where **aMAJOR** is major number of version, **aMINOR** is minor number of version, **aPATCH** is patch number of version, **aPtrResult** is pointer to write the result of checking.

CaptureManagerVersionXMLDocument

CaptureManagerVersionXMLDocument – it is XML document which contains information about version of CaptureManager.

Node "**CaptureManagerVersion**" defines information about version. It includes attribute "**MAJOR**" for major number of version, attribute "**MINOR**" for minor number of version, attribute "**PATCH**" for patch number of version. Additional label information stored as inner text of node.

```
<?xml version="1.0" ?>
<!--XML Document of Capture Manager version-->
<CaptureManagerVersion MAJOR="" MINOR="" PATCH="">
  <!--ADDITIONAL_LABEL-->;
</CaptureManagerVersion>
```

IEVRStreamControl

IEVRStreamControl – it controls video rendering stream by the next methods:

HRESULT STDMETHODCALLTYPE setPosition(IUnknown *aPtrEVROutputNode, FLOAT aLeft, FLOAT aRight, FLOAT aTop, FLOAT aBottom) - set position of rendering in the target surface (from 0.0f to 1.0f), where **aPtrEVROutputNode** is pointer on rendering topology node, **aLeft** is position of left corner on target surface, **aRight** is position of right corner on target surface, **aTop** is position of top corner on target surface, **aBottom** is position of bottom corner on target surface;

HRESULT STDMETHODCALLTYPE setZOrder(IUnknown *aPtrEVROutputNode, DWORD aZOrder) - set the relating order of the current rendering to others, where **aPtrEVROutputNode** is pointer on rendering topology node, **aZOrder** is order position on target surface;

HRESULT STDMETHODCALLTYPE getPosition(IUnknown *aPtrEVROutputNode, FLOAT *aPtrLeft, FLOAT *aPtrRight, FLOAT *aPtrTop, FLOAT *aPtrBottom) - get position of rendering in the target surface (from 0.0f to 1.0f), where **aPtrEVROutputNode** is pointer on rendering topology node, **aPtrLeft** is pointer to write position of left corner on target surface, **aPtrRight** is pointer to write position of right corner on target surface, **aPtrTop** is pointer to write position of top corner on target surface, **aPtrBottom** is pointer to write position of bottom corner on target surface;

HRESULT STDMETHODCALLTYPE getZOrder(IUnknown *aPtrEVROutputNode, DWORD *aPtrZOrder) - get the relating order of the current rendering to others, where **aPtrEVROutputNode** is pointer on rendering topology node, **aPtrZOrder** is pointer to write order position on target surface;

HRESULT STDMETHODCALLTYPE flush(IUnknown *aPtrEVROutputNode) - execute rendering command immediatly, where **aPtrEVROutputNode** is pointer on rendering topology node;

HRESULT STDMETHODCALLTYPE setSrcPosition(IUnknown *aPtrEVROutputNode, FLOAT aLeft, FLOAT aRight, FLOAT aTop, FLOAT aBottom) - set position of rendering from the source surface (from 0.0f to 1.0f), where **aPtrEVROutputNode** is pointer on rendering topology node, **aLeft** is position of left corner on source surface, **aRight** is position of right corner on source surface, **aTop** is position of top corner on source surface, **aBottom** is position of bottom corner on source surface;

HRESULT STDMETHODCALLTYPE getSrcPosition(IUnknown *aPtrEVROutputNode, FLOAT *aPtrLeft, FLOAT *aPtrRight, FLOAT *aPtrTop, FLOAT *aPtrBottom) - get position of rendering from the source surface (from 0.0f to 1.0f), where **aPtrEVROutputNode** is pointer on rendering topology node, **aPtrLeft** is pointer to write position of left corner on source surface, **aPtrRight** is pointer to write position of right corner on source surface, **aPtrTop** is pointer to write position of top corner on source surface, **aPtrBottom** is pointer to write position of bottom corner on source surface;

