



Elecard Codec .NET SDK G4 v.1.4
Reference Manual

Notices

Elecard Codec .NET SDK G4 v.1.4 Reference Manual

First edition: May 2008

Date modified: August 29, 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 © 2008-2011 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	6
1.2.4 Sample Applications	6
1.3 REQUIREMENTS	7
1.3.1 Hardware Requirements	7
1.3.2 Software Requirements	7
1.4 TECHNICAL SUPPORT	7
2. GETTING STARTED	8
2.1 INTRODUCTION	8
2.2 INSTALLING ELECARD CODEC .NET SDK	8
2.2.1 Uninstalling Elecard Codec .NET SDK	8
2.3 DIRECTSHOW FILTERS ACTIVATION	8
2.3.1 Instance Activation	9
2.3.2 Adding a DirectShow Filter to the Graph without Registration	10
2.4 RUNNING ELECARD CODEC .NET SDK SAMPLE APPLICATIONS	10
2.5 DESCRIBING CODEC .NET SDK FOLDER STRUCTURE	10
3. DIRECTSHOW OVERVIEW	12
3.1 INTRODUCTION	12
3.2 DIRECTSHOW MAIN TERMS AND DEFINITIONS	12
3.2.1 DirectShow Filter Types	12
3.2.2 Data Flow Principle	13
4. BUILDING SIMPLE FILTER GRAPHS	14
4.1 INTRODUCTION	14
4.2 SIMPLE PLAYER	14
4.2.1 Building Simple Player	14
5. SAMPLE APPLICATIONS	17
5.1 INTRODUCTION	17
5.1.1 Building sample applications	17
5.2 DECODER SAMPLE APPLICATIONS	18
5.2.1 Simple Player	18
5.3 ENCODER SAMPLE APPLICATIONS	20
5.3.1 Simple Capture	20
5.3.2 Simple Encoder	22
5.3.3 Simple Muxer	23
5.4 NETWORK SAMPLE APPLICATIONS	25
5.4.1 NWPlayer	25
5.4.2 NWServer	29
5.4.3 NW_ServerTranscoder	32

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 Codec .NET SDK. It includes information about the structure of the Elecard Codec .NET 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 Codec .NET 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 Codec .NET 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

The Elecard Codec .NET SDK is a software development kit intended to enable programmers to develop digital video transcoding applications using the Elecard components within the Microsoft® DirectShow® technology.

The Elecard Codec .NET SDK package includes the following:

1.2.1 Documentation

Elecard Codec .NET SDK documentation – consists of the following documents:

- Elecard Codec .NET SDK Reference Manual (this document)
- Elecard Components Reference Manuals

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 Codec .NET 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_16k.ax eavcdec_hd.ax eavcdec_sd.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_16k.ax em2vd_hd.ax em2vd_sd.ax
Elecard MPEG-4 Video Decoder	Software-only decoding solution for MPEG-4 (ISO/IEC 14496-2) streams.	em4vdec.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_16k.ax eavcenc_hd.ax eavcenc_sd.ax eavcenc_cif.ax
Elecard MPEG-2 Video Encoder	DirectShow filter for video encoding into MPEG-2 (ISO/IEC 13818-2) streams.	em2venc_16k.ax em2venc_hd.ax em2venc_sd.ax em2venc_cif.ax
Elecard MPEG-4 Video Encoder	DirectShow filter for video encoding into MPEG-4 SP-ASP (ISO/IEC 14496-2) streams.	em4venc_hd.ax em4venc_sd.ax em4venc_cif.ax
Elecard MPEG Audio Encoder	DirectShow filter that provides audio encoding into MPEG format.	emac.ax
Elecard AAC Audio Encoder	DirectShow filter for the software-only encoding of AAC audio streams.	eaace.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 RTSP NetSource	DirectShow filter that sets the connection with RTSP server sends request for starting, stopping, pausing the media broadcasting and positioning in the stream, receives media data from RTSP server.	ertspnws.ax
Elecard Sink Filter	DirectShow filter that dumps a stream to a file. It supports the indexing of the MPEG-2 VES and MPEG-2 PS and splitting the encoded file into parts of a predefined size.	esf.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 TimeMarker	DirectShow transform filter for stream analysis. It defines the type of data, extracts time stamps from the stream and sets these time stamps on the output media samples.	ETimeMarker.ax
Elecard StreamPump	DirectShow filter for converting asynchronous source to synchronous.	estrp.ax
Elecard MPEG Demultiplexer	DirectShow filter for splitting of MPEG-1 System Streams (ISO/IEC 11172-1), MPEG-2 Program and Transport Streams (ISO/IEC 13818-1) into video and audio streams.	empgdmx.ax
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 LATM Demultiplexer	DirectShow filter that provides demultiplexing of AAC LOAS/LATM stream into elementary AAC streams.	elatmdmx.ax

Component	Description	File Name
Elecard MP4 Demultiplexer	DirectShow filter that provides demultiplexing of ISO/IEC 14496-14 file format (MP4) and 3GPP2 System streams into a MPEG-4, H.263, AVC/H.264 video streams and AAC, AMR, MPEG-1/2 Audio Layer 3 audio streams.	emp4demux.ax
Elecard MPEG Multiplexer	DirectShow filter that provides the MPEG-2 Transport Stream (TS) or MPEG-2 Program Stream (PS) generation.	empegmux.ax
Elecard MP4 Multiplexer	DirectShow filter intended for the generation of MPEG-4 (Intermedia Format (MP4)) System Streams.	emp4mux.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
Elecard InfTee	DirectShow filter that delivers data (samples) received by its input pin to an infinite number of output pins.	einftee.ax
Elecard DVB-ASI Source	DirectShow source filter that gets the data from DVB-ASI board and presents raw data for further processing.	edvbsdektec.ax
Elecard File List Source	DirectShow source filter that provides operations with a list of source files as with a single file.	epls.ax
Elecard Color Space Converter 2	DirectShow filter that provides conversion between different formats of uncompressed video (for example, from RGB 24-bit to YUV 16-bit color).	ecsc2.ax
Elecard Clear Work	DirectShow transform filter that provides video denoising. The component performs automatic detection of the high-frequency noise level and removes the noise according to the detected level.	ecwf.ax
Elecard Module Config Checker	DirectShow filter that provides activation of the Elecard components in the GraphEdit application (contained in the DirectShow package).	echecker.ax

1.2.3 Base Classes

Base Classes – a class library that conforms to the Microsoft .NET CLS (Common Language Specification) and simplifies common tasks, appearing during development of multimedia applications, such as: 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 Codec .NET SDK sample applications are divided into categories according to the components they use (e.g. decoders, encoders, network components). Each Sample is written in Managed C++, C# and Visual Basic languages (CP, CS and VB in the file names, respectively). The following table provides a brief overview of each sample. For further details, see the *Sample Applications* section.

Table 2. Elecard Codec .NET SDK Sample Applications

Sample Category	Sample Name	Description	File Name
Decoder	Simple Player	Plays media files and represents simple and basic functionality of MPEG player.	SimplePlayerCP.exe SimplePlayerCS.exe SimplePlayerVB.exe
Encoder	Simple Capture	Encodes stream from a capture device, multiplexes and dumps it to a file.	SimpleCaptureCP.exe SimpleCaptureCS.exe SimpleCaptureVB.exe
	Simple Encoder	Demonstrates work with the Elecard MPEG-2 Video Encoder, Elecard AVC Video Encoder, Elecard MPEG-4 Video Encoder, Elecard AAC Encoder and Elecard MPEG Audio Encoder filters and their main adjustment.	SimpleEncoderCP.exe SimpleEncoderCS.exe SimpleEncoderVB.exe
	Simple Muxer	Demonstrates work with the Elecard MPEG Multiplexer and Elecard MP4 Multiplexer filters and their main adjustment.	SimpleMuxerCP.exe SimpleMuxerCS.exe SimpleMuxerVB.exe

Network	NWPlayer	Receives streams from network using the Elecard NwSource-Plus filter or the Elecard RTSP NetSource filter.	NWPlayerCP.exe NWPlayerCS.exe NWPlayerVB.exe
	NWServer	Broadcasts media data to the network using Elecard NWRenderer Filter. It is capable of sending RTP, UDP, TCP, and supports the announcement of its data session via sending SAP (SDP) packets.	NWServerCP.exe NWServerCS.exe NWServerVB.exe
	NW_ServerTranscoder	Sample application that demonstrates transcoding and broadcasting media data to the network. The broadcasting capabilities are the same as in the NWServer application.	NW_ServerTranscoderCP.exe NW_ServerTranscoderCS.exe NW_ServerTranscoderVB.exe

1.3 Requirements

The Elecard Codec .NET SDK has the following hardware and software requirements:

1.3.1 Hardware Requirements

Minimum hardware requirements:

- SSE-enhanced CPU (Intel® Pentium III, Celeron, AMD® Athlon, Opteron etc.)
- 128 MB RAM
- Any VGA card

For real-time video encoding, we recommend the following:

- CPU Dual XEON 2.2 GHz
- 256 MB RAM

1.3.2 Software Requirements

- Windows® 2000/XP/2003 Server/Vista/7
- Microsoft® DirectShow® SDK (Microsoft Windows SDK version 7.1 is recommended)
- Microsoft® .NET Framework Version 1.1 or Microsoft® .NET Framework Version 2.0

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 Codec .NET SDK. In addition, it provides the description of the Elecard Codec .NET SDK folder structure.

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

2.2 Installing Elecard Codec .NET SDK

To install the Elecard Codec .NET SDK:

1. Run the Elecard Codec .NET SDK setup. To run, double click the executable file from the Elecard Codec .NET SDK setup package.
2. The *Elecard Codec .NET 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 Codec .NET SDK. Click **Next**.
6. Select the program group in which you want the Elecard Codec .NET 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 Codec .NET SDK has been successfully installed*” will appear and the SDK is ready to use.

