WHITE PAPER

Axis DirectShow Guide

Created: January 20, 2010
Last updated: January 20, 2010
Rev: 1.0

TABLE OF CONTENTS

Introduction

Axis Communication AB provides a rich set of tools and software developer kits (SDK) for application developers. Most developers will use either AXIS Media Control (AMC) or AXIS Media Parser (AMP), which are full-fletched and fully supported client components to use within an application. These components intend to give developers a flying start playing and/or recording media streams from Axis video and audio products. The SDKs also provide other functionality such as PTZ-control. However, in some cases, the developer is unable to use the SDKs for some reason. In these cases it's possible to use the underlying DirectShow filters directly. This requires some or much knowledge of the DirectShow Framework. The purpose of this document is to provide as much technical information on the interfaces of these filters as possible, to enable developers to integrate them with their own components.

If you have a specific question or problem, please take a look at the FAQ before reading the rest of this guide.

1 Disclaimer

We highly recommend considering using the AXIS Media Control or the AXIS Media Parser SDKs. These software developer kits are tested by Axis Communications and full support is offered on the intended usage.

AXIS Media Control (AMC) is a competent *viewer* component with basic recording capabilities implemented as an ActiveX. AXIS Media Parser is primary a *parser* component that are intended for more advanced recording applications.

This documentation requires at least basic knowledge on how to program with the DirectShow API and some knowledge of working with COM interfaces. Axis Communications offers only limited support when using the DirectShow filters directly and is reluctant to advice on generic questions about the DirectShow API.

2 Summary

The DirectShow API provided by the Microsoft Corporation is a powerful media streaming architecture for Microsoft Windows. It is characterized by its modular approach and provides means for creating media pipelines (graphs) using built in or third-party filters.

Axis provides two types of filters that can be used with DirectShow, source filters and decoders. The source filters are always placed in the beginning of the media pipeline and delivers data to the down-stream filters, i.e. compressed video or audio from an Axis video product.

Axis currently provide three source filters:

- AXIS RTP Source Filter (AxisRTPSrcFilter.ax) – Deliver video, audio and/or meta-data streams from Axis video and audio products retrieved using the RTP/RTSP/RTCP protocols.
- AXIS HTTP Multipart Source Filter (AxisHTTPMPSrcFilter.ax) – Deliver Motion JPEG streams from Axis video and audio products retrieved using the HTTP protocol.
- AXIS Matroska File Splitter (AxMatroskaSplitter.ax) – Deliver video and audio streams from Matroska recordings made by the Local-Storage functionality on-board Axis video and audio products.

The compressed output from the source filters must be decoded in order to render them to the user. Most DirectShow video and audio decoders should be able to decompress the media streams from an Axis video and audio product, but Axis recommend using the designated decoders. Currently there are five different decoders provided:

- AXIS Motion JPEG Video Decoder (AxMJPEGDec.dll)
- AXIS MPEG-4 Video Decoder (AxMPEG4Dec.dll)
- AXIS H.264 Video Decoder (AxH264Dec.dll
- AXIS AAC Audio Decoder (AxAACDec.dll)
- AXIS G.7xx Audio Decoder (Audio.dll)

Note that the H.264, MPEG-4 and AAC decoders are licensed (each Axis device provides one decoder license of each supported codec).

AMC and AMP also utilize some other filters that are described briefly in Custom filters:

- AXIS Audio Processor DMO (Audio.dll)
- AXIS Picture Events Filter (AxPicEventsFilter.ax)
- AXIS Overlay Filter (AxisOverlayFilter.ax)

3 Glossary

- **ASF (Advanced Systems Format**) – Video and audio container format from Microsoft suitable for streaming media.
- **GraphEdit –** Tool for building DirectShow graphs (see Using the GraphEdit tool)
- **Local Storage –** Recording to local storage on an Axis video and audio product, e.g. SD-card.
- **Matroska –** Advanced media container format (used for *Local Storage*).
- **Multicast –** Efficient network distribution - sending data to multiple receivers at once. Multicast traffic can normally not pass switches, routers and firewalls. Is suitable for distribution on local network.
- **HTTP Multipart -** Encapsulations of several body parts in the HTTP message.
- **RTP (Real-Time Transport Protocol) –** Network protocol for sending real-time data, e.g. video or audio.
- **RTCP (Real-Time Control Protocol) –** Two-way communication protocol used with RTP for sending control information and statistics between sender and receiver.
- **RTSP (Real-Time Streaming Protocol) –** Session based protocol for configuring and controlling of media servers. Often used together with the RTP protocol.
- **SDP (Session Description Protocol) –** Protocol for describing media content. Used by the RTSP protocol (DESCRIBE response) to specify the properties of available media streams.
- **SPS/PPS (Sequence/Picture Parameter Set) –** Configuration header used in the H.264 format.
- **Unicast –** Network distribution scheme - sending data to a single designated receiver.
- **VAPIX –** Open API for software integration with Axis video and audio products.

