



Elecard Low Delay SDK Reference Manual

(Evaluation version)

Notices

Elecard Low Delay SDK Reference Manual

First edition: July 2010

Date modified: November 24, 2011

For information, contact Elecard.

Tel: +7-3822-492-609; Fax: +7-3822-492-642

More information can be found at: www.elecard.com

For Technical Support, please contact the Elecard Technical Support Team: tsup@elecard.com

Elecard provides this publication “as is” without warranty of any kind, either expressed or implied.

This publication may contain technical inaccuracies or typographical errors. While every precaution has been taken in the preparation of this document, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained herein. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. Elecard may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time.

Other company, product, trademarks, and service names are trademarks or service marks of other companies or corporations.

Copyright © 2011 Elecard. All rights reserved.

CONTENTS

| | |
|---|-----------|
| 1. INTRODUCTION..... | 4 |
| 1.1 ABOUT THIS DOCUMENT..... | 4 |
| 1.1.1 Purpose..... | 4 |
| 1.1.2 Topics Covered..... | 4 |
| 1.1.3 Related Documentation..... | 4 |
| 1.2 PREFACE..... | 4 |
| 1.2.1 Documentation..... | 4 |
| 1.2.2 Components..... | 5 |
| 1.2.3 Base Classes..... | 5 |
| 1.2.4 Sample Applications..... | 5 |
| 1.3 REQUIREMENTS..... | 6 |
| 1.3.1 Hardware Requirements..... | 6 |
| 1.3.2 Software Requirements..... | 6 |
| 1.4 TECHNICAL SUPPORT..... | 6 |
| 2. GETTING STARTED..... | 7 |
| 2.1 INTRODUCTION..... | 7 |
| 2.2 INSTALLING ELECARD LOW DELAY SDK..... | 7 |
| 2.2.1 Uninstalling Elecard Low Delay SDK..... | 7 |
| 2.3 DIRECTSHOW FILTERS ACTIVATION..... | 7 |
| 2.3.1 Instance Activation..... | 8 |
| 2.3.2 Adding a DirectShow Filter to the Graph without Registration..... | 9 |
| 2.4 RUNNING ELECARD LOW DELAY SDK SAMPLE APPLICATIONS..... | 9 |
| 2.5 DESCRIBING LOW DELAY SDK FOLDER STRUCTURE..... | 9 |
| 3. DIRECTSHOW OVERVIEW..... | 10 |
| 3.1 INTRODUCTION..... | 10 |
| 3.2 DIRECTSHOW MAIN TERMS AND DEFINITIONS..... | 10 |
| 3.2.1 DirectShow Filter Types..... | 10 |
| 3.2.2 Data Flow Principle..... | 11 |
| 4. BUILDING SIMPLE FILTER GRAPHS..... | 12 |
| 4.1 INTRODUCTION..... | 12 |
| 4.2 SIMPLE PLAYER..... | 12 |
| 4.2.1 Building Simple Player..... | 12 |
| 5. SAMPLE APPLICATIONS..... | 15 |
| 5.1 INTRODUCTION..... | 15 |
| 5.1.1 Building sample applications..... | 15 |
| 5.2 ELECARD LOW DELAY SDK SAMPLE APPLICATIONS..... | 15 |
| 5.2.1 Low Delay Streamer..... | 16 |
| 5.2.2 Low Delay Client..... | 17 |

1. Introduction

1.1 About This Document

1.1.1 Purpose

This document provides an overview of the installation, set up and use of the Elecard Low Delay SDK. It includes information about the structure of the Elecard Low Delay SDK, provides a quick overview of the DirectShow fundamentals and features a detailed description of the components and sample applications.

1.1.2 Topics Covered

The following lists the topics covered in this document:

- **Section 1: Introduction** – provides a general overview of the SDK and describes the purpose of the document.
- **Section 2: Getting Started** – describes how to install, uninstall, and run the SDK. This section also provides information on the Elecard Low Delay SDK folder structure.
- **Section 3: DirectShow Overview** – provides a brief description of the basic concepts of the DirectShow system, including component model, filters, graphs, and more.
- **Section 4: Building Simple Filter Graphs** – provides information on how to build simple graphs using the SDK filters.
- **Section 5: Sample Applications** – describes samples included in the Elecard Low Delay SDK.

1.1.3 Related Documentation

In order to thoroughly understand the DirectShow® technology, we strongly recommend that you read the following documentation:

- **Microsoft® DirectShow® Programmer Reference** described in Microsoft® DirectX Software Development Kit. You can find these documents at: <http://www.microsoft.com>.

1.2 Preface

Elecard Low Delay SDK is a software kit for development of digital video processing applications for ultra low latency compression, decompression and transmission of video and audio content using the Elecard components within the Microsoft® DirectShow® technology.

The Elecard Low Delay SDK package includes the following:

1.2.1 Documentation

Elecard Low Delay SDK documentation – consists of the following documents:

- Elecard Low Delay SDK Reference Manual (this document)

- Elecard Components Reference Manuals
- Elecard Base Classes Reference Manual
- Elecard Module Configuration Programmer Guide

1.2.2 Components

This section provides a quick overview of the DirectShow filters and other components included in SDK package. For further details, see the Elecard Components Reference documentation.