2.2.1 Uninstalling Elecard Codec .NET SDK

To uninstall the Elecard Codec .NET SDK:

Click *Start*→*Programs*→*Elecard*→*Elecard Codec .NET SDK xx*
→*Uninstall Elecard Codec .NET SDK* (xx – the SDK version number).

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 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.

```
private void ActivateFilter(Filter fltr)
{
    ModuleConfig mc = fltr.GetConfigInterface();
    if (mc != null)
    {
        mc.SetParamValue(ref KEY_GUID, null);
        mc.Dispose();
    }
}
```

Where KEY_GUID is defined as:

```
Guid KEY_GUID = new Guid("00000000-0000-0000-0000-0000-00000000");
```

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 (see "*SDK\Samples\Decoders\cs\SimplePlayer\Form1.cs*" – function **MyPlayer.BuildVideoBranch**) shows how you can create a filter using the file path.

```
//Create Elecard AVC Decoder from file
Guid clsidAVCDecoder = new Guid(Elecard.ElUids.Filters.CLSID_EAVCDEC);
Filter avcDec = Filter.LoadFromFile(decoderPath, ref clsidAVCDecoder);
AddFilter(avcDec, "Elecard AVC Decoder");
ActivateFilter(avcDec);
```

Where the *decoderPath* parameter is the path of decoder file.

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

2.4 Running Elecard Codec .NET SDK Sample Applications

To run the Elecard Codec .NET SDK sample applications:

Click *Start*→*Programs*→*Elecard*→*Elecard Codec .NET SDK xx*→*Bin* and select application (xx – the SDK version number).

2.5 Describing Codec .NET SDK Folder Structure

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

The SDK folder has the following structure:

1. **Binaries** – contains the SDK binary files (DS filters, engine DLLs, samples executable binaries etc.).
2. **Base Classes** – contains the source codes of the Elecard Codec .NET SDK base class libraries.
3. **Doc** – contains all SDK-related documentation.
4. **Samples** – contains the source codes of the Elecard Codec .NET SDK sample applications:
 - **Decoder** – demonstrates the decoding components usage

- **Encoder** – demonstrates the encoding components usage
- **Network** – demonstrates the network components usage

The each samples folder contains the following sub folders:

- o **CP** – contains the Managed C++ source codes of the SDK specific sample applications.
- o **CS** – contains the C# source codes of the SDK specific sample applications.
- o **VB** – contains the Visual Basic source codes of the SDK specific 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\Elecard\<Product name>\Components* 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.

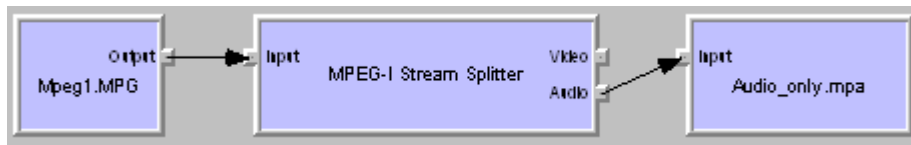
3.2.2 Data Flow Principle

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:

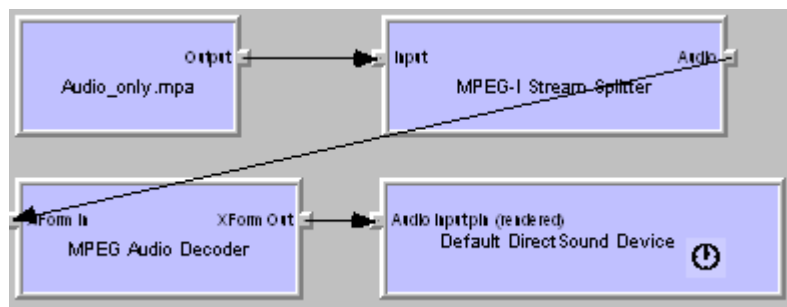
- **Async File Source** – gets data from the MPEG-1 file.
- **MPEG-1 Splitter** – splits MPEG-1 streams into MPEG-1 video and MPEG-1 audio elementary streams.
- **File Dump Filter** – writes compressed audio to the hard drive.

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 Codec .NET SDK filters.

4.2 Simple Player

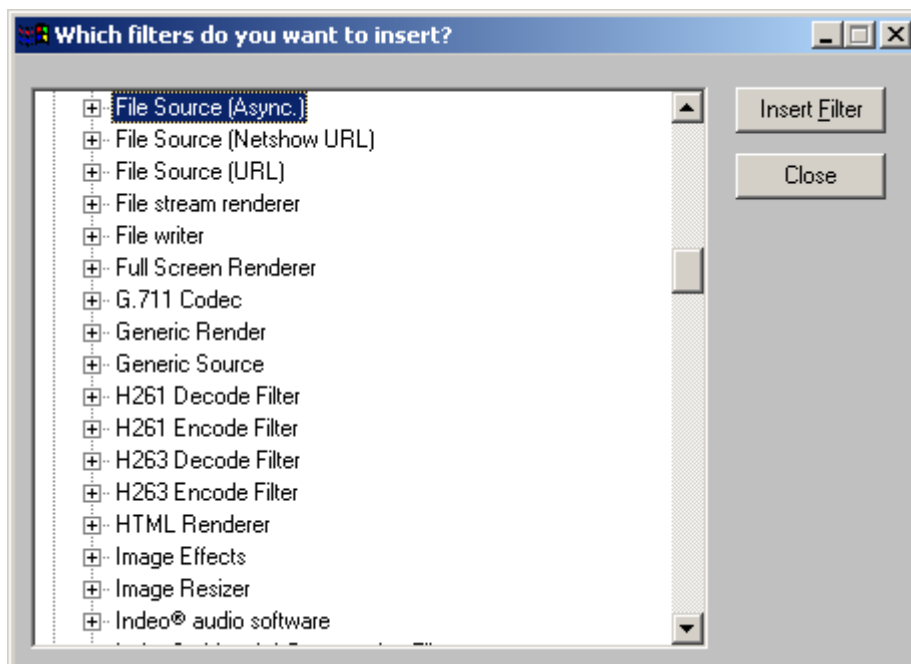
This Simple Player example demonstrates the building of the DirectShow graph for playback of an MPEG-1/MPEG-2 media files using the standard File Source filter and other Elecard developed filters such as: Elecard MPEG-2 Video Decoder.

4.2.1 Building Simple Player

To build the Simple Player:

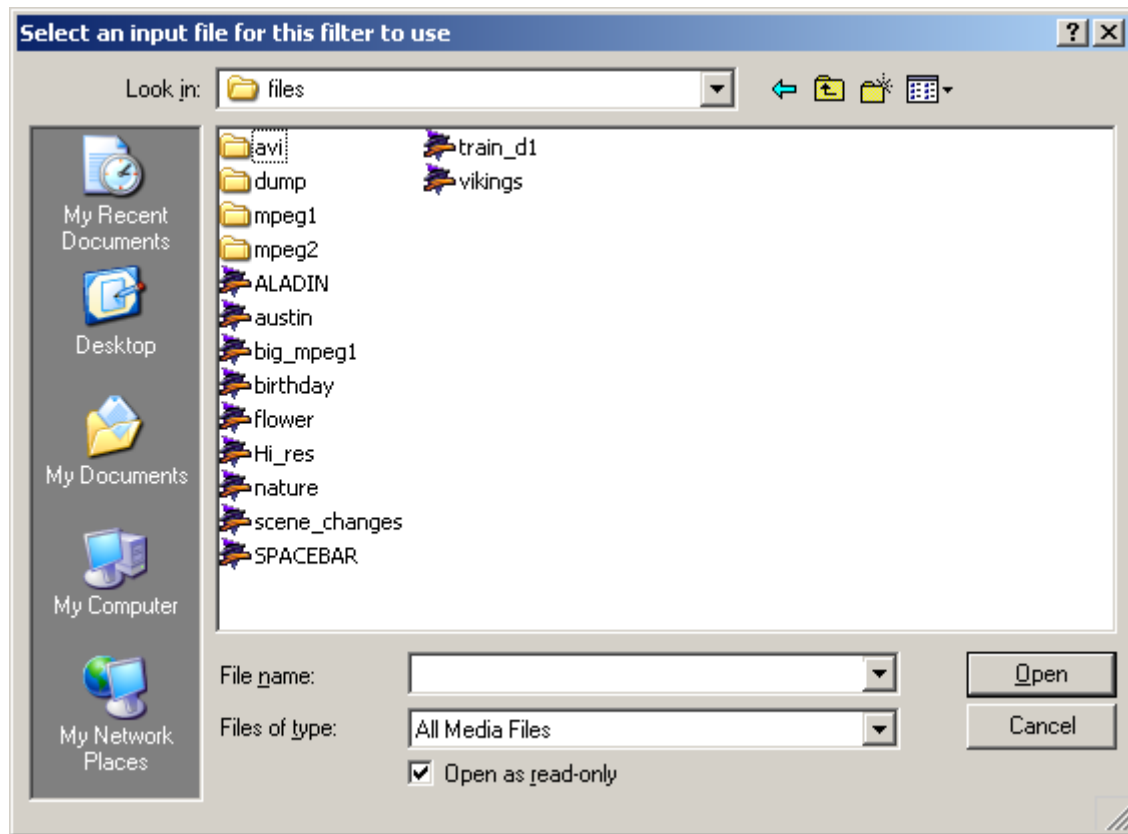
1. Start GraphEdit from the installed DirectShow SDK. (`..\Program Files\Microsoft SDKs\Windows\<version>\Bin\graphedt.exe`).
2. On the **Graph** menu, click **Insert Filters**.
3. From the “Which filters do you want to insert?” window, open DirectShow Filters, choose **File Source (Async.)** and click **Insert Filter**.