4 Source Filters

4.1 AXIS RTP Source Filter

The AXIS RTP Source Filter is capable of delivering H.264, MPEG-4 or Motion JPEG video and AAC or G.711 audio and meta-data from one or several Axis video and audio products using the RTP/RTSP protocol. The media samples will be time-stamped in order to synchronize the different media streams. The source filter will by default utilize RTCP communication with the device in order to compensate for client-server-clock drift.

4.1.1 Loading the RTP source Filter

The AXIS RTP Source Filter implements *IFileSourceFilter* interface, which can be used to programmatically load the filter into the DirectShow graph. Use the *Load* method with the parameter *pszFileName* pointing to the RTSP control URL of an Axis video and audio product, e.g.:

```
VAPIX version 3 (firmware 5.xx):
"axrtpm://<server-address>/axis-media/media.amp?videocodec=h264"

VAPIX version 2 (firmware 4.xx):
"axrtpm://<server-address>/mpeg4/media.amp"
```

This source filter register custom protocol schemes in order for DirectShow to load the source filter automatically, e.g. when using Windows Media Player or GraphEdit.

The protocol schemes used by this source filter and their meaning:

Scheme	Description
axrtpm	Multicast RTP stream(s) (UDP)
axrtpu	Unicast RTP stream(s) (UDP)
axrtsp	Unicast RTSP stream (TCP)
axrtsphttp	Unicast RTSP stream over HTTP

It is possible to control the properties of the requested media stream(s) by adding parameters to the stream path. For example:

```
"axrtpm://<server-address>/axis-
media/media.amp?videocodec=h264&resolution=320x240&fps=1"
```

Note that the RTSP path and supported parameters may differ between VAPIX versions. Please consult the VAPIX documentation (RTSP API) for a comprehensive understanding on how to construct the URL.

From version 2.5.2 of the RTP Source Filter it is possible to load the source filter using multiple RTSP sessions, i.e. the filter is able to synchronize media streams from different Axis video and audio products. For example, the filter can deliver a video stream from one device and audio streams from two other Axis video and audio products. Call the *Load* function with the different RTSP URLs concatenated and separated with commas.

4.1.2 Configuring the RTP Source Filter

The AXIS RTP Source Filter can be configured using the *IAxPropertySet* interface (see Using IAxPropertySet). The most common usage is to set session credentials and network timeout value. Below is a complete summary of all available properties.

Name	Type	R/W[1]	SB[2]	Default	Description
url	BSTR	R	Yes	-	Session URL
net_timeout	LONG	R/W		15000	Network timeout (ms)
stream_uname	BSTR	R/W	Yes	-	Session username
stream_pwd	BSTR	R/W	Yes	-	Session password
net_multicast_if	BSTR	W		-	Multicast interface to use
Popups	LONG	R/W		2	Notification level (0,1,2)[3]
add_decoders	BSTR	W		'yes'	Whether the filter should try to add required decoders ('yes', 'no')
rtp_delay	LONG	R/W		0	RTP buffer delay (ms)
rtp_delay_audio	LONG	R/W		150	RTP buffer delay (ms) when delivering audio
aspect_ratio_info	LONG	R		-	Pixel aspect ratio of first video stream[4]
synchronize_ using_rtcp	BOOL	R/W		TRUE	Whether to compensate for clock drift using RTCP.
bit_rate_video	LONG	R		-	RTP packet bit rate (bits/s) for the first video stream
bit_rate_audio	LONG	R		-	RTP packet bit rate (bits/s) for the first audio stream
error_code	LONG	R	Yes	-	Latest error code
error_msg	BSTR	R	Yes	-	Latest error message
max_num_video	LONG	R/W		INF	Max allowed video streams
max_num_audio	LONG	R/W		INF	Max allowed audio streams
time_format	BSTR	R/W		'default'	Output time format ('default', 'ntp'[5])

Note that properties should be set prior to calling *Load*.

4.1.3 Retrieving notifications and error messages

To receive error and notification events from the RTP source filter, the host application should implement the *IAxEventListener* interface and register the host application instance using the *IAxEventNotifier* filter interface (see IAxEventNotifier.h).

Note that after registering your object, the events are passed directly to the listener and the popup messages become disabled.

See Error and notification codes for possible error codes. Session specific errors, when running multiple RTSP sessions, can be queried using the *IAxPropertySet* interface (*'error_code'* and *'error_msg'* property).

