Nero Audio Plug-in Manager

v1.00

Supported by
Nero 5.5.9.0 or later
and
NeroMIX 1.3.0.0 or later!

Nero AG, Im Stoeckmaedle 18, 76307 Karlsbad, Germany

1. Contents

1. **Contents** 2
2. **License Agreement** 6
3. **Introduction** 7
   3.1. Motivation 7
   3.2. Overview 7
   3.3. Required Skills 7
   3.4. Related Topics 7
4. **Necessary Steps To Create A New Plug-in** 8
   4.1. Naming Conventions 8
   4.2. Programming The Plug-in 8
   4.3. Installing The Plug-in 8
5. **Usage Examples** 9
   5.1. Initialization And Termination 10
   5.2. Open Source File 11
   5.3. Transfer Data 12
6. **Component Structure** 13
   6.1. Functions 13
      6.1.1. NERO_PLUGIN_GetPrimaryAudioObject 13
      6.1.2. NERO_PLUGIN_ReadyToFinish 13
   6.2. Objects (audio components) 13
      6.2.1. Audio Plug-in Manager 13
      6.2.2. Component Enumerator 14
      6.2.3. Audio URL Source Factory 14
      6.2.4. Audio URL Target Factory 14
      6.2.5. URL Audio Source Object 15
      6.2.6. URL Audio Target Object 15
      6.2.7. Audio Raw Convertor Factory 16
      6.2.8. Audio Raw Convertor 16
      6.2.9. Status Object 16
      6.2.10. Information Callback Object 16
   6.3. Interfaces 17
      6.3.1. IStatus 17
      6.3.2. IAudioPluginMgr 17
      6.3.3. IAudioComponent 18
      6.3.4. IIdentifiable 19
      6.3.5. IComponentEnum 19
      6.3.6. IAudioTargetFactory 20
      6.3.7. IURLSupport 20
      6.3.8. IAudioSourceFactory 21
      6.3.9. IURLAudioSourceFactory 21
      6.3.10. IURLAudioTargetFactory 22
      6.3.11. IURLHolder 22
      6.3.12. IProcess 23
      6.3.13. IAudioItem 23
      6.3.14. IAudioSource 24
      6.3.15. IAudioRawWriter 24

6.3.16. IInfoReader 24
6.3.17. IInfoWriter 25
6.3.18. ISrcInfoCallback 26
6.3.19. IAudioRawBlockInfo 26
6.3.20. ISrcInfoViewerEditor 27
6.3.21. IConvertorFactory 28
6.3.22. IConvertor 29
6.3.23. ILanguage 30
6.3.24. IExtEnum 30
6.3.25. IProfile 31
6.3.26. ILimit 31
6.3.27. ISeekable 32
6.3.28. IAggregatable 32
6.3.29. IAudioRawReader 33
6.3.30. IControl 33
6.3.31. IEventReceiver 34
6.3.32. IConfigurable 34
6.3.33. IVendorInfo 35
6.4. Types 36
6.4.1. EAudioComponentType 36
6.4.2. EURLType 36
6.4.3. EAuxFlags 36
6.4.4. EMainScreenMode 37
6.4.5. EAppearMethod 37
6.4.6. EEvent 38
6.4.7. EWndMode 39
6.4.8. EAudioRawState 39
6.4.9. EMediaType 39
6.4.10. ELimitType 40
6.4.11. EConvFactoryMode 40
6.5. The Plug-in Utility Files 41
6.5.1. reg_SetPluginParam 41
6.5.2. reg_GetPluginParam 41
6.5.3. file_DoesExist 42
6.5.4. file_GetSize 42
6.5.5. file_GetTime 42
6.5.6. IsURLLocal 42
6.5.7. EAudioErrorMapSysErrorToAudio 42
6.6. Files For Localization 43
6.6.1. NeroPluginNls.h And NeroPluginNls.cpp 43
6.6.2. Translation File Format 43
6.6.3. Translation File Example 44
7. Error Handling 45
7.1. EAudioError 45
8. Creating A Wave Plug-in 46
8.1. Module Overview 47
8.2. A New Project 48
8.3. Adding Include Directories 49