Figure 3. GraphEdit ‘Insert Filters’ Dialog



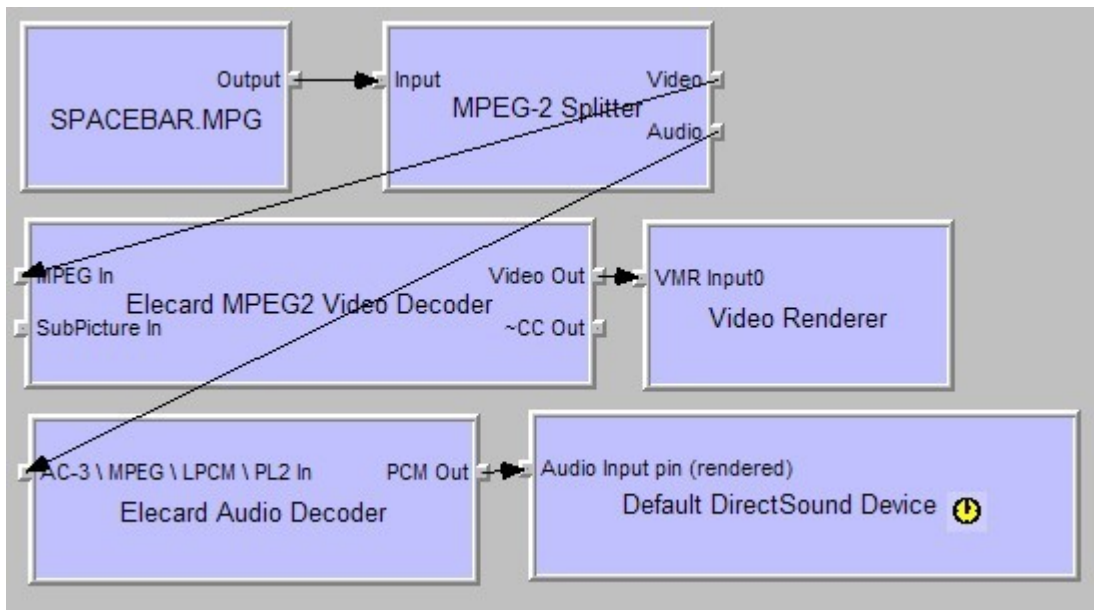
4. The “Select an input file...” window will appear. Choose the MPEG file your want to render.

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 Codec .NET SDK package. They are available from the SDK program group (*Start*→*Programs*→*Elecard*→*Elecard Codec SDK*→*Samples*).

The Elecard Codec .NET SDK Sample Applications are divided into the following categories:

- Decoder Samples
- Encoder Samples
- Network Samples

Each category contains three solutions – *v8.sln* (for work with Microsoft Visual Studio 2005), *<SolutionName>_v9.sln* (for work with Microsoft Visual Studio 2008) and *<SolutionName>_v10.sln* (for work with Microsoft Visual Studio 2010).

5.1.1 Building sample applications

Note: You need to install the Microsoft® DirectShow® SDK **before** building samples included in the Elecard Codec .NET SDK.

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

1. Build and link the DirectShow BaseClasses library.

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

In the 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 building and linking of the SDK samples, adjust the DirectShow BaseClasses project settings.

In the Configuration Properties→General->Character Set section set the Use Multi-Byte Character Set value. 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.

In the Configuration Properties→C/C++ →Language->Treat wchar_t as Built-in Type section set the Yes value.

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 Microsoft Visual Studio 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 Microsoft Visual Studio settings.
3. To build sample applications, open the Solution file (*Elecard Codec .NET SDK root*)\Samples\Decoders\DecoderSamples_<Visual Studio version>.sln. This solution contains all projects of the SDK

samples and two satellite projects - Elecard Codec .NET SDK Classes and Elecard Codec .NET SDK Consts. All the necessary dependences among the projects are already set; so on building any sample all the necessary assemblies will be automatically built. Satellite projects Elecard Codec .NET SDK Classes and Elecard Codec .NET SDK Consts are included in the solution to clarify the .NET SDK dependencies and make first usage of SDK more convenient. It is not necessary to include these projects into your own applications solutions. You should just add references to the compiled Classes and Consts assemblies. Thus, after the Elecard Codec .NET SDK Classes and Consts projects are built for all configurations (Release, Debug) you can remove them from the Samples solution and add references to Elecard Codec .NET SDK Classes and Consts assemblies. The built Elecard Codec .NET SDK Classes assembly is stored in the directory (*Elecard Codec .NET SDK root*)\Common\Classes and the built Elecard Codec .NET SDK Consts assembly is stored in the directory (*Elecard Codec .NET SDK root*)\Common\Consts.

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 Decoder Sample Applications

The following describes the Decoder samples, including:

- Simple Player

5.2.1 Simple Player

Simple Player is a sample application that plays media files and represents simple and basic functionality of an MPEG player.

5.2.1.1 Description

Simple Player application consists of one class inherited from the **Form** class:

```
public class Form1 : Form {};
```

One of the class members is the **player** – instance of the **MyPlayer** class (inherited from the **GenericPlayer** class) which performs all playback operations. It is also responsible for all graph routines. It is created every time a media file is opened. Thus, there is a **player** instance for every opened file. The instance related to an old file is destroyed before opening a new media file. **InitializeEvents()** method is called to enable the application handle the graph events. When one of the following events takes place, a relative handler function would be called. The graph events handled by the Simple Player application are:

- IdleEventHandler
- StartingEventHandler
- PausingEventHandler
- StoppingEventHandler
- CompletedEventHandler

Playback states are changed with the help of following functions – members of the **GenericPlayer** Class: **player.Run()**, **player.Pause()** and **player.Stop()**.

A media file positioning is implemented in Simple Player application. It is possible to change position during the playback as well as in pause or stop states. To change a position in a stop state, firstly it is necessary to change the graph state into pause. A **TrackBar** object is inserted into the application to reflect a current position inside a media file. All the positioning routines are handled by the **MediaSeeking** class (described in .NET Base Classes). There is the **MediaSeeking** class instance in the **Graph** class which is inherited by the

GenericPlayer class. There are two ways to get this instance inside the application:

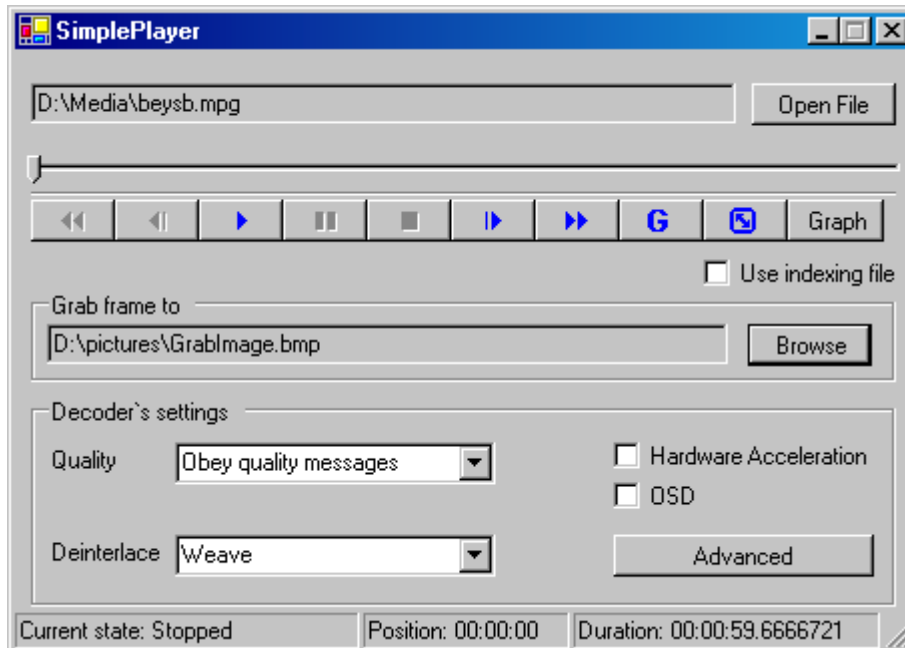
```
MediaSeeking mediaSeeking = Graph.op_MediaSeeking(player);
```

or

```
MediaSeeking mediaSeeking = player.ToMediaSeeking();
```

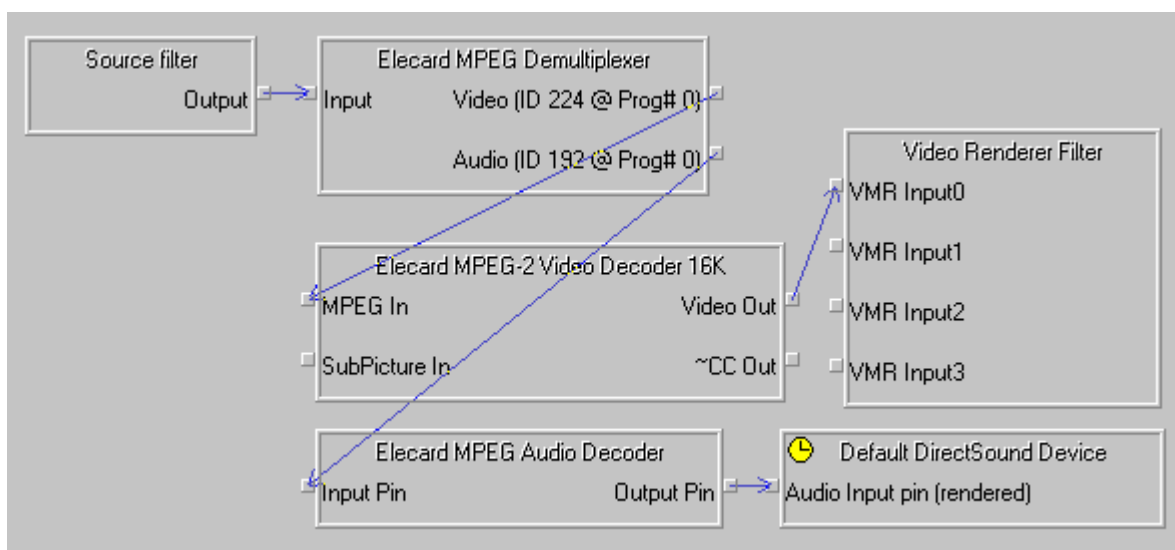
After you get the **mediaSeeking** instance you can easily do positioning inside a media file. For example, to get or set the position value the property **mediaSeeking.Position** is used. The read-only property **mediaSeeking.Duration** is called to get the opened file duration.