[1] R/W = read/write

[2] SB = Session based, i.e. when using multiple RTSP sessions it is possible to specify a session dependant property by appending the session index immediately after the property name. Ex, 'url1' denotes the URL of the second RTSP session.

[3] Notification levels: 0 = No message boxes, 1 = Only Login Dialogs are shown when required, 2 = All messages will be shown including Login Dialogs and error messages.

[4] MAKELPARAM(X,Y)

[5] It is possible to force the source filter to output media sample timestamps in server NTP time (ticks in 100-nano second units since 1900-01-01 00:00).

4.1.4 Using the recording interface

Note that Axis encourages using AMC or AMP/AMV for recording purposes.

The *IAxMediaRecorder* interface exposed by the RTP source filter (see IAxMediaRecorder.h**)** could be used to store the media streams delivered from the filter to an ASF-container**.**

This functionality requires that the *AXIS File Writer* (AxisFileWriter.dll) is installed on the host system in addition to the WMF (Windows Media Format) distribution. WMF is distributed with Windows Media Player. Check that all required components are installed by calling the *IsRecordable* method of *IAxMediaRecorder*.

Code example of usage:

```
myRecorder->BeginRecording(L"C:\\test.asf", RECM_VIDEO_AUDIO);
/*...*/
myMediaRecorder->EndRecording();
```

The media recorder is limited to record one video and/or one audio stream at a time.

4.2 AXIS HTTP Multipart Source Filter

This source filter is capable of delivering Motion JPEG video from an Axis video and audio product using HTTP Multipart. The samples will be correctly time-stamped - the filter will use the timestamps in the JPEG header (for details see Axis JPEG Format).

Note that audio cannot be used with this filter. The filter has no recording capabilities.

4.2.1 Loading the HTTP Multipart Source Filter

The AXIS HTTP Multipart Source Filter implements *IFileSourceFilter* interface, which can be used to programmatically load the filter into the DirectShow graph. Use the *Load* method with the parameter *pszFileName* pointing to the Motion JPEG stream URL of an Axis video and audio product, e.g.:

```
"axmphttp://<server-address>/axis-cgi/mjpg/video.cgi"
```

This source filter register custom protocol schemes in order for DirectShow to load the source filter automatically, e.g. when using Windows Media Player or GraphEdit.

The protocol schemes used by this source filter and their meaning:

Scheme	Description
axmphttp	HTTP Multipart
axmphttps	HTTP Secure Multipart

It is possible to control the properties of the requested Motion JPEG stream by adding parameters to the stream path. For example:

```
"axmphttp://<server-
address>/mjpg/video.mjpg?resolution=320x240&fps=1"
```

Please consult the VAPIX documentation (HTTP API) for a comprehensive understanding on how to construct the URL.

4.2.2 Configuring the HTTP Multipart Source Filter

The AXIS HTTP Multipart Source Filter can be configured using the *IAxPropertySet* interface (see Using IAxPropertySet). The most common usage is to set session credentials and network timeout value. Below is a complete summary of all available properties

Name	Type	R/W[6]	Default	Description
url	BSTR	R/W	-	Session URL
net_timeout	LONG	R/W	15000	Network timeout (ms)
stream_uname	BSTR	R/W	-	Session username
stream_pwd	BSTR	R/W	-	Session password
Popups	LONG	R/W	2	Notification level (0,1,2)[7]
add_decoders	BSTR	W	'yes'	If filter should try to add required decoders ('yes', 'no')

Note that properties should be set prior to calling *Load*.

4.2.3 Retrieving notifications and error messages

To receive error and notification events from the HTTP Multipart source filter the host application should implement the *IAxEventListener* interface and register the host application instance using the *IAxEventNotifier* filter interface (see IAxEventListener.h and IAxEventNotifier.h).

Note that after registering your object, the events are passed directly to the listener and the popup messages become disabled.

See Error and notification codes for possible error codes.

4.3 AXIS Matroska File Splitter

The AXIS Matroska File Splitter is a source filter capable of delivering media streams from recordings stored in the Matroska container format (.mkv) originating from an Axis video and audio product. Typically, a recording has been created using the Local Storage functionality of an Axis video and audio product. The recording can then be played on the client computer using this source filter.

In most cases it should be possible to use any Matroska Splitter available. However AXIS Matroska Splitter is required for playback of H.264 recordings originating from older 5.xx firmware versions.

This filter is ideal and optimized for Matroska files generated by Axis video and audio products. However for more general usage we recommend using another Matroska splitter (e.g. Haali Media Splitter).

4.3.1 Loading the Matroska File Splitter

The AXIS Matroska File Splitter implements *IFileSourceFilter* interface, which can be used to programmatically load the filter into the DirectShow graph. Use the *Load* method with the parameter *pszFileName* pointing to the location of the mkv-file. The configuration file ('.conf') must be located side-by-side with the mkv-file in order to play H.264-recordings of older firmware versions.