8.4. Linking with the Windows Multimedia Library 49
8.5. Adding An NLS Resource 50
8.6. Wave Format Definitions And Structs 52
The Wave Target Settings Dialog 53
8.8. The File Details Dialog 55
8.9. Adding The Wave Target Settings Dialog Class 56
8.10. Adding The File Details Dialog Class 57
8.11. The Wave Target Dialog Class 57
8.12. The Wave Source Class 58
8.13. The Wave Target Class 58
8.14. The Wave Source Factory Class 59
8.14.1. WavSrcFactory.h 59
WavSrcFactory.cpp 61
8.15. The Wave Target Factory Class 65
8.15.1. WavTgtFactory.h 65
8.15.2. WavTgtFactory.cpp 66
8.16. The Component Enumerator Class 70
8.16.1. WavEnum.h 70
8.16.2. WavEnum.cpp 71
8.17. Implementation of exported functions 75
8.17.1. Global Definitions 75
8.17.2. NERO_PLUGIN_GetPrimaryAudioObject 75
8.17.3. NERO_PLUGIN_ReadyToFinish 76
8.18. Additional Includes In stdafx.h 76
8.19. Defining Exported Functions 76
Files In The Project 77
8.21. Building The Project 77
8.22. Copying And Installing The DLL 77
9. Bibliography 78
9.1. C++ Programming Books 78
9.2. C++ Online Resources 78
9.3. COM books 78
9.4. General CD/CD-ROM Online Resources 78
9.5. Audio CD Online Resources 78
10. Listings 79
10.1. dlg_WavSettings.h 79
10.2. dlg_WavSettings.cpp 80
10.3. dlg_WavSrcInfo.h 84
10.4. dlg_WavSrcInfo.cpp 86
10.5. FileHelper.h 91
10.6. FileHelper.cpp 92
10.7. myWav.h 94
10.8. myWav.cpp 95
10.9. WavEnum.h 97
10.10. WavEnum.cpp 98
10.11. WavSrc.h 101
10.12. WavSrc.cpp 104
10.13. WavSrcFactory.h 114

10.14. WavSrcFactory.cpp 116
10.15. WavTgt.h 119
10.16. WavTgt.cpp 121
10.17. WavTgtFactory.h 127
10.18. WavTgtFactory.cpp 129

2. License Agreement

IMPORTANT: PLEASE READ THE SOFTWARE LICENSE AGREEMENT ("LICENSE") CAREFULLY BEFORE USING THE SOFTWARE.

USING THE SOFTWARE INDICATES YOUR ACKNOWLEDGMENT THAT YOU HAVE READ THE LICENSE AND AGREE TO ITS TERMS.

The license agreement is contained in a text file, “NeroSDK_License.txt”, to be found in the root folder of the installation package.

3. Introduction

3.1. Motivation

New audio formats appear all the time - and *Nero* helps you keep up with them. ***Nero 5.5.9.0's*** **Audio Plug-in Interface** makes it easy to create your own custom-written support files (DLLs) for the new formats of your choice.

The Audio Plug-in Interface will not work with older versions of *Nero*!
It requires at least *Nero 5.5.9.0* or *NeroMIX 1.3.0.0*.

3.2. Overview

A Plug-in Architecture makes it possible to add new features to a software without having to the change the software itself.

Nero and *NeroMIX* offer a plug-in interface that enables software vendors or individual programmers to extend the existing capabilities. Audio Plug-ins can be used to support the processing (reading, writing, conversion) of new audio formats as they emerge, or handle existing formats in a different way, e.g. by offering enhanced conversion features or audio effects.

The architecture of the audio plug-in environment is presented by audio component and media item objects. Each of them exposes Microsoft COM interfaces and is responsible for performing certain tasks. Audio components are implemented in form of DLLs. Those DLLs must expose a function GetPrimaryAudioObject which returns a pointer to a component. Audio plug-in DLLs can contain one or more audio components. If the plug-in contains more than one component GetPrimaryAudioObject must first return a plug-in enumerator object which enumerates others.

This paper, the documentation of the *Nero Audio Plug-in Interface*, contains some practical guidelines on how to write and install plug-ins.

3.3. Required Skills

This documentation is directed towards Software developers who have gathered some experience in programming C++ together with Microsoft COM (Component Object Model).