Table 1. Elecard Low Delay SDK Components

| Component | Description | File Name |
|--|--|----------------------|
| Elecard AVC Video Decoder | Software-only decoding solution for ISO/IEC 14496 part 10 AVC / ITU-T Recommendation H.264 video streams. | eavcdec_hd.ax |
| Elecard MPEG-2 Video Decoder | DirectShow filter for software-only decoding MPEG-2 video (ISO/IEC 13818-2) and MPEG-1 video (ISO/IEC 11172-2) streams. | em2vd_hd.ax |
| Elecard MPEG Audio Decoder | DirectShow filter for the software-only decoding of MPEG-1, MPEG-2, MPEG-2.5 and LPCM audio streams. | emad.ax |
| Elecard AAC Audio Decoder | DirectShow filter for the software-only decoding of AAC and HE-AAC audio streams. | eaacd.ax |
| Elecard AVC Video Encoder | Software module for video encoding into AVC/H.264 (MPEG-4 Part 10, ISO/IEC 14496-10) streams. | eavcenc_hd.ax |
| Elecard MPEG-2 Video Encoder | DirectShow filter for video encoding into MPEG-2 (ISO/IEC 13818-2) streams. | em2venc_hd.ax |
| Elecard AAC Audio Encoder | DirectShow filter for audio encoding into AAC streams. | eaace.ax |
| Elecard MPEG Audio Encoder | DirectShow filter for audio encoding into MPEG streams. | emac.ax |
| Elecard NWSource-Plus | DirectShow filter for receiving media data from the network. It receives the RTP and UDP packets and feeds the filter graph with stream data contained in these packets. | enwsplus.ax |
| Elecard NWRenderer | DirectShow filter for broadcasting media data to the network. It is capable of sending RTP, UDP and TCP packets and supports the announcement of its data session via SAP (SDP) packets. | enwr.ax, enwr.dll |
| Elecard MPEG Push Demultiplexer | DirectShow filter for the software-only splitting of MPEG-1 System Streams, MPEG-2 Program Streams and MPEG-2 Transport Streams into video and audio streams. | empgpdmx.ax |
| Elecard MPEG Multiplexer | DirectShow filter that provides the MPEG-2 Transport Stream (TS) or MPEG-2 Program Stream (PS) generation. | empegmux.ax |
| Elecard Desktop Capture | DirectShow filter that provides capturing of the specified screen area. | edc.ax |
| Elecard Graph Viewer | Elecard Graph Viewer is a utility for the presentation of graphs built by any application. Elecard Graph Viewer allows the viewing and changing the filter properties, building of the filter graph (filter adding, deleting and connection), controlling of the graph state (run, stop, pause) and positioning in the media stream. | ElGViewer.dll |

1.2.3 Base Classes

Base Classes – a C++ class library that simplifies common tasks, appearing during development of multimedia applications, such as: DirectShow graphs building, filters and pins control etc. Base classes are used in SDK sample applications and are delivered in source form.

1.2.4 Sample Applications

The Elecard Low Delay SDK samples are simple applications that demonstrate capabilities of the Elecard components providing the highest data processing speed. The samples are written in C++. The following table

provides a brief overview of each sample. For further details, see Section Sample Applications.

Table 2. Elecard Low Delay SDK Sample Applications

| Sample Name | Description | File Name |
|--------------------|---|--------------------|
| Low Delay Streamer | Sample application that captures video from the specified screen area and audio from the specified PC audio device, transcodes the data into selected formats, multiplexes the encoded streams into MPEG-2 Transport Stream and broadcasts the stream to network clients. | eldstreamer.exe |
| Low Delay Client | Sample application for receiving and playback of video and audio data that is captured and broadcast with Low Delay Streamer. | LowDelayClient.exe |

1.3 Requirements

The Elecard Low Delay SDK has the following hardware and software requirements:

1.3.1 Hardware Requirements

Minimum hardware requirements:

- 2 GHz 32-bit (x86) or 64-bit (x64) processor
- 1 GB RAM (32-bit) or 2 GB RAM (64-bit)
- DirectX 9.0C (and higher) compatible VGA card

1.3.2 Software Requirements

- Windows® 2000/XP/2003 Server/Vista/7
- Microsoft® DirectShow® SDK (Microsoft Windows SDK version 7.1 is recommended)

Note: For Windows versions earlier than Vista, audio capturing is performed, if the system sound card supports Stereo mixer.

Note: Previous versions of the DirectShow SDK were included in the DirectX SDK. The last version of the DirectX SDK containing DirectShow was the DirectX 9.0 SDK Update - (February 2005) Extras. After this version, DirectShow was moved to the Windows SDK. To get the latest version of the DirectShow headers, libraries, and samples, download the Windows SDK from <http://msdn.microsoft.com>.

1.4 Technical Support

For technical support contact the Elecard Technical Support Team:

tsup@elecard.com

2. Getting Started

2.1 Introduction

The following section details the procedures for installing the Elecard Low Delay SDK. In addition, it provides the description of the Elecard Low Delay SDK folder structure.