4.3.2 Configuring the Matroska File Splitter

This source filter can be configured using the *IAxPropertySet* interface (see Using IAxPropertySet). Currently, there is only one parameter available for configuration: *'cue_repair_option'*. This

[6] R/W = read/write
[7] Notification levels: 0 = No message boxes, 1 = Only Login Dialogs are shown when required, 2 = All messages will be shown including Login Dialogs and error messages.

parameter configures how the filter should act when a Matroska file is missing cueing information. Cueing information is needed for the media stream to be seekable. Below is a list of the available options.

Value (LONG)	Meaning
0	Never reconstruct cues
1	Reconstruct cues when needed
2 (default)	Prompt the user whether or not to reconstruct cues

Note that reconstructing the cues might take several minutes for large files.

4.4 Using IAxPropertySet

This interface enables reading and writing property values (Strings, Integers, Booleans) of the supported class using specific property names. All source filters described in this guide implements this interface and specifies a set of supported properties.

Below follows an example on how to disable cue reconstruction of the Matroska File Splitter (see Configuring the Matroska File Splitter).

```
#include "IAxPropertySet.h"

const IID IID_IAxPropertySet = { 0xde69f62b, 0xfcec, 0x4902, { 0xbc,
0xdd, 0x1e, 0x6e, 0x44, 0x33, 0x12, 0x75 } };

#define PROP_NAME_CUE_REPAIR  "cue_repair_option"

#define CUE_REPAIR_OPTION_DISABLE  0
#define CUE_REPAIR_OPTION_ENABLE   1
#define CUE_REPAIR_OPTION_PROMPT   2

/*...*/

IBaseFilter* aMatroskaSource;
IAxPropertySet* aPropertySet;
hr = aMatroskaSource.QueryInterface(IID_IAxPropertySet,
&aPropertySet);
if(SUCCEEDED(hr))
{
  hr = aPropertySet->SetProperty(_bstr_t(PROP_NAME_CUE_REPAIR),
    variant_t(CUE_REPAIR_OPTION_DISABLE));
  if(FAILED(hr)) {/*...*/};

  aPropertySet->Release();
}
```

5 DMO Decoders

In order to render the compressed media streams delivered by the different source filters, a decoder must be added to the media pipeline before the stream can be sent to an audio or video renderer. Axis provides three video decoders and two audio decoders for this purpose. The source filters will normally insert the appropriate decoder(s) to the DirectShow graph automatically (if installed). This behavior can be disabled if necessary by setting the *'add_decoders'* property to *'no'* (see Configuring the RTP Source Filter or Configuring the HTTP Multipart Source Filter).

All decoders are of the type DirectX Media Objects, which means that it is possible to use them outside the DirectShow environment (see Processing Data with DMOs).

It is of course possible to add the decoders manually to the DirectShow graph. This is done using the *DMOWrapperFilter* provided by DirectShow (see Using the DMO Wrapper Filter).

5.1 Licensing issues

Three of the decoders (H.264, MPEG-4 and AAC) require usage license. Each Axis device provides one decoder license of each supported codec.

A license must be requested from MPEG LA to distribute these decoders with your application.

5.2 Quality Control

All of the video decoders implement the IDMOQualityControl interface, which means that they can respond to late samples. For example if the decoder doesn't manage to process all received samples in time it can respond by lowering the decoding quality or throwing samples. The exact behavior differs between the decoders and can be configured by the application. This is described in detailed below. The IDMOQualityControl interface also offers the application to disable or enable the quality control functionality of the decoder completely.

If the time-stamp information of the media samples is not used, the application should always force the time stamps to zero. In that case Quality Control won't interfere even if enabled.

5.3 Receiving video frame callback notifications

It is possible to receive notifications from the video decoders each time a frame has been decoded. The notification call is attaching a buffer with the decoded frame in a specified color space.

Notifications are received by implementing the *IAxMediaCB* interface in the application class and register the application object using the decoder's *SetCallback()* method (*IAxH264Dec, IAxMPEG4Dec, IAxMJPEGDec*), specifying also the color space (see the corresponding interface definition file).

5.4 AXIS H.264 Video Decoder

This decoder decodes a H.264 video stream, also known as AVC or MPEG-4 part 10.

5.4.1 Supported input types

The media subtype required by the decoder is *MEDIASUBTYPE_H264*. Three possible format types can be used for the input:

- VIDEOINFOHEADER
- VIDEOINFOHEADER2 – enable use of picture aspect ratio information.

- MPEG2VIDEOINFO – enable use of Sequence and Picture Parameter Set[8].