It is absolutely required that you know the basic concepts of the C++ programming language and Microsoft COM to use the *Nero Audio Plug-in Interface*.

3.4. Related Topics

Please look up additional information concerning *Nero* and the *Nero Audio Plug-in Interface* at http://www.nero.com.

4. Necessary Steps To Create A New Plug-in

4.1. Naming Conventions

The names of custom Plug-ins are expected to be prefixed by “nx”. For example, a Plug-in that processes Wave data could be named “**nx**MyWav.dll”.
If the “nx” prefix is missing, the Audio Plug-in Manager will not load the Plug-in.

4.2. Programming The Plug-in

The steps listed below are sufficient to create a plug-in, which is capable of reading and writing Wave data.
However, it is not required to support both reading and writing. Some plug-ins may only support one type of operation, while others may support multiple operations (reading, writing, conversion) for a number of different formats.

1. Create the target and target factory (e.g. CWavTgt and CWavTgtFactory classes for audio data in wave format).
2. Create the source and source factory (e.g. CWavSrc and CWavSrcFactory classes for audio data in wave format).
3. Create the factory enumerator (CWavEnum class), which exposes the source and target factories.
4. All targets, sources and their factories must be derived from CAggregatable class, which allows the objects to be aggregated and their behavior to be extended/changed externally.
5. NLS (national language support) is facilitated by the CTranslator class.
   a) Prepare the file with translations and add it under “NLSDATA” branch to resources.
   b) Create an object of CTranslator class and pass the ID of translation file in resources to it’s constructor.
   c) Call its TranslateWindow function.

We have included a detailed description and source code for the creation of the Wave reader/writer plug-in in this document (*chapter 6. 8. Creating A Wave* Plug-in).

4.3. Installing The Plug-in

Copy your plug-in-DLL to the *ahead\Shared\AudioPlugins* directory.

5. Usage Examples

This section describes the order of calls that *Nero* will perform when using an Audio Plug-in.

We will describe how the Plug-in will be initialized and terminated, what happens when the user selects a source file for reading and how data is transferred from a source to a target.

The diagrams on the next pages are related to the example implementation of a wave source/target plug-in. This implementation is available in source code, and its creation will be explained in detail later.

The following table shows an overview of the instances that were used in the message sequence charts.