Figure 6. Simple Player GUI



All visual controls are initialized after the application is loaded and are used to control the media file playback.

Figure 7. Sample Filter Graph for MPEG-2 File Decoding



5.2.1.2 Path

Managed C++:

Sources (Elecard Codec .NET SDK)\Samples\Decoder\cp\SimplePlayer

Binaries	(Elecard Codec .NET SDK)\Binaries\SimplePlayerCP.exe
C#:	
Sources	(Elecard Codec .NET SDK)\Samples\Decoder\cs\SimplePlayer
Binaries	(Elecard Codec .NET SDK)\Binaries\SimplePlayerCS.exe
Visual Basic:	
Sources	(Elecard Codec .NET SDK)\Samples\Decoder\vb\SimplePlayer
Binaries	(Elecard Codec .NET SDK)\Binaries\SimplePlayerVB.exe

5.2.1.3 Features

Simple Player has the following features:

- Media file playback
- Basic player functionality (start, stop, fast forward, fast backward, forward and backward steps, etc)
- Positioning in 'start', 'stop', and 'pause' modes
- Adjustment of basic parameters of Elecard video decoders using the Player main window controls
- Access to all Elecard video decoders settings (the **Advanced** button)
- Image grabbing
- Support of index files (creation and loading) for MPEG-2 streams

5.3 Encoder Sample Applications

The following describes the Encoder samples, including:

- Simple Capture
- Simple Encoder
- Simple Muxer

5.3.1 Simple Capture

Simple Capture is a sample application that encodes stream from a capture device(video into MPEG-2 format, audio into MPEG format), multiplexes it into MPEG-2 format and dumps it to a file.

5.3.1.1 Description

This sample demonstrates work with any capture devices supported by the system. Only one capture device is enabled at once. In case of digital (DV) devices it allows the camcorder control (start playback, stop and rewind).

An inheritor of the **CaptureGraph** class (described in Elecard Codec .NET SDK BaseClasses Reference Manual) plays the main part in the application. It has two principal methods **BuildPreviewGraph()** and **BuildCaptureGraph()** which are responsible for the graph building process. As the **BuildPreviewGraph()** method is intended for video preview, it is not used in audio capturing process.

Figure 8. Simple Capture GUI



5.3.1.2 Path

Managed C++:

Sources (Elecard Codec .NET SDK)\Samples\Encoder\cp\SimpleCapture
Binaries (Elecard Codec .NET SDK)\Binaries\SimpleCaptureCP.exe

C#:

Sources (Elecard Codec .NET SDK)\Samples\Encoder\cs\SimpleCapture
Binaries (Elecard Codec .NET SDK)\Binaries\SimpleCaptureCS.exe

Visual Basic:

Sources (Elecard Codec .NET SDK)\Samples\Encoder\vb\SimpleCapture
Binaries (Elecard Codec .NET SDK)\Binaries\SimpleCaptureVB.exe

5.3.1.3 Features

Simple Capture has the following features:

- Selection of capture device
- Access to all of the Elecard encoders and multiplexer settings (the **Advanced** buttons)
- Access to all of the capture device and crossbar filter settings
- Independent control of the Preview and Capture graph branches
- Preview of the built filter graph

5.3.2 Simple Encoder

Simple Encoder is a sample application that encodes media stream (video - into AVC, MPEG-2 or MPEG-4 format, audio - into AAC or MPEG format) and dumps it to a file using Elecard Sink Filter.

5.3.2.1 Description

This sample demonstrates work with the Elecard MPEG-2 Video Encoder, Elecard AVC Video Encoder, Elecard MPEG-4 Video Encoder, Elecard AAC Encoder and Elecard MPEG Audio Encoder filters and their basic adjustment. Encoding settings may be modified with the controls located on the application main form. This sample allows the encoded video stream preview and the graph state and the current file size tracking.

Figure 9. Simple Encoder GUI

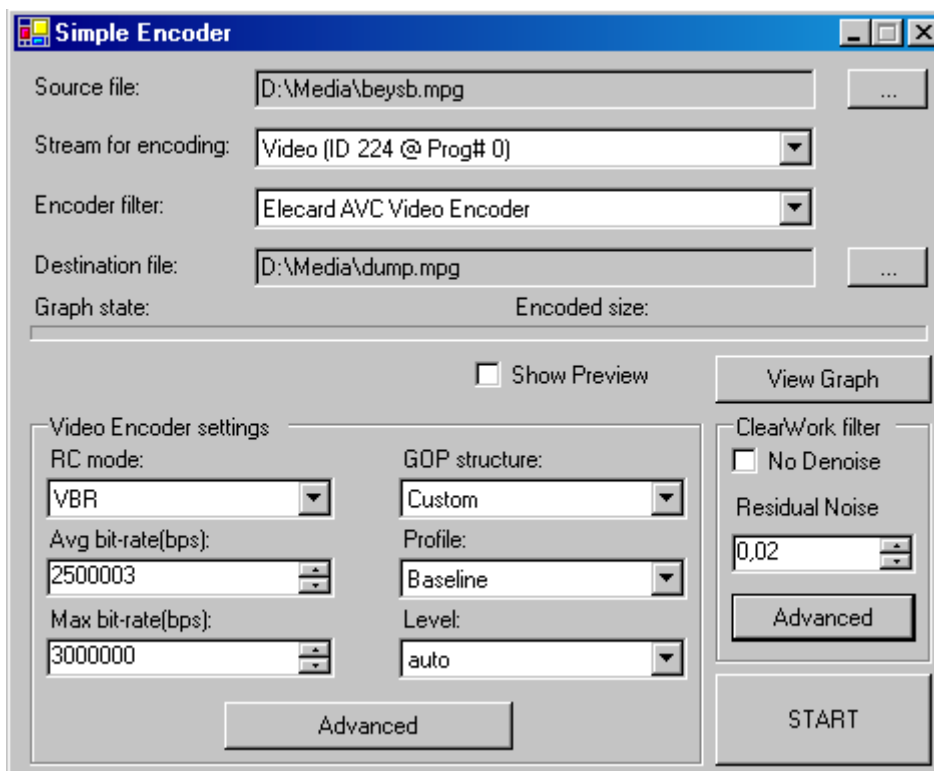


Figure 10. Sample Filter Graph for MPEG→AAC Audio Transcoding

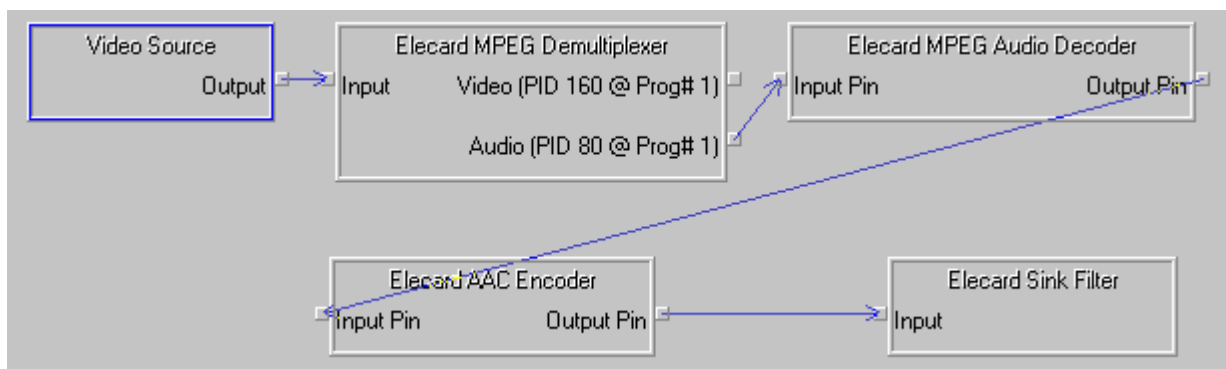
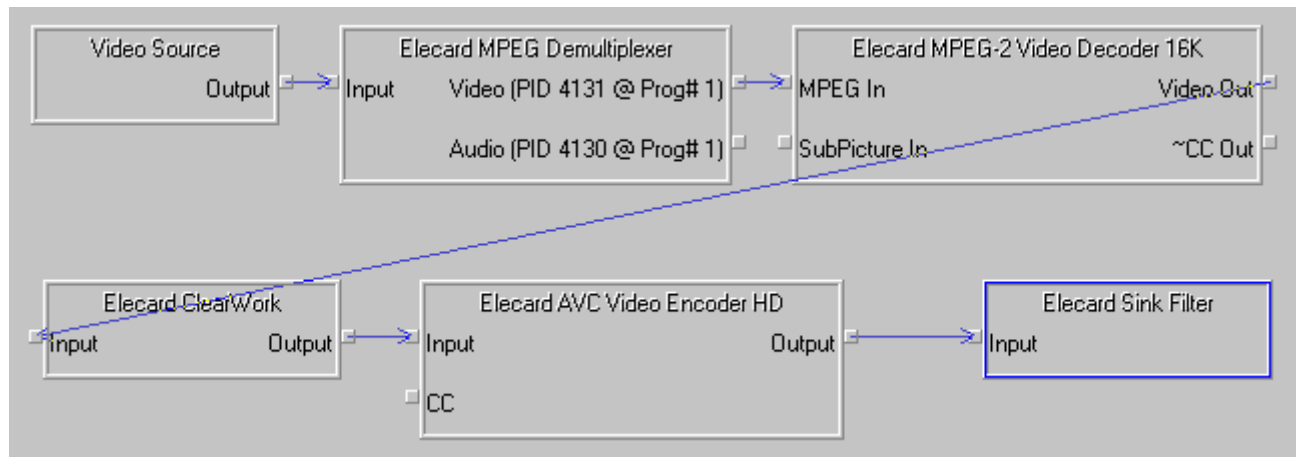