HRESULT STDMETHODCALLTYPE getCollectionOfFilters(IUnknown *aPtrEVROutputNode, BSTR *aPtrPtrXMLstring) - get collection of applicable video filters in XML format ([EVROStreamFilterXMLDocument](#)), where **aPtrEVROutputNode** is pointer on rendering topology node, **aPtrPtrXMLstring** is pointer on created string;

HRESULT STDMETHODCALLTYPE setFilterParametr(IUnknown *aPtrEVROutputNode, DWORD aParametrIndex, LONG aNewValue, BOOL aIsEnabled) - set parametr of the selected applicable video filter, where **aPtrEVROutputNode** is pointer on rendering topology node, **aParametrIndex** is index of parametr, **aNewValue** is new value, **aIsEnabled** is flag for enabling or disabling parametr;

HRESULT STDMETHODCALLTYPE getCollectionOfOutputFeatures(IUnknown *aPtrEVROutputNode, BSTR *aPtrPtrXMLstring) - get collection of renderer features in XML format ([EVROStreamOutputFeatureXMLDocument](#)), where **aPtrEVROutputNode** is pointer on rendering topology node, **aPtrPtrXMLstring** is pointer on created string;

HRESULT STDMETHODCALLTYPE setOutputFeatureParametr(
IUnknown *aPtrEVROutputNode, DWORD aParametrIndex, LONG
aNewValue) - *set parametr of the selected renderer feature,*
*where **aPtrEVROutputNode** is pointer on rendering topology*
*node, **aParametrIndex** is index of parametr, **aNewValue** is new value.*

EVRStreamFilterXMLDocument

EVRStreamFilterXMLDocument – it is XML document which defines current filters of the video stream renderer output.

Node "**Filters**" is declaration of the root XML node.

Node "**Filter**" is declaration of one filter. Attribute "**Title**" defines friendly name of the filter, attribute "**CurrentValue**" defines current value of the filter, attribute "**Default**" defines default value of the filter, attribute "**Index**" defines index of the filter, attribute "**Min**" defines minimum acceptable value of the filter, attribute "**Max**" defines maximum acceptable value of the filter, attribute "**Step**" defines value of changing between minimum and maximum values of the filter, attribute "**Multiplier**" defines scale value of the filter, attribute "**IsEnabled**" defines enabling or disabling of the filter.

```
<!?xml version="1.0" ?>
<!-- XML Document of render stream filters -->
<Filters>
  <Filter Title="" CurrentValue="" Default="" Index="" Max="" Min=""
Step="" Multiplier="" IsEnabled="" />
</Filters>
```

EVRStreamOutputFeatureXMLDocument

EVRStreamOutputFeatureXMLDocument – it is XML document which defines current output features of the shared output renderer.

Node "**Features**" is declaration of the root XML node of feature list.

Node "**Feature**" is declaration of one feature. Attribute "**Title**" defines friendly name of the feature.

Node "**Color**" define color type of feature. It defines list of color's channels which are declared by node "**Channel**". Node "**Channel**" contains attribute "**CurrentValue**" defines current value of the channel, attribute "**Index**" defines index of the feature, attribute "**Min**" defines minimum acceptable value of the channel, attribute "**Max**" defines minimum acceptable value of the channel, attribute "**Step**" defines value of changing between minimum and maximum values of the channel, attribute "**Title**" defines friendly name of the channel.

```
<?xml version="1.0" ?>
<!-- XML Document of render stream output features -->
<Features>
  <Feature Title="Background Color">
    <Color>
      <Channel CurrentValue="0" Index="1" Max="255" Min="0"
Step="1" Title="Red"/>
      <Channel CurrentValue="0" Index="2" Max="255" Min="0"
Step="1" Title="Green"/>
      <Channel currentvalue="0" index="3" Max="255" Min="0"
Step="1" Title="Blue"/>
    </Color>
  </Feature>
</Features>
```