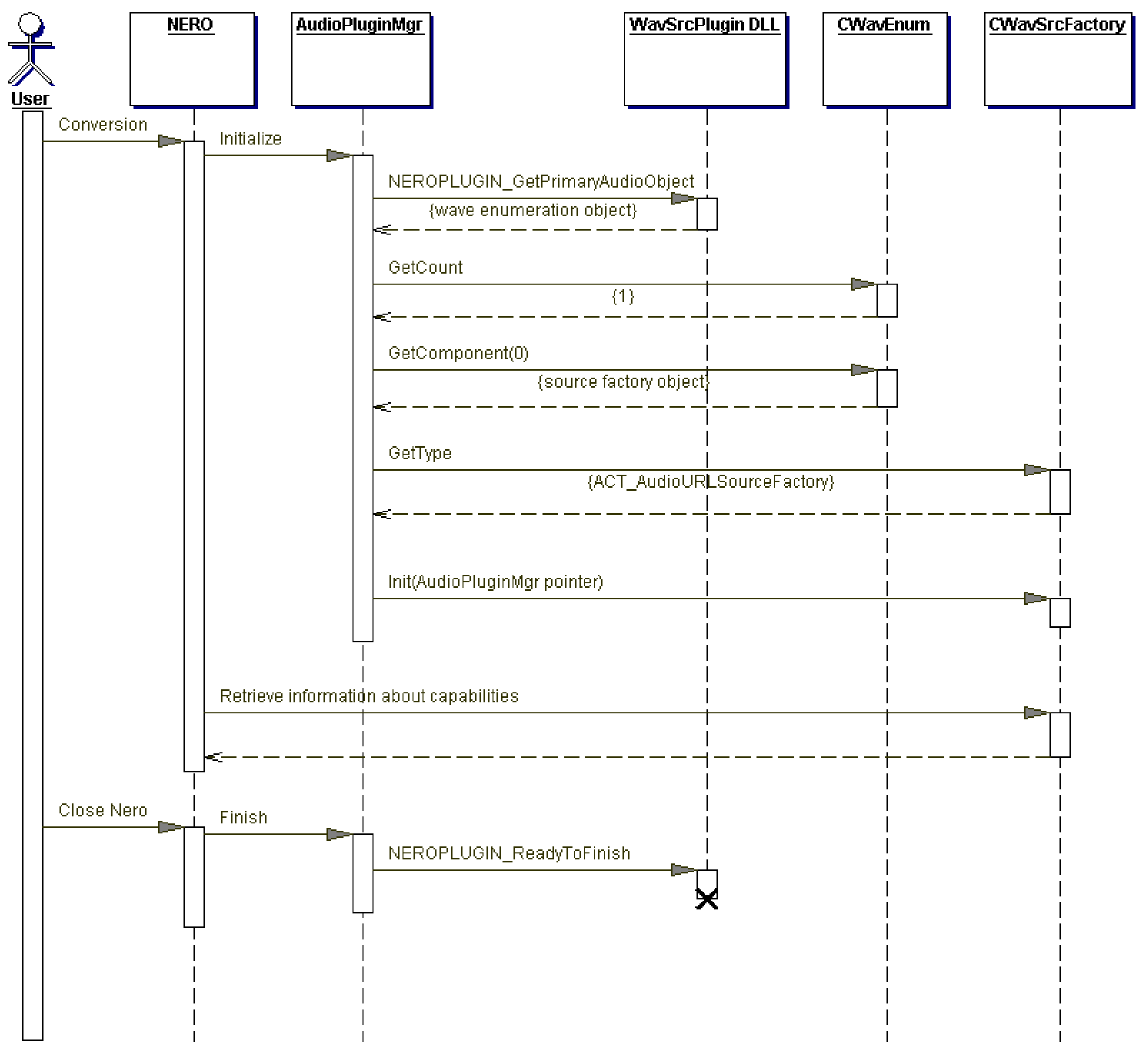
Instance Name	**Description**
User	The user who works with *Nero*.
NERO	The *Nero* application
Audio Plugin Manager	The Audio Plug-in Manager which will is controlled by *Nero*.
WavSrcPluginDLL	A plug-in DLL, which exposes the NEROPLUGIN_GetPrimaryAudioObject and NEROPLUGIN_ReadToFinish functions.
CWavEnum	The wave enumeration object of the plug-in.
CWavSrcFactory	The wave source factory object of the plug-in. The source factory creates wave sources.
CWavSrc	A wave source object, which is capable of reading wave data.
CWavTgt	A wave target object, which is capable of writing wave data.

5.1. Initialization And Termination

Here we will explain how a plug-in and its objects are initialized and terminated by the Plug-in Manager.

When the user selects *Nero's* "conversion" menu entry, a request is sent to the Audio Plug-in Manger. The Manager queries all available plug-ins, one of them being the Wave Source plug-in. Through a call to NEROPLUGIN_GetPrimaryAudioObject(), a pointer to a CWavEnum class object is retrieved. The Manager then requests information about the number of implemented items. One by one those items are retrieved and their type is determined. Pointers to the objects are forwarded to *Nero*. If the items are Factories, *Nero* requests information about the number of supported extensions and permitted operations.

When the user closes the *Nero* application, the plug-in DLL is unloaded after a call to NEROPLUGIN_ReadyToFinish().