Note: The described installation/uninstallation process is relevant for all versions of the Elecard Low Delay SDK.

2.2 Installing Elecard Low Delay SDK

To install the Elecard Low Delay SDK:

1. Run the Elecard Low Delay SDK setup. To run, double click the executable file from the Elecard Low Delay SDK setup package.
2. The *Elecard Low Delay SDK setup* window will appear. Read the recommendations and warnings. Click **Next**.
3. The Release Notes will appear. Click **Next**.
4. The license agreement will appear. Read the agreement and if you accept the terms within, select the “*Yes I agree with the terms of this license agreement*” check box. Click **Next**.
5. Select the destination folder in which you want to install the Elecard Low Delay SDK. Click **Next**.
6. Select the program group in which you want the Elecard Low Delay SDK to be located. Click **Next**.
7. To complete installation, follow the onscreen instructions. When setup has finished installing all of the necessary files on your computer, the appropriate message box with the text “*Elecard Low Delay SDK has been successfully installed*” will appear and the SDK is ready to use.

2.2.1 Uninstalling Elecard Low Delay SDK

To uninstall the Elecard Low Delay SDK application:

Click *Start*→*Programs*→*Elecard*→*Elecard Low Delay SDK*
→*Uninstall Elecard Low Delay SDK*.

Follow the onscreen instructions to complete removal of the application.

2.3 DirectShow Filters Activation

Most of the encoder and decoder (video and audio) filters from the SDK have a copy-protection mechanism: without activation these filters operate in an evaluation mode (e.g. overlay logo on the video). After activation filters operate in a demo mode (e.g. still overlay logo on the video, but without restrictions imposed on an expired mode). When the development is finished, the OEM pack with the components used in the product must be ordered (licensed) from Elecard.

There are two ways to perform activation:

1. Activation via *Registrator* using a special activation number. In this case every end-user should start the *Registrator* application and type the unique activation number. This activation method is intended for end-users.
2. "Instance Activation" from an application without any additional tools (without *Registrator*). This activation method is intended for OEM customers that distribute filters within their own applications.
3. Activation with the Elecard Module Config Checker filter using a special GUID. This activation method is intended for tests using the GraphEdit application. To utilize the Elecard Module Config Checker features, insert the filter into your filter graph, open its property page, select components for activation, and type an activation GUID.

Choosing the activation way depends on the development project requirements.

If it is required to redistribute the application and the codec plug-in separately, then it is reasonable to get the OEM pack as a special Installer with built-in Registrator. Certain parameter presets can be included as the default for each filter. So, the application does not need to reconfigure components. The package needs to be installed and activated by the end-user or by the application ('silent' installation and activation are available when you pass `-var:"SilentMode=1"` and `-var:"SerialNumber=xxxxxxxxxxxxxxxxxxxx"` as the Installer command-line parameters). Serial number is provided.

If the Elecard components must be included into the application install-pack, then it is reasonable to get the OEM pack as the filter binaries (*.ax files). When the application is installed, the binaries need to be placed to the known location on target computer. The filters can be registered in DirectShow environment and loaded in standard way. It is possible to use components without registration, but the only way to activate them is to use "Instance Activation" by a KEY_GUID inside the application. The KEY_GUID which is provided with the OEM pack is valid for all the filters in the package.

The direct filter usage without registration (is implemented in all SDK samples) is the most preferable from the security and stability point of view. Even if some other Elecard components are installed/uninstalled on target computer, the application uses its own set of binaries and does not depend on the current DirectShow environment.

2.3.1 Instance Activation

In the case of "Instance Activation" an application should pass a special GUID named KEY_GUID into the filter.

KEY_GUID is unique for every OEM customer and common for all filters supplied to him. Every filters update will be made with this KEY_GUID.

There is the sample code that should be integrated in the application for the filter activation.

```
HRESULT CBaseEncoder::ActivateFilter(CFilterWrapper* filter)
{
    HRESULT hr = E_FAIL;
    IModuleConfig* IMC;
    hr = filter->QueryInterface(&IMC);
    if (SUCCEEDED(hr))
    {
        IMC->SetValue(KEY_GUID, NULL);
        IMC->Release();
    }
}
```

```
return hr;
```

```
}
```

Where KEY_GUID is defined as:

```
static const GUID KEY_GUID = { 0x00000000, 0x0000, 0x0000, { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 } };
```

Zeros should be replaced by KEY_GUID assigned to a certain OEM customer.

2.3.2 Adding a DirectShow Filter to the Graph without Registration

Please note that placing files into the shared components folder is potentially unsafe. If the user installs an application which contains the same modules as your application, the original files can be overwritten. So, new components with other KEY_GUIDs will not be activated by your application any more.

The most reliable method of preventing the modules collision is using the particular folder for the filters supplied with your application.

It is recommended to insert filter in the graph without its registration in the Windows (without using *regsvr32*).