If the Sequence and Picture Parameter Set are not included in the media type using *MPEG2VIDEOINFO*, they must be sent to the decoder as the first sample. **Note** that the SPS and PPS are not included in the media stream, but must be parsed from the SDP data.

```
From SDP:
sprop-parameter-sets=[base64-coded-SPS],[base64-coded-PPS]

First sample to the decoder:
[0x00 0x00 0x01][SPS][0x00 0x00 0x01][PPS]
```

5.4.2 Supported output types

The decoder supports the following output subtypes:

- *MEDIASUBTYPE_YUY2* (16bits per pixel)
- *MEDIASUBTYPE_YV12* (12 bits per pixel)
- *MEDIASUBTYPE_RGB24* (24 bits per pixel)
- *MEDIASUBTYPE_RGB32* (32 bits per pixel)

Two possible format types can be used for the output:

- VIDEOINFOHEADER
- VIDEOINFOHEADER2 – enable use of picture aspect ratio information.

5.4.3 Configuring the H.264 decoder

The decoder implements two custom interfaces for configuration purposes, *IAxH264Dec* and *IAxH264Dec2* (see AxH264Dec.h). Many of the methods contained in the interfaces are not implemented. Note that *IAxH264Dec2* is available from version 2.2.0 of the decoder.

By using the *SetDecodeFlags* method of *IAxH264Dec* it is possible to force the decoder to only decode IDR frames (an *IDR frame* is a special kind of I-frame used in H.264 encoding). Set the method argument to 0 to only decode IDR frames and to 2 to decode all frames.

The decoding quality of the processed data can be configured using the *SetQualitySettings* method of *IAxH264Dec2*. The method argument specifies a bitwise combination of flags described below:

Value	Meaning
0x000-0x009 (mask: 0x0F)	Decoding quality (9 = highest quality, 0 = lowest CPU usage)
0x010	Dynamic decoding quality (compensate data flooding by lowering decoding quality dynamically)
0x100	Allow frame dropping if decoder falls behind.

By default, the decoder uses the best decoding quality and allows frame dropping. Frame dropping will start if the samples are more than 500 ms behind schedule.

If RGB is used as output subtype it is possible to mirror the output image vertically by using the *RGBMirror* function. This is especially useful if you need to change the bottom-up orientation to top-down when getting frame callback notifications or when using the *GetCurrentPicture* method.

The GetCurrentPicture function returns the latest decoded bitmap (BITMAPINFOHEADER + scan data). The function also returns the size of the image data. If the buffer pointer is *NULL* the function

[8] dwFlags should be set to the NAL unit size length + 1. SPSs and PPSs are concatenated and stored at dwSequenceHeader, prefixed with two bytes big-endian size fields (without start codes).

will return the required buffer size. To save the image to a bmp-file a BITMAPFILEHEADER must be prefixed to the returned buffer.

5.5 AXIS MPEG-4 Video Decoder

This decoder decodes a MPEG-4 video stream.

5.5.1 Supported input types

The media subtype required by the decoder is *DX50* (see Custom media subtypes). Three possible format types can be used for the input:

- VIDEOINFOHEADER
- VIDEOINFOHEADER2 – enable use of picture aspect ratio information.
- MPEG2VIDEOINFO – enable use of the MPEG-4 configuration header[9].

If the configuration header is not included in the media type using *MPEG2VIDEOINFO*, it must be sent to the decoder as a sample. The configuration header can be found in the SDP data but is also present in the media stream.

5.5.2 Supported output types

The decoder supports the following output subtypes:

- *MEDIASUBTYPE_YUY2* (16bits per pixel)
- *MEDIASUBTYPE_YV12* (12 bits per pixel)
- *MEDIASUBTYPE_RGB24* (24 bits per pixel)
- *MEDIASUBTYPE_RGB32* (32 bits per pixel)

Two possible format types can be used for the output:

- VIDEOINFOHEADER
- VIDEOINFOHEADER2 – enable use of picture aspect ratio information.

5.5.3 Configuring the MPEG-4 decoder

The decoder implements a custom interface for configuration purposes, *IAxMPEG4Dec* (see AxMPEG4Dec.h). Many of the methods contained in the interface are not implemented.

By using the *SetDecodeFlags* method of *IAxMPEG4Dec* it is possible to force the decoder to only decode I-frames. Set the method argument to 0 to only decode I-frames and to 2 to decode all frames.

The decoding quality of the processed data can be configured using the *SetQualitySettings* method of *IAxMPEG4Dec*. The method argument specifies a bitwise combination of flags described below:

Value	Meaning
0x100	Allow frame dropping if decoder falls behind.