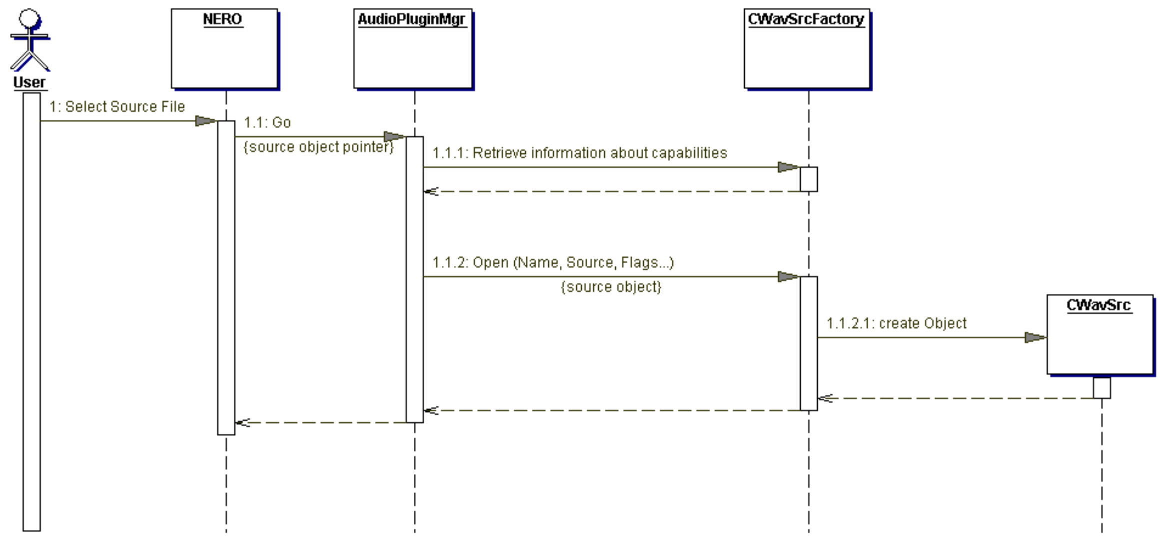


5.2. Open Source File

This example shows how a source item is created by a Source Factory.

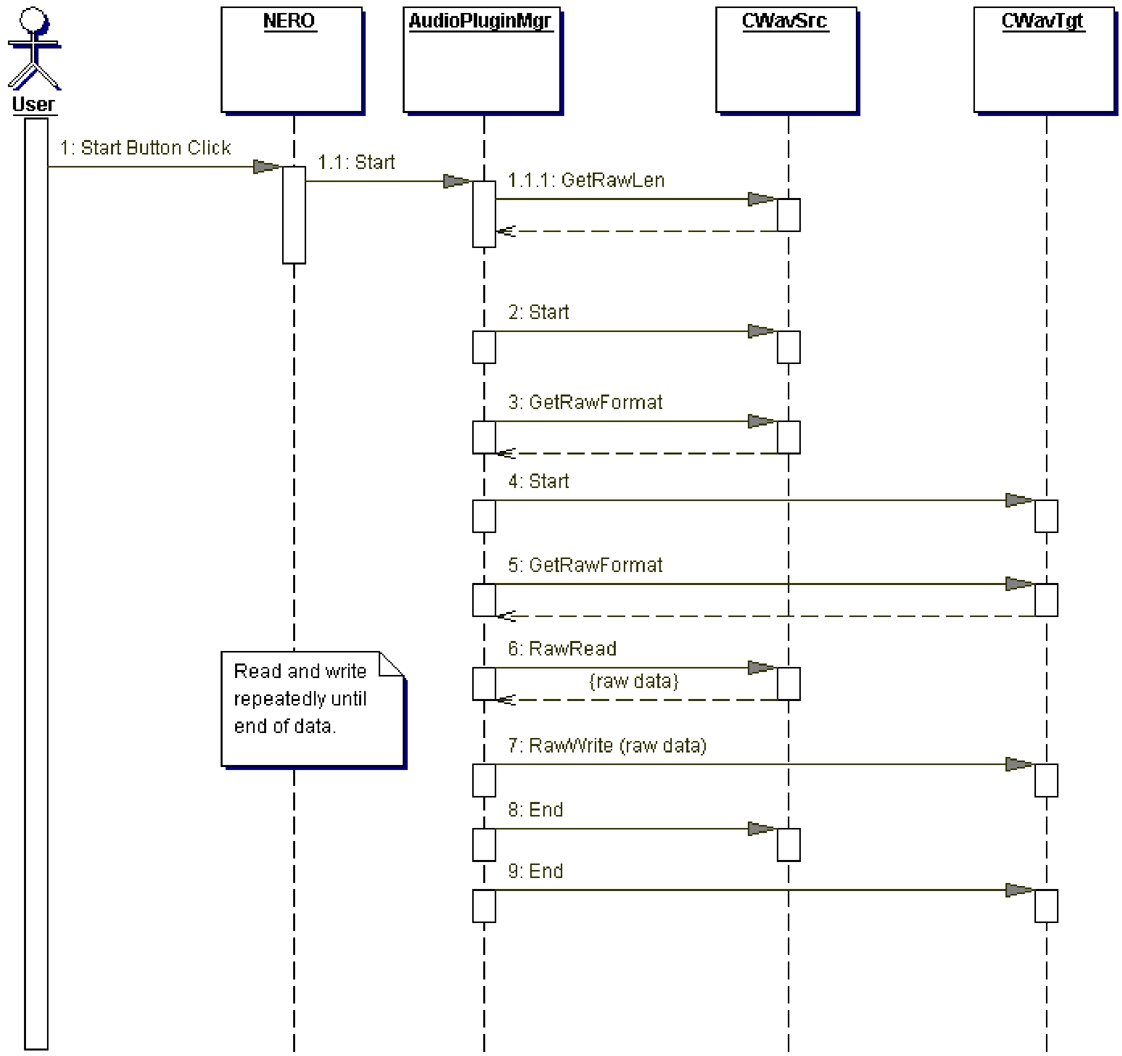
When the user selects a source file, which is supported by the plug-in, *Nero* will create a request to the Plug-in Manager. After gathering information about the capabilities from the Source Factory, the Plug-in Manager calls the Factory's Open() method, and the Factory creates an object of the CWavSource class. This object represents one audio file.