The following code shows how you can create a filter using the file path.

```
m_fVEncoder = CFilterWrapper(CLSID_Encoder, A2OLE(m_EncoderPath));
if (m_fVEncoder)
    m_hrError = AddFilter(m_fVEncoder, _T("Video Encoder"));
```

where *m_EncoderPath* is path to encoder file.

This method of filter creation ensures the use of the appropriate filters.

2.4 Running Elecard Low Delay SDK Sample Applications

To run the Elecard Low Delay SDK sample applications:

Click *Start*→*Programs*→*Elecard*→*Elecard Low Delay SDK*→*Bin* and select application.

2.5 Describing Low Delay SDK Folder Structure

After installing the Elecard Low Delay SDK, the *Elecard Low Delay SDK* folder will appear in the destination folder specified during installation.

The SDK folder contains:

1. **Bin** – contains the executable binaries for the SDK samples
2. **Doc** – includes all SDK-related documentation
3. **Base Classes** – includes the source codes of the Elecard Low Delay SDK base class libraries
4. **Components** – contains the SDK binary and header files (DS filters, engine DLLs, *.h files, *.inc files)
5. **Samples** – contains the source codes of the Elecard Low Delay SDK sample applications

3. DirectShow Overview

3.1 Introduction

The following section provides an overview of the DirectShow components, including filter types and data flow graphs. For detailed graph building instruction, see Section 4 Building Simple Filter Graphs.

3.2 DirectShow main terms and definitions

At the heart of [DirectShow®](#) is a modular system of pluggable components called *Filters*. These filters are arranged in a configuration called a *Filter Graph*. The *Filter Graph Manager* component oversees the connection of these filters and controls the data flow stream. Applications that use DirectShow architecture control the activities of the Filter Graph by communicating with the Filter Graph Manager.

Most of the filters included in Microsoft® DirectShow® runtime reside in quartz.dll, while others are standalone .AX files. The DirectShow filters are DLL files. Many of them are installed to Windows\system directory and others are installed in the products specific catalogs (for example, the Elecard filters are installed in the *Program Files\Common Files\Elecard* folder on the system disk).

The following is important to know about filter data streaming, connections, and pins:

- Data streams in packets (called *MediaSample*) from Source to Renderer through Transform filters.
- The connection from one filter to another is realized with the help of a *Pin*. A pin is an object that belongs to the filter. It provides a connection to other filter pins. The *Input Pin* receives a *MediaSamples* from upstream filter and the *Output Pin* sends the *MediaSamples* to downstream filter.
- The Source filter has at least one output pin, the Renderer filter has at least one input pin, and the Transform filter has both input and output pins.

3.2.1 DirectShow Filter Types

DirectShow filters have the following three main classes:

- **Source Filter** – provides the multimedia stream. It ranges from the File Source Filter to the MPEG encoding device source, or Network source.
- **Renderer Filter** – finishes a graph. The most common Renderer filters are Video Renderer and Audio Renderer, which play video and audio streams. A Renderer filter can also be a File Dump or File Writer filter and a Network Render filter.
- **Transform Filter** – the widest range of filters. All transformations on DirectShow® streams are made in transform filters. This type of filter is divided into Transform and TransInPlace filters. The TransInPlace filter differs from the Transform filter in memory allocation. It does not provide its own allocators, but uses ones from the upstream or downstream filter. It sends the same *MediaSample* that it received from the upstream to downstream filter and makes all data transformations in the *MediaSample* data buffer without changing its size.

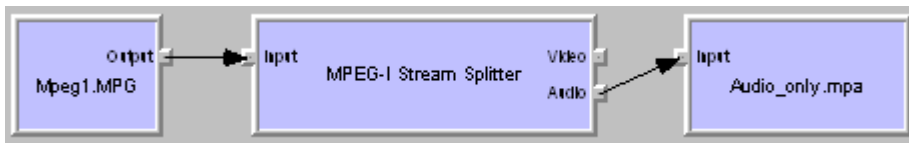
3.2.2 Data Flow Principle

The following describes the DirectShow data flow principle using the example of an audio stream being split from an MPEG stream and then dumped to a file.

For this task realization Filter Graph will be consist of 3 following filters:

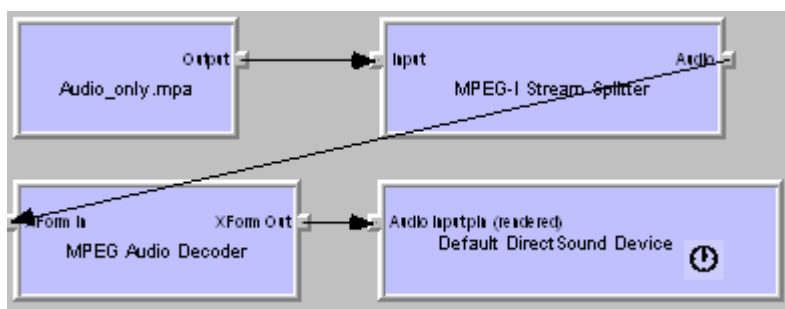
- **Async File Source** – gets data from the MPEG-1 file.
- **MPEG-1 Splitter** – splits MPEG-1 streams into MPEG-1 video and MPEG-1 audio elementary streams.
- **File Dump Filter** – writes compressed audio to the hard drive.