Figure 11. Sample Filter Graph for MPEG-2→AVC Video Transcoding



5.3.2.2 Path

Managed C++:

Sources (Elecard Codec .NET SDK)\Samples\Encoder\cp\SimpleEncoder

Binaries (Elecard Codec .NET SDK)\Binaries\SimpleEncoderCP.exe

C#:

Sources (Elecard Codec .NET SDK)\Samples\Encoder\cs\SimpleEncoder

Binaries (Elecard Codec .NET SDK)\Binaries\SimpleEncoderCS.exe

Visual Basic:

Sources (Elecard Codec .NET SDK)\Samples\Encoder\vb\SimpleEncoder

Binaries (Elecard Codec .NET SDK)\Binaries\SimpleEncoderVB.exe

5.3.2.3 Features

Simple Encoder has the following features:

- Selection of stream for encoding from the list of available streams
- Encoding video into AVC, MPEG-2 or MPEG-4 format
- Encoding audio into AAC or MPEG format
- Adjustment of basic parameters of Elecard video encoders using the Encoder main window controls
- Access to all of the Elecard encoders settings (the **Advanced** button)
- Video denoising with the ClearWork filter
- Preview of the encoded video

5.3.3 Simple Muxer

Simple Muxer is a sample application that demonstrates work with the Elecard MPEG Multiplexer and Elecard MP4 Multiplexer filters and their main adjustment.

5.3.3.1 Description

This sample allows work with multiplexers in the *Remux* and *Encode* modes.

In the *Remux* mode the source stream is remultiplexed into MPEG-2 TS or MP4 format without transcoding. In

the *Encode* mode the source video and audio streams are encoded into AVC and AAC formats with further multiplexing into MPEG-2 TS or MP4 format. Only one video and one audio streams are supported.

Figure 12. Simple Muxer GUI

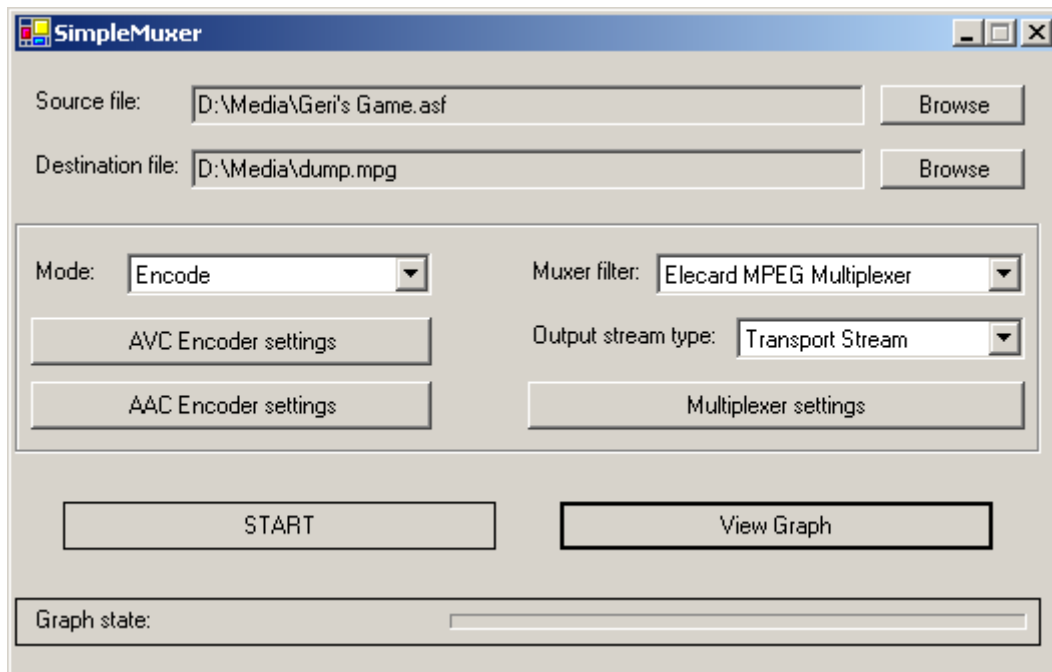


Figure 13. Sample Filter Graph for Remultiplexing

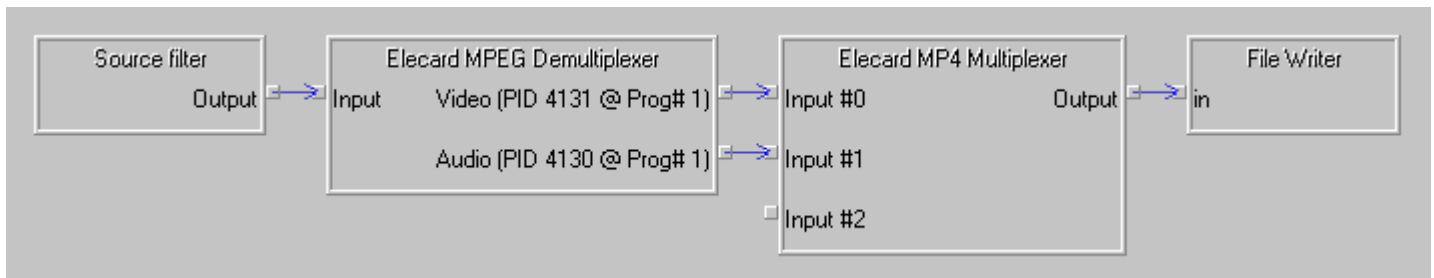
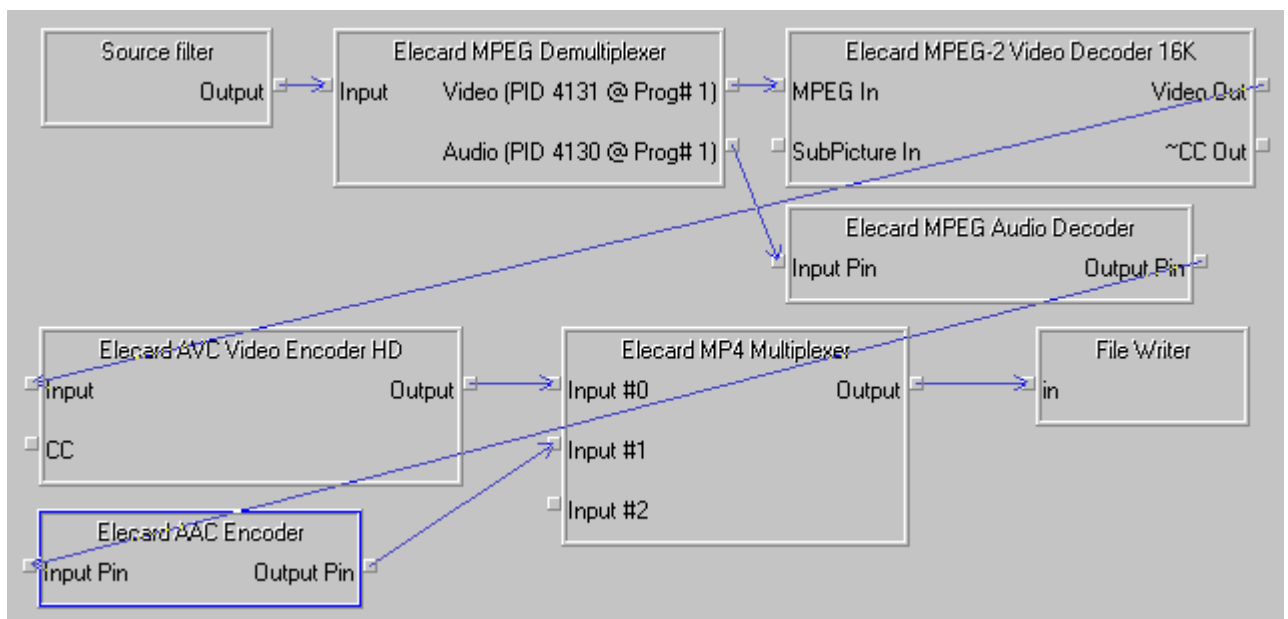


Figure 14. Sample Filter Graph for Transcoding



5.3.3.2 Path

Managed C++:

Sources (Elecard Codec .NET SDK)\Samples\Encoder\cp\SimpleMuxer

Binaries (Elecard Codec .NET SDK)\Binaries\SimpleMuxerCP.exe

C#:

Sources (Elecard Codec .NET SDK)\Samples\Encoder\cs\SimpleMuxer

Binaries (Elecard Codec .NET SDK)\Binaries\SimpleMuxerCS.exe

Visual Basic:

Sources (Elecard Codec .NET SDK)\Samples\Encoder\vb\SimpleMuxer

Binaries (Elecard Codec .NET SDK)\Binaries\SimpleMuxerVB.exe

5.3.3.3 Features

Simple Muxer has the following features:

- Work mode selection (*Remux* or *Encode*)
- Video/audio encoding into AVC/AAC formats (in the *Encode* mode)
- Multiplexing of video and audio into MPEG-2 Transport Stream or MP4 format
- Access to all Elecard AVC Video Encoder settings (in the *Encode* mode)
- Access to all Elecard AAC Audio Encoder settings (in the *Encode* mode)
- Access to all multiplexer filter (Elecard MPEG Multiplexer or Elecard MP4 Multiplexer) settings

5.4 Network Sample Applications

The following describes the Network samples, including:

- NWPlayer
- NWServer
- NW_ServerTranscoder

5.4.1 NWPlayer

NWPlayer is a sample application that receives streams from network using the Elecard NwSource-Plus filter or the Elecard RTSP NetSource filter. A choice between the source filters depends on the assigned task. The Elecard NwSource-Plus filter allows user to receive the list of broadcasted streams, to choose one stream from the list and to play it in the video window. The Elecard RTSP NetSource filter is used to play files from Video On Demand (VOD) server. The NW Player uses the **PlayerView** class (described in Elecard .NET BaseClasses Reference Manual) and has the video window placed on the application main window.