A pointer to this this object is then forwarded to Nero by the Audio Plug-in Manager.



5.3. Transfer Data

In this diagram source and target items are used to read data from a source in one format and write it to a target in a different format.



6. Component Structure

6.1. Functions

6.1.1. NERO_PLUGIN_GetPrimaryAudioObject

```
bool NERO_PLUGIN_GetPrimaryAudioObject(IAudioComponent **pAC)
```

GetPrimaryAudioObject returns a pointer to a component. Audio plug-in DLLs can contain one or more audio components. If the plug-in contains more than one component GetPrimaryAudioObject must first return a plug-in enumerator object which enumerates others.

6.1.2. NERO_PLUGIN_ReadyToFinish

```
bool NERO_PLUGIN_ReadyToFinish()
```

This function is called when the application terminates. It shall return true if the plug-in can be unloaded or false if interfaces of the plug-in are still being used.

6.2. Objects (audio components)

6.2.1. Audio Plug-in Manager

The Audio Plug-in Manager represents the top level node of the plug-in enviroment hierarchy. The application must initially refer to this object. The Audio Plug-in Manager gathers all the plug-in DLLs and initializes them. After initialization all the audio components receive a pointer to this object. Thus they can query interfaces from it and use appropriate profiles, languages and so on.

Interfaces:

- IAudioPluginMgr (main)
- IComponentEnum
- ILanguage
- IProfile

6.2.2. Component Enumerator

Enumerates other audio components. If a plug-in DLL contains more than one component it must expose an object of this type first which will enumerate others. Audio Plug-in Manager also is a component enumerator.

Interfaces:

- IComponentEnum
- IIdentifiable
- IAudioComponent

6.2.3. Audio URL Source Factory

Opens audio files by URLs and creates URL audio source objects. When necessary it creates wrapper- aggregator objects on source items in order to perform conversions.

Interfaces:

- IIdentifiable
- IAudioComponent
- IURLAudioSourceFactory
- IExtEnum
- IUsageLimit

6.2.4. Audio URL Target Factory

Creates audio files at URLs and creates URL audio target objects. When necessary it creates wrapper- agregater objects on target items in order to perform conversions.

Interfaces:

- IIdentifiable
- IAudioComponent
- IURLAudioTargetFactory
- IExtEnum
- IUsageLimit

6.2.5. URL Audio Source Object

Represents an audio file opened from an URL. Allows to read RAW data from it, obtain other information such as artist, title names, in certain cases play the file or perform other operations.

Interfaces:

- IURLHolder
- IAudioItem
- IProcess
- IAudioSource
- IAudioRawReader
- ISeekable
- IAudioRawBlockInfo
- IAggregatable
- ISrcInfoViewerEditor
- IInfoReader
- IInfoWriter

6.2.6. URL Audio Target Object

Represents an audio file created at an URL.