Figure 1. Audio Stream Splitting and Dumping to a File



Once the Audio file is split from the MPEG Stream and dumped into a file, it can easily be raised from the file, parsed, decoded and then played back by the following graph:

Figure 2. Dumped Audio File Playback Filter Graph



4. Building Simple Filter Graphs

4.1 Introduction

This section provides instructions for building simple graphs using Elecard Low Delay SDK filters.

4.2 Simple Player

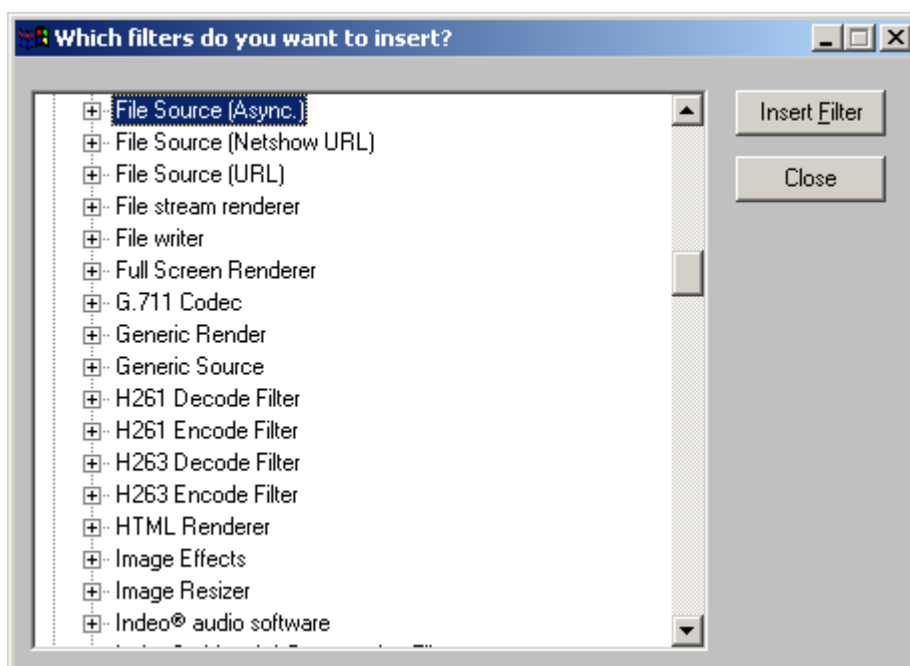
This Simple Player example demonstrates the building of the DirectShow graph for playback of an MPEG-1/MPEG-2 media files using the standard File Source filter and other Elecard developed filters such as: Elecard MPEG-2 Video Decoder.

4.2.1 Building Simple Player

To build the Simple Player:

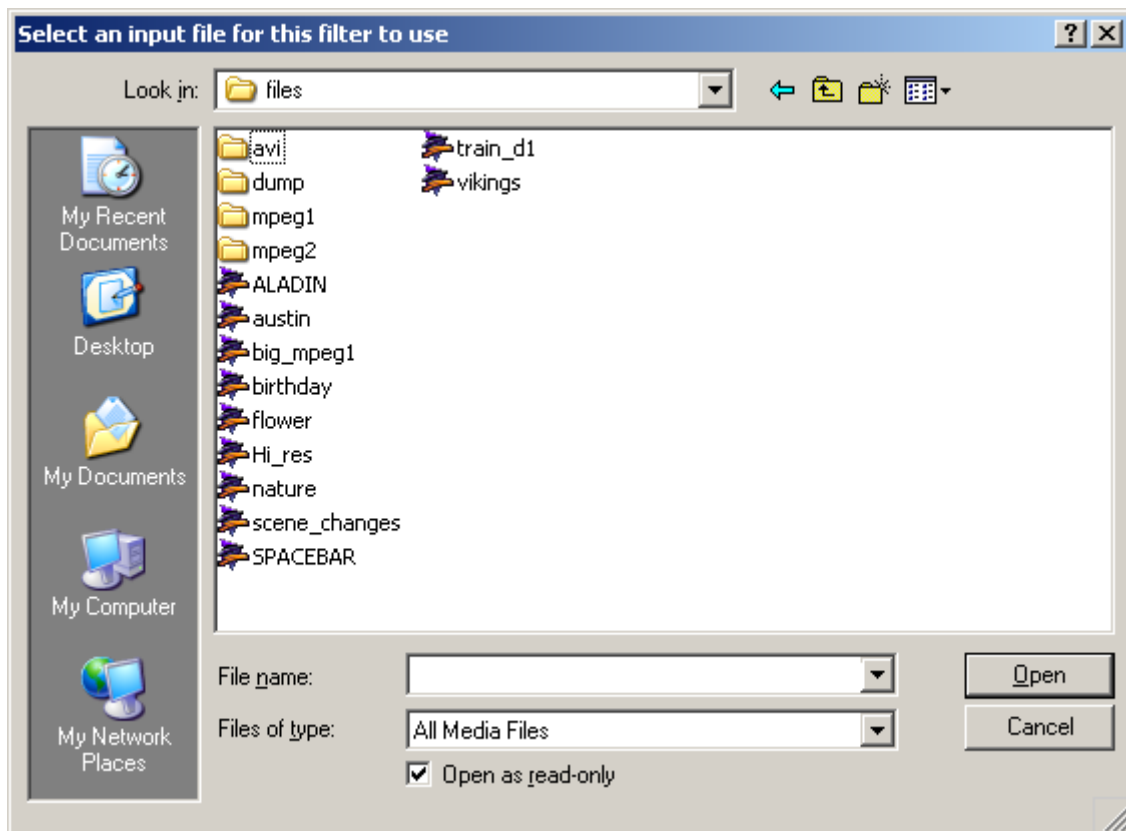
1. Start GraphEdit from the installed DirectX SDK.
(*Start*→*Programs*→*Microsoft DirectX SDK*→*DirectX Utilities*→ *GraphEdit*).
2. On the **Graph** menu, click **Insert Filters**.
3. From the “Which filters do you want to insert?” window, open DirectShow Filters, choose **File Source (Async.)** and click **Insert Filter**.