5.4.1.1 Description

NW Player consists of three classes: **Form1** class (derived from the **Form** class), **NWPlayer** class(derived from the **GenericPlayer** class) and **Announces** class.

```
public class Form1 : Form {};  
public class NWPlayer : GenericPlayer {};
```

```
public class Announce {};
```

Also NW Player contains the `NWSourceType` enumerator which specifies a source filter type.

```
public enum NWSourceType
{
    NWSourcePlus      = 0x00,
    RTSPNetSource     = 0x01,
};
```

Form1 is the main class. It contains instance of **NWPlayer** class, which is responsible for filter graph building and runtime playback control.

When the application is initialized process or if the source filter is changed, the `CreateNWPlayer()` static method of the **NWPlayer** class is called. It returns the instance of the **NWPlayer** class only in case if the selected source filter was successfully created and added to the graph. The default source filter is Elecard NwSource-Plus.

The Elecard NWSource-Plus Filter plays a main part in the application functionality. It is used in receiving announces from the network, choosing and playing streams from the network. Therefore it is created and added in the filter graph right after the application load. To get or set the `NWSourceFilter` settings the **ModuleConfig** interface is used. This interface is received from `NWSourceFilter` by calling the **GetConfigInterface** method:

```
ModuleConfig mc = player.NWSourceFilter.GetConfigInterface();
```

So now we are able to get or set the `NWSourceFilter` settings using instance of the **ModuleConfig** class. In this sample it is used to get the announces list from the various servers and configure `NWSourceFilter` to receive data from the chosen server. To get the servers announces the **GetParamValue** method of the **ModuleConfig** class is called:

```
Guid clsidENSAnnouncesExt = new Guid(Elecard.ElUids.Properties.ENS_announces_bstr);

Object ans = mc.GetParamValue(ref(clsidENSAnnouncesExt));
```

This method returns an object, which represents a structure. The first 4 bytes of this structure contains information about announces amount. In the sample this information is used to organize a cycle on the servers announces. The next bytes of the structure contain the servers announces. To get each server information an instance of the **Announce** class is created during each cycle pass:

```
Int64 announcesExt = (Int64)ans;
IntPtr announcesExtIntPtr = (IntPtr)announcesExt;
if (announcesExt != 0)
{
    int count = (int)Marshal.ReadInt32(announcesExtIntPtr);
    announcesExtIntPtr = (IntPtr)(announcesExt + 4);
    announcesExtIntPtr = Marshal.ReadIntPtr(announcesExtIntPtr);
    for(int i = 0; i < count; i++)
    {
        Announce curAnnounce = new Announce(ref announcesExtIntPtr);
    }
}
```

A pointer to the memory block containing the current announce is passed as the parameter to the **Announce** class constructor method. The **Announce** class is used to get the parameters of the server from which the announce was received. The parameters – public members of the **Announce** class are listed below:

```
public TransportType transport;
public StreamType payload;
public String sessionName;
public String sessionInformation;
public String description;
public String email;
public String phone;
public String bandwidthInformation;
public String mCastAddress;
public String ipInterface;
public String serverAddress;
public String mCastPort;
public short timeToLive;
public String sessionID;
public String SDPdata;
```

TransportType and StreamType are enumerations, defined in the Base Classes.

Each parameter's value is obtained by moving the pointer in the announce structure by a certain value. Thus, after the **Announce** class constructor call the pointer is moved to the begging of a memory block which contains next announce received from the network.

Each instance of the **Announce** class is placed in the **announceList** member of the **Form1** class. It will later be used to display the servers information in the listView control, choose the server to play stream from by setting the server parameters taken from the corresponding announce with the help of **ModuleConfig** interface. (Double click in the **listView** control starts the data receiving from the chosen server.)

To receive contemporary information regarding to the broadcast streams, create the **ModuleCallbackNotify** object, specify the component for event tracking, and subscribe to receiving of the **ModuleNotify** event notifications:

```
if (player != null) {
    nwSourceConfig = player.NWSourceFilter.GetConfigInterface();

    if (nwSourceConfig != null) {
        callbackNotify = new ModuleCallbackNotify();

        callbackNotify.ModuleNotify += new ModuleCallbackNotify.ModuleNotifyEventHandler(
            ModuleCallbackNotifyEvent);

        nwSourceConfig.RegisterForNotifies(callbackNotify);
    }
}
```

The *nwSourceConfig* and *callbackNotify* objects must be the **Form1** class members because it is needed to unsubscribe *callbackNotify* from the Elecard NWSource-Plus filter event notifications before the graph rebuilding (when Elecard RTSP NetSource is selected as the source filter):

```

if (player != null) {
    if (nwSourceConfig != null) {

        if (callbackNotify != null) {
            nwSourceConfig.UnRegisterFromNotifies(callbackNotify);
            callbackNotify.Dispose();
            callbackNotify = null;
        }
        nwSourceConfig.Dispose();
        nwSourceConfig = null;
    }
    player.Dispose();
    player = null;
}
}

```

The Elecard RTSP NetSource filter requests the selected media file from VOD server with string of the following type:

```
elecard_rtsp://server_address:port[/data_path]
```

where

server_address – server IP address

port – number of port used for RTSP connections (unsigned integer value from 1 to 65535; default value is 554)

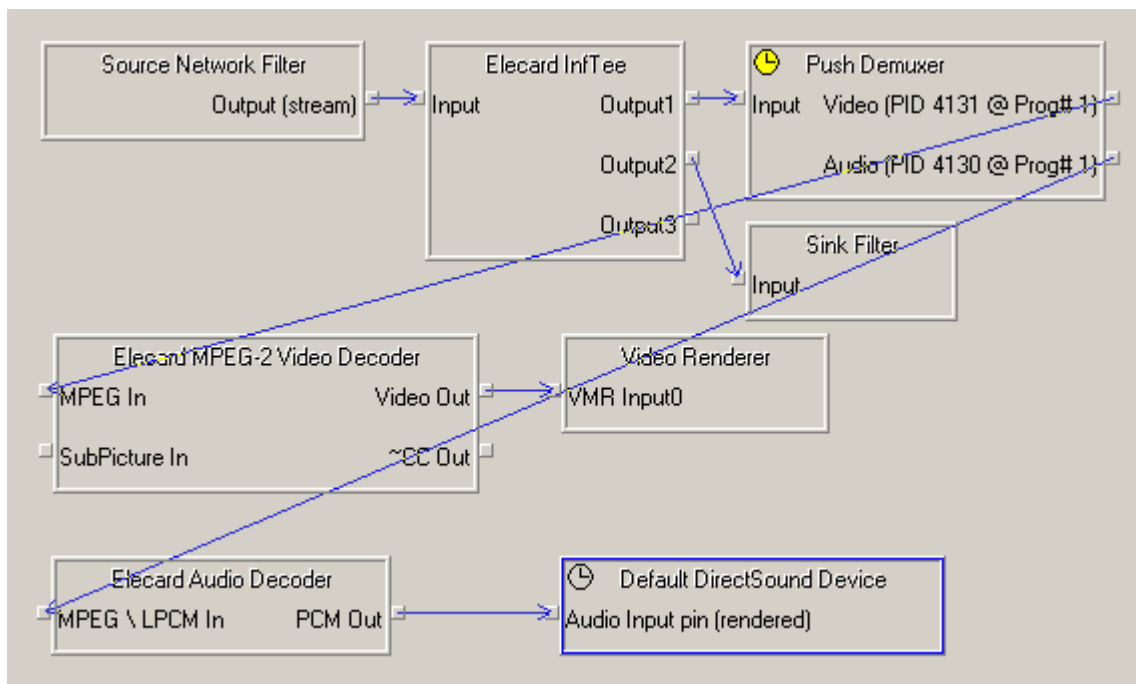
data_path – media file for playback

For example: `elecard_rtsp://127.0.0.1:554/movie.mpg`

The Elecard RTSP NetSource filter allows the stream positioning.

Whether Elecard NWSource-Plus or Elecard RTSP NetSource is selected as the source filter the graph displayed on the figure below is built, if the received stream conforms to MPEG-2 Transport or Program Stream specification. It allows both previewing and dumping the currently received stream. But positioning within a stream using the RTSP protocol is not allowed in the *Dump to file* mode.