IRenderingControl

IRenderingControl – it controls video rendering process by the next methods:

HRESULT STDMETHODCALLTYPE enableInnerRendering(IUnknown *aPtrEVROutputNode, BOOL aIsInnerRendering) - enable or disable rendering by inner thread, where **aPtrEVROutputNode** is pointer on rendering topology node, **aIsInnerRendering** is state for enabling and disabling rendering by CaptureManager inner thread;

HRESULT STDMETHODCALLTYPE renderToTarget(IUnknown *aPtrEVROutputNode, IUnknown *aPtrRenderTarget, BOOL aCopyMode) - execute rendering to the rendering target, where **aPtrEVROutputNode** is pointer on rendering topology node, **aPtrRenderTarget** is pointer rendering target surface, **aCopyMode** is flag for enabling or disabling rendering with the proxy copy texture.

Screen Capture

Screen Capture – embedded into the CaptureManager live source which streams video from the screen of the desktop.

Screen Capture supports the next arguments for modifying of source:

1. **--options**
2. **--normalize**
3. **--HWND**

Argument "**--options**" allows define list of options which modify some source functionality. It is defined in format of XML document:

```
<?xml version='1.0' encoding='UTF-8'?>
<Options>
  <Option Type='Cursor' Visiblity='True'>
    <Option.Extensions>
      <Extension Type='BackImage' Height='100' Width='100'
Fill='0x7055ff55' Shape='Ellipse'/>
    </Option.Extensions>
  </Option>
  <Option Type='Clip' >
    <Option.Extensions>
      <Extension Left='0' Top='0' Height='100' Width='100' />
    </Option.Extensions>
  </Option>
</Options>
```

Node "**Option**" is a declaration of new option. Attribute "**Type**" defines type of option modification:

- **Cursor** - modify cursor.
- **Clip** - cut region of desktop area.

Attribute "**Visiblity**" defines type of option modification:

- **True** - make option visible.
- **False** - make option invisible.

Node "**Option.Extensions**" is a declaration of list of the extensions for the current option.

Node "**Extension**" is a declaration of the one extension for the current option.

Attribute "**Type**" defines type of extension:

- **BackImage** - image on the backbround

Attribute "**Left**" defines left offset of extension. Attribute "**Top**" defines top offset of extension. Attribute "**Height**" defines height of extension. Attribute "**Width**" defines width of extension. Attribute "**Fill**" colour for filling of extension in *ARGB Hex* format. Attribute "**Shape**" defines shape of extension:

- **Rectangle** - shape of filling is rectangle
- **Ellipse** - shape of filling is ellipse

example on C# code:

```
string lextendSymbolicLink = ISymbolicLink + " --options=" +
"<?xml version='1.0' encoding='UTF-8'?>" +
"<Options>" +
  "<Option Type='Cursor' Visiblity='True'>" +
    "<Option.Extensions>" +
      "<Extension Type='BackImage' Height='100' Width='100'
Fill='0x7055ff55' Shape='Ellipse'/>" +
    "</Option.Extensions>" +
  "</Option>" +
  "<Option Type='Clip'>" +
    "<Option.Extensions>" +
      "<Extension Left='0' Top='0' Height='100' Width='100'/>" +
    "</Option.Extensions>" +
```

```
"</Option>" +  
"</Options>";
```

Argument "**--normalize**" commands to apply post processing for transformation any screen orientation to the specific form:

- **Landscape** - form screen while top on upper part of image and bottom on lower part of image

example on C# code:

```
lextendSymbolicLink += " --normalize=Landscape";
```

Argument "**--HWND**" commands to set HWND handler of the window as a source for "Screen Capture" embedded live source. Value is set as a value of the selected window HWND handler. It supports only client window area screen capturing.

example on C# code:

```
lextendSymbolicLink += " --HWND=" +  
m_CurrentHWND.ToInt32().ToString();
```