Figure 3. GraphEdit ‘Insert Filters’ Dialog



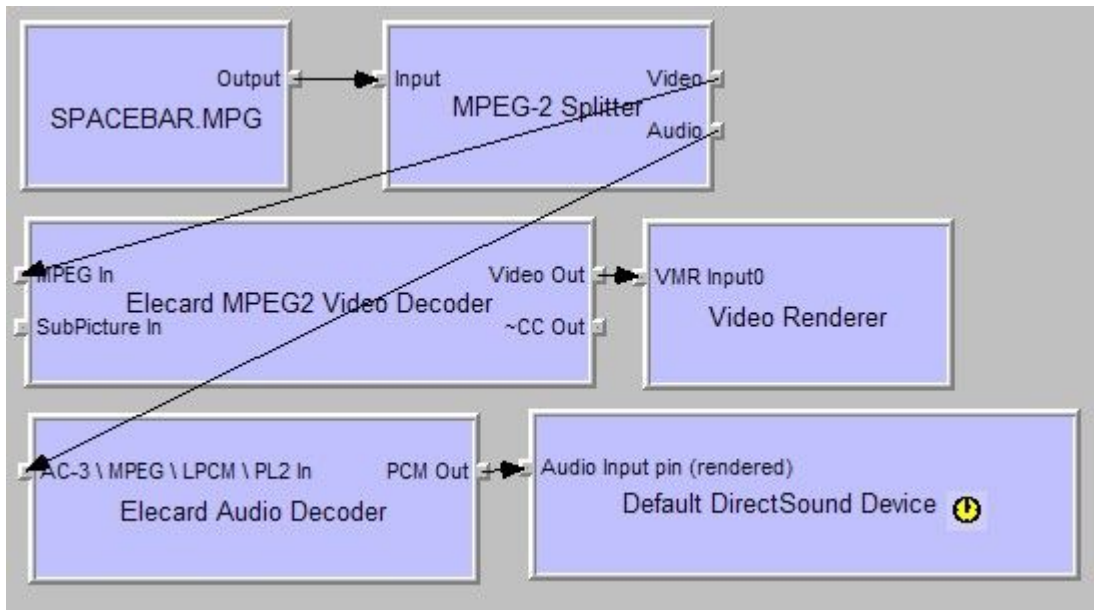
4. The “Select an input file...” window will appear. Choose the MPEG file you want to render.

Figure 4. File Source (Async.) 'Select an input file...' Dialog



5. Insert the following DirectShow filters into the graph (as described above in Step 3): Elecard MPEG Demultiplexer (or MS MPEG-2 Splitter), Elecard MPEG-2 Video Decoder.
6. Connect the File Source output pin with the Elecard MPEG Demultiplexer input pin.
7. Then, connect the Elecard MPEG Demultiplexer video pin with Elecard MPEG-2 Video Decoder input pin.
8. To create an automatic connection between Audio and Video Renderers, right-click the output pins of the Audio and Video Decoders and choose **Render** menu items.
9. The following graph will appear:

Figure 5. Simple Player Filter Graph



10. To start playback, on the **Graph** menu, click **Play**.

Note: For additional information on working with GraphEdit see the GraphEdit help.

5. Sample Applications

5.1 Introduction

This section describes the sample applications included in the Elecard Low Delay SDK package. They are available from the SDK program group (*Start*→*Programs*→*Elecard*→*Elecard Low Delay SDK*→*Samples*).

5.1.1 Building sample applications

Note: You need to install the Microsoft® DirectX® SDK **before** building samples included in the Elecard Low Delay SDK.

To build the sample applications it is necessary to perform the following actions:

1. Build and link the DirectShow BaseClasses library.

In DirectX 8.1 the sources of the Base Classes are stored in the directory: (*SDK root*)\Samples\Multimedia\DirectShow\BaseClasses

In DirectX 9.0 the sources of the Base Classes are stored in the directory: (*SDK root*)\Samples\C++\DirectShow\BaseClasses

Note: To avoid problems in the SDK samples building and linking, adjust the DirectShow BaseClasses project settings. In the Configuration Properties→C/C++ →Code Generation→RuntimeLibrary section set the Multi-threaded(/MT) value for Release configuration and the Multi-threaded Debug (/MTD) value for Debug configuration.