Figure 15. Filter Graph for Broadcast Stream Receiving, Previewing and Dumping



5.4.1.2 Path

Managed C++:

Source (Elecard Codec .NET SDK)\Samples\Network\cp\NWPlayer

Binaries (Elecard Codec .NET SDK)\Binaries\NWPlayerCP.exe

C#:

Source (Elecard Codec .NET SDK)\Samples\Network\cs\NWPlayer

Binaries (Elecard Codec .NET SDK)\Binaries\NWPlayerCS.exe

Visual Basic:

Source (Elecard Codec .NET SDK)\Samples\Network\vb\NWPlayer

Binaries (Elecard Codec .NET SDK)\Binaries\NWPlayerVB.exe

5.4.1.3 Features

NWPlayer supports the following features:

- Servers announcements receiving
- Network streams playback
- Video Zoom (including drag and drop of the video being enlarged)
- Ability to dump the received stream to file (if the stream sybtype is MPEG2_TRANSPORT or MPEG2_PROGRAM)
- Stream positioning (only for RTSP protocol, if the 'Dump to file' mode is not selected)
- Playback pause, if media data is received from VOD server

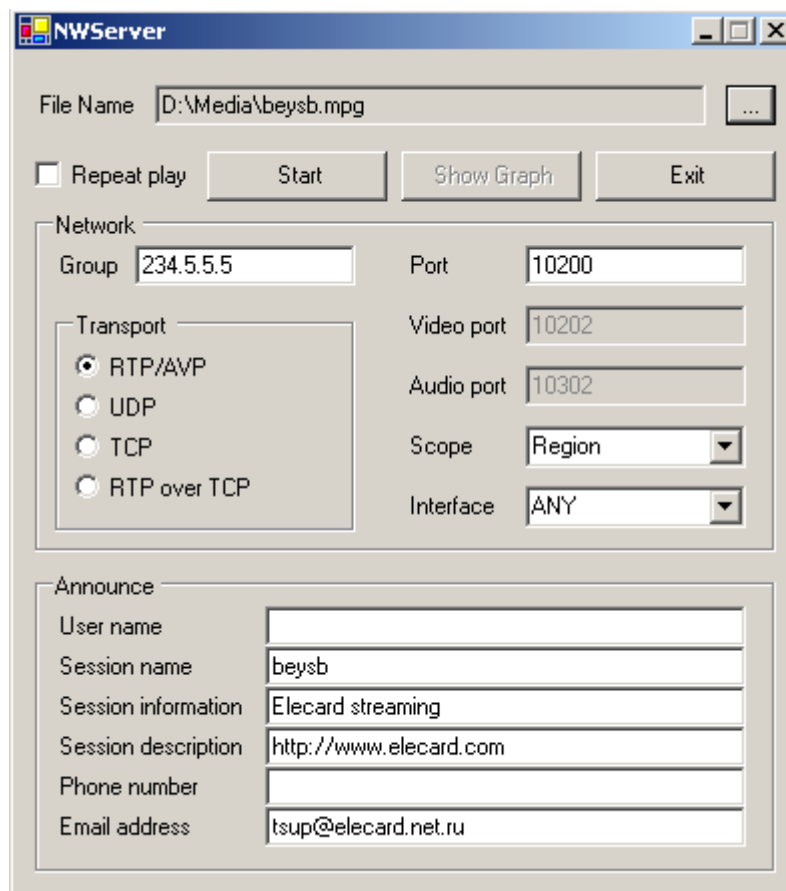
5.4.2 NWServer

NWServer is a sample application that demonstrates broadcasting media data to the network with the use of Elecard NWRenderer Filter. It is capable of sending RTP, UDP, TCP, and supports the announcement of its data session via sending SAP (SDP) packets.

5.4.2.1 Description

The NWServer sample gives a user an opportunity to choose a MPEG-2 file to broadcast and set the broadcasting settings, such as: transport type of the network packets, destination UDP port, announce information etc.

Figure 16. NWServer GUI



NWServer consists of four classes: **Form1** class (derived from the **Form** class), **StreamingServer** class (derived from the **GenericPlayer** class), **MPEG2StreamingServer** class (derived from the **StreamingServer** class) and **MP4StreamingServer** class (derived from the **StreamingServer** class).

```
public class Form1 : Form {};  
public class StreamingServer : GenericPlayer {};  
public class MPEG2StreamingServer: StreamingServer {};  
public class MP4StreamingServer : StreamingServer {};
```

Form1 is the main class. It contains instance of the **StreamingServer** class. This class is inherited by the **MPEG2StreamingServer** and **MP4StreamingServer** classes. After a new file for broadcasting is chosen one of these two classes is used for the graph building process depending on the file stream type.

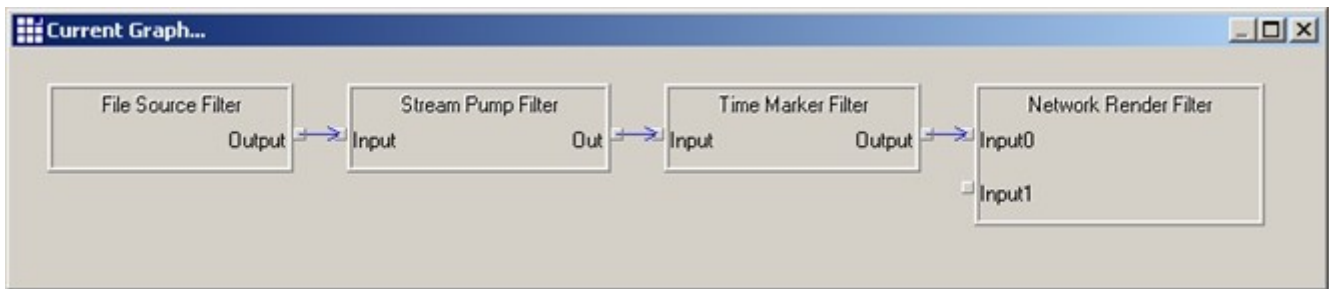
First a static method **MP4StreamingServer.CreateMP4StreamingServer(fileName)** is called, which creates a Source Filter for the chosen file and tries to connect it with the Elecard MP4 Demultiplexer. If the connection succeeds, then the stream to broadcast is of the MP4 format. In that case after the *player.InitGraph()* call the Elecard MP4 Demultiplexer is connected to the Elecard NWRenderer directly and the graph is ready to broadcast (fields containing values of the video and audio ports are enabled in the application user interface).

Figure 17. Filter Graph for MP4 File Broadcasting



If the Source Filter and MP4 Demultiplexer connection fails, then the stream to broadcast is not of the MP4 format. Then the *MPEG2StreamingServer.CreateMPEG2StreamingServer(fileName)* static method is called, which creates a Source Filter for the chosen file and tries to connect it with the Elecard Demultiplexer filter. If the connection fails then the method returns null, otherwise we conclude that the stream is of the mpeg2 format. In that case Source and Demultiplexer filters are disconnected and the Demultiplexer filter is removed from the graph. For the mpeg2 format after the *player.InitGraph()* call the following graph is built.

Figure 18. Filter Graph for MPEG-2 File Broadcasting



The NWServer sample application supports the following formats:

1. MP4 file format (ISO/IEC 14496-14)
2. MPEG-1 System Stream (ISO/IEC 11172-1)
3. MPEG-2 Program Stream (ISO/IEC 13818-1)
4. MPEG-2 Transport Stream (ISO/IEC 13818-1)

The NWServer sample application supports the following elementary streams:

1. MPEG-1 Video (ISO/IEC 11172-2)
2. MPEG-1 Audio (ISO/IEC 11172-3)
3. MPEG-2 Video (ISO/IEC 13818-2)
4. MPEG-2 Audio (ISO/IEC 13818-3)
5. Dolby Digital (ATSC A-52)

5.4.2.2 Path

Managed C++:

Source (Elecard Codec .NET SDK)\Samples\Network\cp\NWServer

Binaries (Elecard Codec .NET SDK)\Binaries\NWServerCP.exe

C#:

Source (Elecard Codec .NET SDK)\Samples\Network\cs\NWServer

Binaries (Elecard Codec .NET SDK)\Binaries\NWServerCS.exe

Visual Basic:

Source (Elecard Codec .NET SDK)\Samples\Network\vb\NWServer
Binaries (Elecard Codec .NET SDK)\Binaries\NWServerVB.exe

5.4.2.3 Features

NWServer supports the following features:

- Sending UDP, RTP, TCP streams
- Sending SAP announcements with SDP data

5.4.3 NW_ServerTranscoder

5.4.3.1 Description

NW_ServerTranscoder is a sample application that demonstrates the media data transcoding, multiplexing and broadcasting to network. Broadcasting capabilities are the same as in the NWServer application.

The NW_ServerTranscoder sample application supports the following output formats:

- Multiplexing formats – MPEG-2 TS/PS
- Video formats – MPEG-2, AVC
- Audio formats – MPEG, AAC

The following figures illustrate the NW_ServerTranscoder GUI.