By default, the decoder allows frame dropping. Frame dropping will start if the samples are more than 500 ms behind schedule.

De-blocking filtering was added in version 2.2.0 of the decoder and is by default disabled. De-blocking enhance image quality by smoothing the sharp edges between different sub-image blocks. Configure using the *SetDeblock* method of *IAxMPEG4Dec* (0 = no de-blocking, 1 = de-blocking). De-blocking will decrease performance.

[9] The header is stored in dwSequenceHeader.

If RGB is used as output subtype it is possible to mirror the output image vertically by using the *RGBMirror* function. This is especially useful if you want to change the bottom-up orientation to top-down when getting frame callback notifications or when using the *GetCurrentPicture* method.

The *GetCurrentPicture* function returns the latest decoded bitmap (BITMAPINFOHEADER + scan data). The function also returns the size of the image data. If the buffer pointer is *NULL* the function will return the required buffer size. To save the image to a bmp-file the BITMAPFILEHEADER must be prefixed to the returned buffer.

5.6 AXIS Motion JPEG Video Decoder

This decoder decodes a Motion JPEG video stream.

5.6.1 Supported input types

The media subtype required by the decoder is *MEDIASUBTYPE_MJPG*. Two possible format types can be used for the input:

- VIDEOINFOHEADER
- VIDEOINFOHEADER2 – enable use of picture aspect ratio information.

5.6.2 Supported output types

The decoder supports the following output subtypes:

- *MEDIASUBTYPE_YUY2* (16bits per pixel)
- *MEDIASUBTYPE_YV12* (12 bits per pixel)
- *MEDIASUBTYPE_RGB24* (24 bits per pixel)
- *MEDIASUBTYPE_RGB32* (32 bits per pixel)

Two possible format types can be used for the output:

- VIDEOINFOHEADER
- VIDEOINFOHEADER2 – enable use of picture aspect ratio information.

5.6.3 Configuring the Motion JPEG decoder

The decoder implements a custom interface for configuration purposes, *IAxMJPEGDec* (see AxMJPEGDec.h). Many of the methods contained in the interface are not implemented.

If RGB is used as output subtype it is possible to mirror the output image vertically by using the *RGBMirror* function. This is especially useful if you want to change the bottom-up orientation to top-down when getting frame callback notifications or when using the *GetCurrentPicture* method.

The *GetCurrentPicture* function returns the latest decoded bitmap (BITMAPINFOHEADER + scan data). The function also returns the size of the image data. If the buffer pointer is *NULL* the function will return the required buffer size. To save the image to a bmp-file the BITMAPFILEHEADER must be prefixed to the returned buffer.

If quality control is enabled frame dropping will always engage if the samples are more than 500 ms behind schedule.

5.7 AXIS AAC Audio Decoder

This decoder decodes an AAC-LC (Low Complexity) mono audio stream.

5.7.1 Supported input types

The media subtype required by the decoder is (raw) *AAC* (see Custom media subtypes). WAVEFORMATEX should be used as format type. Note that only one-channel audio (mono) is supported.

5.7.2 Supported output types

The decoder supports MEDIASUBTYPE_PCM (stereo 16 bits) as output subtype and WAVEFORMATEX as format type. The decoder does not support re-sampling, i.e. *nSamplePerSec* must be the same as the input type.

5.8 AXIS G.7xx Audio Decoder

This decoder decodes G.711, G.726-32 kbps and G.726-24 kbps audio streams (8 kHz mono). Note that the G.7xx decoder is included in the Audio Component (audio.dll), which is distributed with the AMC and AMP SDKs.

5.8.1 Supported input types

The media subtype required by the decoder is *G711*, *G726-32* and *G726-24* (see Custom media subtypes). WAVEFORMATEX should be used as format type. Note that only one-channel audio (mono) is supported and only 8 kHz sampling rate. The block align is 1 for G.711 and G.726-32 kbps and 3 for G.726-24 kbps.

5.8.2 Supported output types

The decoder supports MEDIASUBTYPE_PCM (stereo 16 bits 8 kHz) as output subtype and WAVEFORMATEX as format type.

6 Custom filters

6.1 AXIS Audio Processor DMO

This DMO is capable of up-sampling PCM audio by an integer factor.

6.2 AXIS Picture Events Filter

This filter sends notifications for every processed video sample.

6.3 AXIS Overlay Filter

AXIS Overlay Filter is currently used to mix video with time-stamped geometrical data from meta-data streams.

7 FAQ

7.1 Should I use the developer or the embedded version of the filters?

Always use the *developer* versions of the components, which are distributed with the AMC and AMP SDKs!