As the result the *strmbase.lib* and *strmbasd.lib* files will be created.

2. Add paths to BaseClasses header files, *strmbase.lib* and *strmbasd.lib* to Visual C++ settings. The alternative way is to put *strmbase.lib* and *strmbasd.lib* to the directory (*SDK root*)\lib\, the path of which also must be set in Visual C++ settings.
3. To build samples from Elecard Low Delay SDK open the Solution file (*Elecard Low Delay SDK root*)\Samples\<Sample>\,sample>.sln). In this Workspace there are all the projects of the SDK samples and the project of Elecard BaseClasses. All the necessary dependences among the projects are already set; so on building any sample all the necessary libraries of BaseClasses will be automatically built and linked. The relative paths to the Elecard BaseClasses are included into the MSVC project files as "Additional include directories". The paths to the libraries are also set in the MSVC project files as "Additional library path". If you want to replace a sample project, do not forget to change corresponding paths or replace the project together with BaseClasses keeping the structure of directories.
4. To build the any another SDK samples open the corresponding Workspace file and act in the same way as described in the previous item.

5.2 Elecard Low Delay SDK Sample Applications

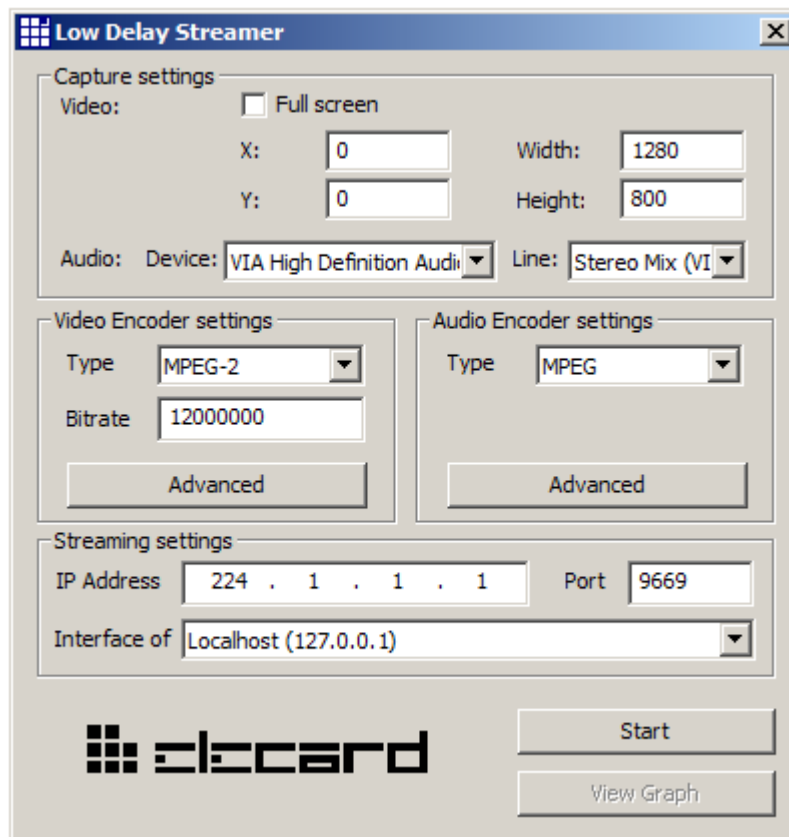
The following describes the Low Delay SDK Samples, including:

- Low Delay Streamer
- Low Delay Client

5.2.1 Low Delay Streamer

Low Delay Streamer is a sample application that captures video from the specified screen area and audio from the specified PC audio device, transcodes the data into selected formats, multiplexes the encoded streams into MPEG-2 Transport Stream and broadcasts the stream to network clients. Due to the special adjustment of the Elecard components (so-called 'low delay' mode) the data processing speed is considerably increased.

Figure 6. Low Delay Streamer GUI



Building of filter graph is performed after the **Start** button click. The **Stop** button click removes all components from the graph.

Video and audio capturing is performed with the Elecard Desktop Capture filter. The **Capture settings** group box of the application GUI represents the filter main features.

The **X** and **Y** parameters set the captured screen area top-left coordinates. The **Width** and **Height** parameters set the area dimensions. The **Full screen** option allows the full-screen capturing. The `EDC_VideoCaptureRect` parameter is used to pass the area coordinates to the Elecard Desktop Capture filter as the [left, top, right, bottom] array, where (left, top) and (right, bottom) are the top-left and bottom-right corner coordinates of the captured screen area, respectively.

To capture audio data, select audio device and input that provide the data.

To retrieve the list of available audio devices, use the `IPParamConfig::EnumValidValues()` method for the `EDC_AudioInputDevice` parameter.