Figure 19. NW_ServerTranscoder GUI - File Tab

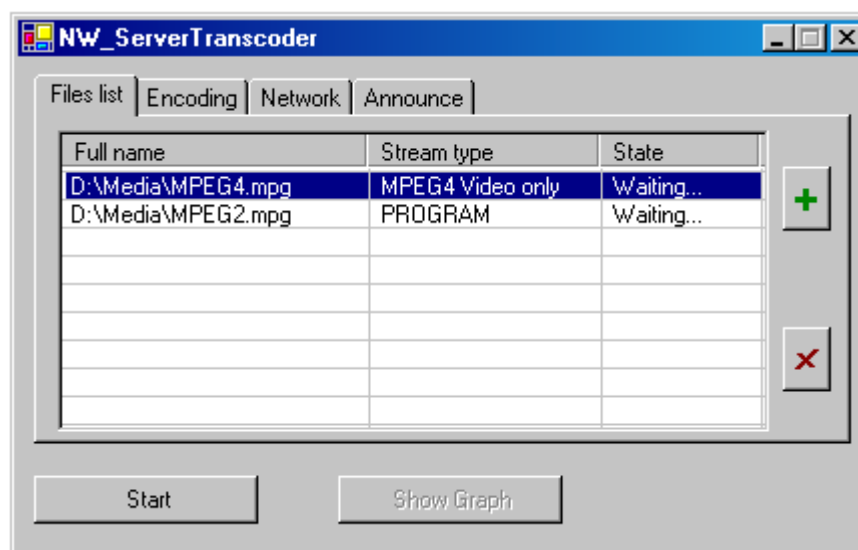


Figure 20. NW_ServerTranscoder GUI – Encoding Tab

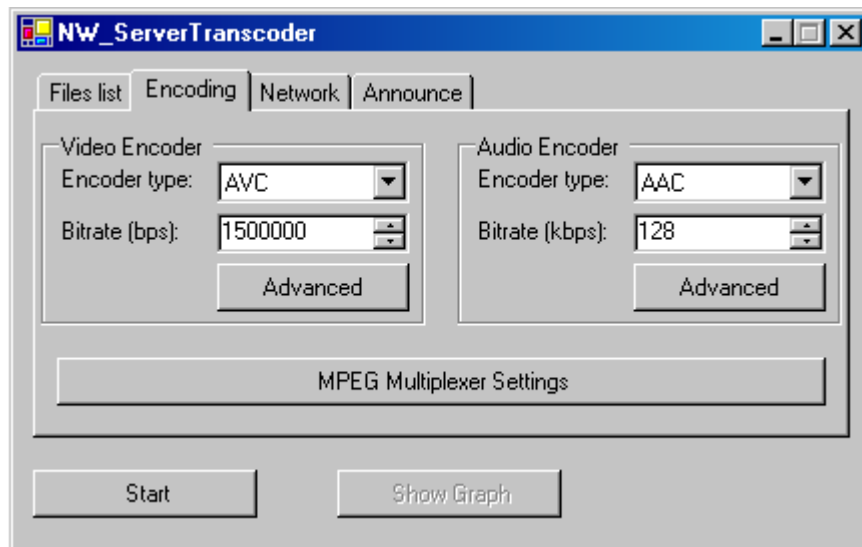


Figure 21. NW_ServerTranscoder GUI - Network Tab

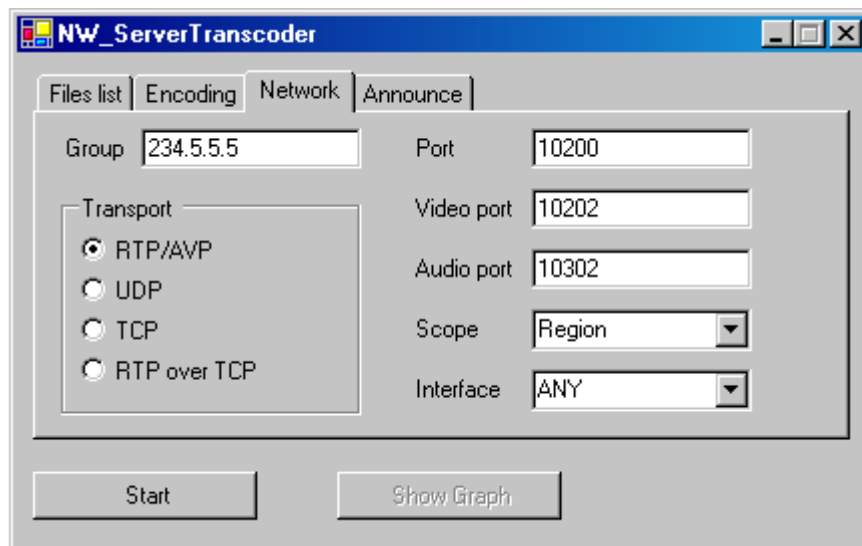


Figure 22. NW_ServerTranscoder GUI – Announce Tab

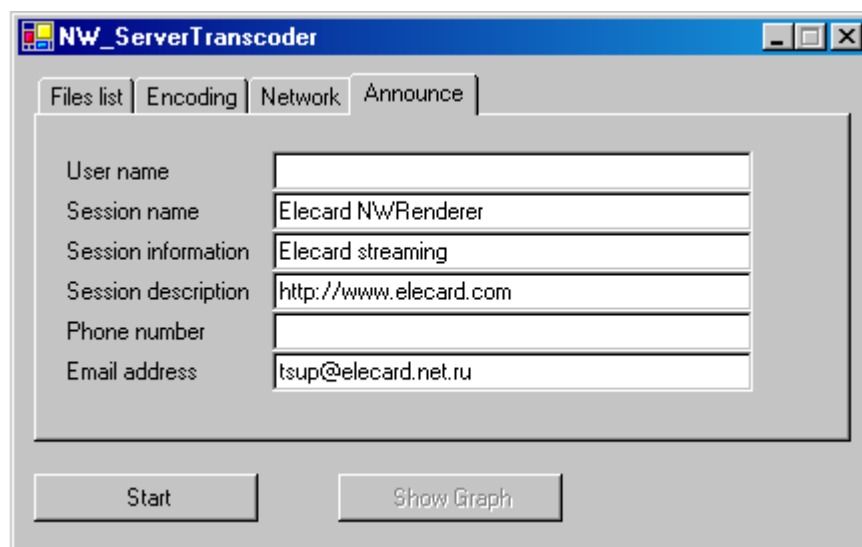
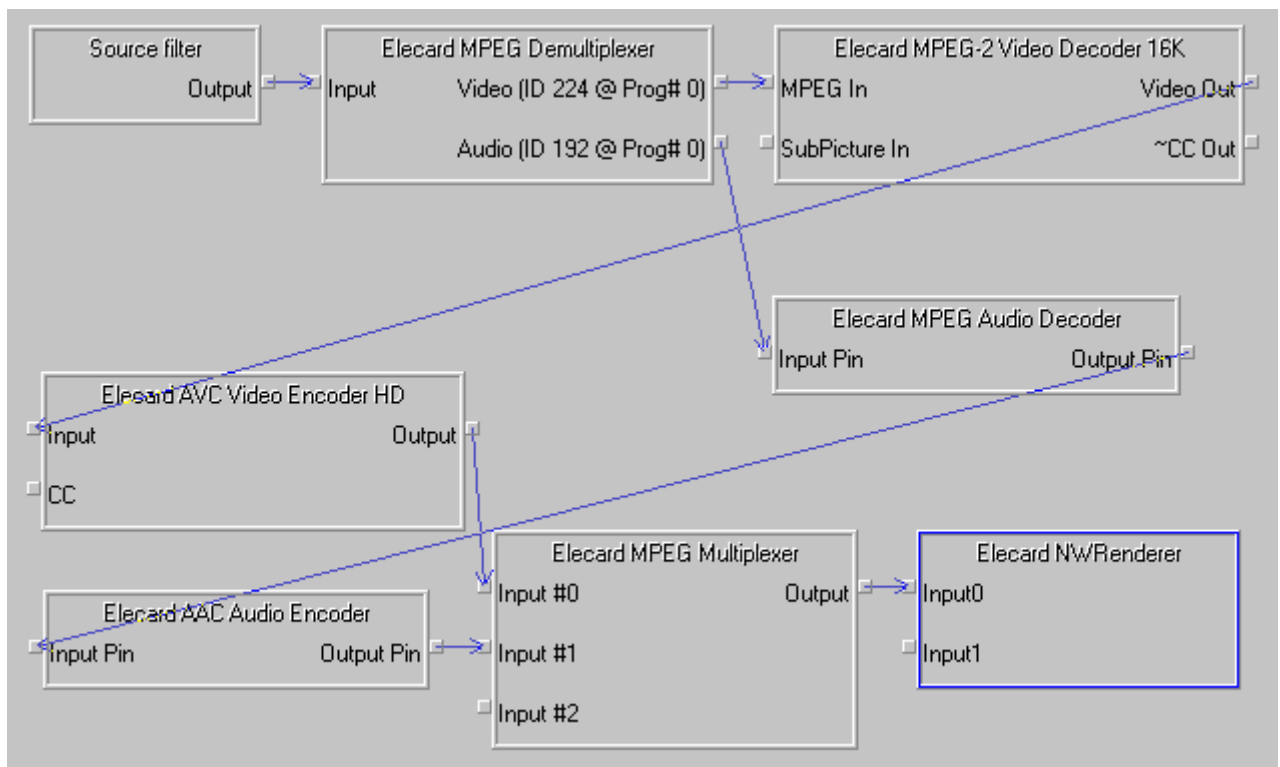


Figure 23. Filter Graph for File Transcoding and Broadcasting



5.4.3.2 Path

Managed C++:

Source (Elecard Codec .NET SDK)\Samples\Network\cp\NW_ServerTranscoder

Binaries (Elecard Codec .NET SDK)\Binaries\NW_ServerTranscoderCP.exe

C#:

Source (Elecard Codec .NET SDK)\Samples\Network\cs\NW_ServerTranscoder

Binaries (Elecard Codec .NET SDK)\Binaries\NW_ServerTranscoderCS.exe

Visual Basic:

Source (Elecard Codec .NET SDK)\Samples\Network\vb\NW_ServerTranscoder

Binaries (Elecard Codec .NET SDK)\Binaries\NW_ServerTranscoderVB.exe

5.4.3.3 Features

The NW_ServerTranscoder sample application performs the following tasks:

- Transcoding and multiplexing media streams
- Broadcasting (multicast or unicast) media streams over the Internet/Intranet
- Sending UDP, RTP and custom multicast/unicast streams
- Sending SAP announcement with SDP data