Most of the components used by AMC and AMP comes in two flavors: *embedded* version (filename appended with 'Emb') and *developer* version. The components are (almost) identical and use the same version series but have different CLSIDs. The *embedded* versions are used by the *embedded* AMC, shipped with Axis video and audio products, and used embedded on the Live View page. By using the *developer* versions you can avoid the risk of breaking your application when the user upgrades the *embedded* versions of the components when visiting a Live View page that are requiring newer versions of the components than the version your application is tested with.

A common disbelief is that the *embedded* AMC version should be used in web sites in general. But this is wrong. Axis Developer Partners should always use the *developer* version.

7.2 Is Axis Filters compatible with Windows Media Foundation?

All of the video and audio decoders implement the IMFTransform interface and can be used with WMF. However, the source filters is not compatible with WMF.

Note that DirectShow is supported on both Vista and Windows 7.

7.3 The H.264 decoder fails to decompress the H.264 stream. What am I doing wrong?

Check for errors from the decoder (see Debug output).

A common fault is to not include the SPS/PPS in the media type or in the bit stream (see Supported input types).

7.4 There are jitter and stutter in the video/audio (RTP) stream. What can I do?

This can be caused by different factors:

- Network jitter – try increasing the RTP buffer size of the RTP Source Filter (see Configuring the RTP Source Filter).
- The client system is overloaded - You might consider lowering the decoding quality (H.264) or configuring the decoder to only decode key-frames (MPEG-4/H.264). You could also disable frame dropping if you are suspecting the stutter to be caused by dropped frames (see Configuring the H.264 decoder or corresponding section).
- The client machine may be giving inaccurate times to the RTP Source Filter, causing the RTCP synchronization to malfunction. This typically happens on slow systems, e.g. laptops, and results in reoccurring jumps in the media streams. This can be resolved by disabling RTCP synchronization (see Configuring the RTP Source Filter).

7.5 The host machine is performing badly when running multiple high-resolution H.264 streams. What can I do?

Lower the decoding quality (Configuring the H.264 decoder) and/or allow frame dropping.

7.6 How can I get rid of the login dialog?

Set the *'Popups'* property to 0. See Configuring the RTP Source Filter or Configuring the HTTP Multipart Source Filter. In this case you need to programmatically set the credentials to the source filter and subscribing to error events (see Retrieving notifications and error messages (RTP) or Retrieving notifications and error messages (HTTP)).

7.7 I don't want to use DirectShow. What are my options?

You can use the decoders outside DirectShow (see DMO Decoders) but you need to handle the media streams/files yourself.

7.8 How do I know the CLSIDs, GUIDs and IIDs?

All Axis defined GUIDs used in this guide is summarized in section Interface, filter and media type identifiers.

8 Debugging

This section describes some useful techniques and tools for debugging with DirectShow and the Axis filters.

8.1 Using the GraphEdit tool

GraphEdit is a really useful tool for visually building DirectShow graphs. GraphEdit is distributed with the Windows SDK.

For building a graph in GraphEdit, use File->Render URL with the URL address e.g.

"*axrtsphttp://<server-address>/axis-media/media.amp?videocodec=h264*"

See Loading the RTP source Filter and Loading the HTTP Multipart Source Filter for details on building the URL.

8.2 Debug output

The filters can output errors and useful information to a file or the debug output. Using the AXIS Diagnostics Tool (AxDiag) is the easiest way to enable/disable logging for different components and packages.

8.3 Using the Windows Media ASF Viewer 9 Series

If you are having problems with the ASF-recordings, we recommend using the ASF Viewer tool from Microsoft. It allows you to inspect the content stored in the ASF-container, e.g. time stamping data.

9 Error and notification codes

Error and notifications codes are HRESULTs with facility set to FACILITY_ITF (4) error id in the range 0x0200-0x09FF (see Retrieving notifications and error messages (RTP) and Retrieving notifications and error messages (HTTP))

Error code	R/H[10]	Description
0x80040200	R/H	A network error occurred
0x80040201	R/H	The stream ended while expecting more data
0x80040202	R	The UDP media stream (multicast) could not be received (firewall/router problem)
0x80040203	R/H	Network resource not found
0x80040204	R	One or several RTSP sources failed to connect, but not all. Use *IAxPropertySet* interface to get session specific errors.
0x80040205	R	All RTSP sources failed to connect
0x80040232	R	Failed to create instance of AXIS AAC Audio Decoder
0x80040240	H	Failed to read content type
0x80040300	R/H	Unexpected initialization error
0x80040301	R/H	Access denied for network resource
0x80040302	R/H	Critical component missing
0x80040303	R	Component too old
0x80040909	R	Wrong username and/or password

Notification code	R/H	Description
0x80040600	R	Pixel aspect ratio has been updated.
0x80040601	R	*Deprecated*. The filter requires a restart.