If it is not proposed to capture audio, set the 'not set' value instead of the audio board name. In this case the `EDC_CaptureAudio` parameter value is set to 0 and the Elecard Desktop Capture filter has only one output pin

– video output pin.

For each audio board the list of available input lines is formed in the same way using `EDC_AudioInputLine` parameter.

The device list is compiled once in the `OnInitDialog()` function during the dialog initialization. While the input line list is updated every time, if a new device is selected. For Vista and later Windows versions there is the additional virtual *'Loopback'* line as the hardware StereoMix (What You Hear) line replacement.

Low Delay Streamer allows selection of the data encoding formats (video – MPEG-2 or AVC, audio – MPEG or AAC) and adjustment of the encoded video bitrate. The **Advanced** button opens the **Settings** dialog for the corresponding encoder.

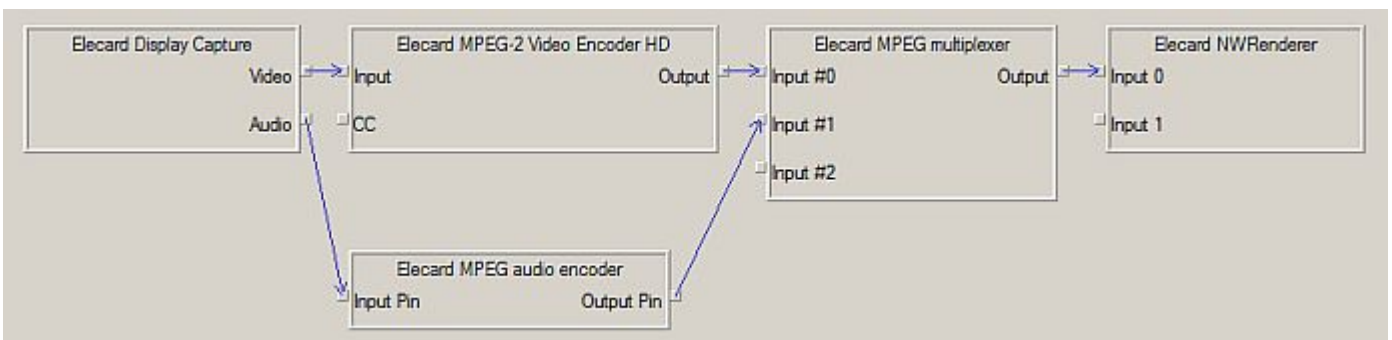
For the data broadcasting, the broadcast IP address, port and network interface must be specified. These values are set via the corresponding parameters of the Elecard NWRenderer filter.

The network interface is an IP address of the network card that is used for the data broadcasting. The list of network interfaces is compiled in the `OnInitDialog()` function (with the standard `GetAdaptersInfo()` function) during the dialog initialization and contains the following values:

- not set – the interface is not set, the network card for broadcasting is selected automatically
- Localhost (127.0.0.1) – local broadcasting, the network card is not used
- available network cards – the selected card IP address is set as the network interface

The **View Graph** button displays the built filter graph.

Figure 7. Low Delay Streamer Graph

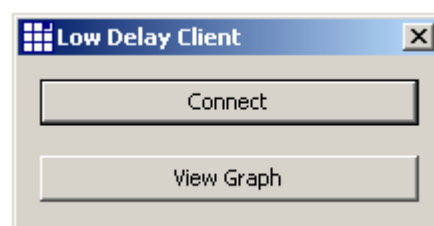


5.2.2 Low Delay Client

Low Delay Client sample application demonstrates receiving and playback of video and audio data that is captured and broadcast with **Low Delay Streamer**.

The application user interface is very simple and contains only two buttons.

Figure 8. Low Delay Client GUI



To build the broadcast receiving filter graph, click the **Connect** button.

The Elecard NWSource-Plus filter is used as a source filter that receives data from network. The filter receives the list of available announcements. The first announcement with the *Desktop* session name is selected from the list. Then the filter is configured according to the requested and received SDP data. If the announcement with the *Desktop* session name is not received during 3 seconds, it is accepted as data absence and the message *"Cannot build client graph. The 'Desktop' session is not available."* is displayed. Otherwise, the source filter is connected to the Elecard MPEG Push Demultiplexer filter and the graph is set to the *Running* state for initialization of the demultiplexer output pins.

The demultiplexer initialization is completed, if the EMPGPDMX_INITIAL_PARSING_DONE parameter value is equal to 1. If the initialization is not completed during 30 seconds, it is accepted as data absence and the message *"Cannot build client graph. The 'Desktop' session is not available."* is displayed.

If the received stream contains both audio and video data, extra latency is added for the playback synchronization. That is the EMPGPDMX_LATENCY_VALUE demultiplexer parameter value is set to 2 (corresponds to 60 milliseconds). If only video is received, the synchronisation is not needed and the EMPGPDMX_LATENCY_VALUE parameter value is equal to 0.

During the further graph building the demultiplexer is connected to decoders and the decoders are connected to the corresponding rendering filters.

To reduce delay time to the minimal value, the decoders must be adjusted in the proper way.

The **View Graph** button displays the built filter graph.

Figure 9. Low Delay Client Graph