[10] Applicable for R = AXIS RTP Source Filter and/or H = AXIS HTTP Multipart Source Filter

10 Interface, filter and media type identifiers

This section lists all Axis specific identifiers described in this guide. Note that many of these definitions are present in corresponding h-files, e.g. AxH264Dec.h.

10.1 Filter CLSIDs

The CLSIDs below corresponds to the ADP versions of the filters, i.e. the filters that are distributed with the AMC and AMP SDKs, which are recommended to use for ADP development.

Filter	CLSID
RTP Source Filter	{4f1d0c59-5ecc-4028-87f3-482191d2230f}
HTTP Multipart Source Filter	{1a643a97-f45e-40bd-82bb-3da946a249eb}
Matroska File Splitter	{c696dddc-d62b-48a0-a449-369700f96903}
H.264 Video Decoder	{7340f0e4-aeda-47c6-8971-9db314030bd7}
MPEG-4 Video Decoder	{c32fe9f1-a857-48b0-b7bf-065b5792f28d}
Motion JPEG Video Decoder	{29020fe0-3833-4be2-a7f0-42841d4b45f0}
AAC Audio Decoder	{ba7a56eb-d1b9-443b-96e9-086532a378f1}
G.7xx Audio Decoder	{d1f283e1-df6f-41e4-aed1-eb1d6b4b0ae1}
Picture Events Filter	{833caf80-a132-4fef-8188-2a6004e3f239}
Audio Processor Filter	{b2505399-a5ce-4824-876d-77df16986cda}
Overlay Filter	{f83004ad-524b-4887-b049-2d3407161d24}

10.2 Interface IID

Interface	IID
IAxPropertySet	{de69f62b-fcec-4902-bcdd-1e6e44331275}
IAxEventListener	{26e556c1-e0e2-4cd2-b897-230a292f8cb1}
IAxEventNotifier	{c365f894-b856-4c80-9d90-3cec6e742f33}
IAxMediaRecorder	{e2b8c062-0079-451a-aac5-317ff1979f16}
IAxH264Dec	{e8f81ea3-4c67-4f85-ba08-490e67509d10}
IAxH264Dec2	{ad61a4b9-b878-47c8-a2fa-ff18786dc520}
IAxMP4Dec	{6ff6697e-507b-4753-9afb-7542ea0cd5bb}
IAxMJPGDec	{7404c99f-5b2b-42c2-951b-73d09fb66fba}

10.3 Custom media subtypes

This list contains definitions of the media sub types used by the filters in this guide and which are not defined in DirectShow. The first group (four bytes) defines the *format tag* of the media type.

Subtype	GUID
DX50	{30355844-0000-0010-8000-00AA00389B71}
AAC	{000000FF-0000-0010-8000-00AA00289B71}
G711	{00000007-0000-0010-8000-00AA00289B71}
G726-32	{00000014-0000-0010-8000-00AA00289B71}
G726-24	{00000040-0000-0010-8000-00AA00289B71}

11 Interface definition files (external)

List of interface definition files bundled with this guide.

- IAxPropertySet.h
- IAxEventNotifier.h
- IAxMediaRecorder.h
- AxH264Dec.h
- AxMP4Dec.h
- AxMJPGDec.h

12 References

- Microsoft DirectShow API
  http://msdn.microsoft.com/en-us/library/dd375454.aspx
- Alphabetical List of DirectShow Interfaces
  http://msdn.microsoft.com/en-us/library/dd373420.aspx
- VAPIX Application Programming Interface
  https://www.axis.com/techsup/cam_servers/dev/cam_http_api_index.php
- DirectX Media Objects
  http://msdn.microsoft.com/en-us/library/aa916423.aspx
- DirectShow Structures
  http://msdn.microsoft.com/en-us/library/dd375469.aspx
- DirectShow Media Types
  http://msdn.microsoft.com/en-us/library/ms932033.aspx
- DirectShow H.264 Video Types
  http://msdn.microsoft.com/en-us/library/dd757808(VS.85).aspx
- Media Foundation Interfaces
  http://msdn.microsoft.com/en-us/library/ms696268(VS.85).aspx
- Simulating Graph Building with GraphEdit
  http://msdn.microsoft.com/en-us/library/dd377601.aspx
- AXIS Diagnostics Tool
  http://www.axis.com/ftp/pub_soft/cam_srv/amc/tools/
- Windows Media ASF Viewer 9 Series
  http://www.microsoft.com/windows/windowsmedia/forpros/format/asfviewer.aspx
- Codes in FACILITY_ITF
  http://msdn.microsoft.com/en-us/library/ms679751.aspx
- Axis Application Development Partner Program
  http://www.axis.com/partner/adp_program/index.htm