Interfaces:

- IURLHolder
- IAudioItem
- IProcess
- IAudioRawWriter
- IAggregatable
- IInfoWriter

6.2.7. Audio Raw Convertor Factory

The raw convertor factory creates audio raw convertor components. Any audio component can access this object in order to create a RAW convertor for its purposes. If many convertor factories are available, the application must provide a way for users to select one of them. After that the application must call SetConvertorFactory from IAudioPluginMgr.

Interfaces:

- IConvertorFactory
- IIdentifiable
- IAudioComponent

6.2.8. Audio Raw Convertor

Converts audio RAW data from one format to another, e.g. by adapting frequency and bit resolution.

Interfaces:

- IConvertor

6.2.9. Status Object

Contains information about operation and completion status. Allows to obtain the completion status code, message in text form, pointer to the object it's issued by an other information.

Interfaces:

- IStatus

6.2.10. Information Callback Object

If the application wants to view/edit a source item's additional information through ISrcInfoViewerEditor interface, it can also interfere in the editing process. To do so it has to supply an object of this type and pass the IInfoCallback interface pointer to the item.

6.3. Interfaces

6.3.1. IStatus

Pointers of this type are passed to some functions in audio items, factories and others. If an error occurs those functions create objects exposing this interface and the application can obtain detailed information about the error.

```
interface IStatus : public IUnknown
{
	bool			IsSuccess();
	bool			IsError();
	const char *		GetText();
	DWORD			GetErrorCode();
};
```

Exposed by

- Status object

Function Name	Description
IsSuccess	Operation successful.
IsError	An error occurred.
GetText	This can return NULL if no text is supplied for the current status.
GetErrorCode	Retrieves the error code if an error occurred.

6.3.2. IAudioPluginMgr

The general interface of the central plug-in manager. All the components in the system receive its pointer during initialization and can refer to it to obtain - for example - the profile information, language settings, and more. Also applications work with this interface to open and create source/target audio items.

```
interface IAudioPluginMgr : public IUnknown
{
	bool		SetConvertorFactory(IConvertorFactory *pCF, IStatus **ppStatus);
	bool		GetConvertorFactory(IConvertorFactory **ppCF);
	bool		GetUnsupportedComponents(IComponentEnum **ppEnum);
	bool		SendEvent(EEvent event, IUnknown *pSource);
	bool		SubscribeToEvents(IEventReceiver *pReceiver);
	bool		UnsubscribeToEvents(IEventReceiver *pReceiver);

	bool		OpenURLAudioFile(	const char		*szURL,
						const SWavFormat	*pFormat,
						IUnknown		**ppSrc,
						EAuxFlags		flagsInclude,
						EAuxFlags		flagsExclude,
```

```
                                             IStatus           **ppStatus;

       bool           CreateURLAudioTarget(        URLAudioTargetFactory  *pFactory,
                                                   IUnknown                **pTgt,
                                                   const SWavFormat        &formatSrc,
                                                   IStatus                 **ppStatus);
};
```

Exposed by

- Audio Plug-in Manager

Function Name	Description
SetConvertorFactory	Specify a default Convertor Factory.
GetConvertorFactory	Retrieve a pointer to an IConvertorFactory interface.
GetUnsupportedComponents	Retrieve a component enumerator for unsupported components.
SendEvent	Sends an event to all the subscribed event receivers, they can be either application-side objects, plug-ins, plug-in manager itself or other objects.
SubscribeToEvents	Initially none of audio components, items or application parts are subscribed to events. If an object needs to handle events, it has to subscribe to this method. The SendEvent method of IAudioPluginManager only sends events to subscribed objects.
UnsubscribeToEvents	If the events are not anymore expected, objects can unsubscribe them.
OpenURLAudioFile	Opens an audio file from an URL and, if the source quality is not the same as "pFormat", creates a wrapper object which aggregates the source. pFormat can be NULL, in this case the source will expose the original format.
CreateURLAudioTarget	Works in analogy to OpenURLAudioFile, but here a file is opened for writing.

6.3.3. IAudioComponent

This interface is exposed by all the audio plug-ins, i.e. source and target audio factories, audio RAW convertor factories, etc.

```
interface IAudioComponent : public IUnknown
{
       const char *             GetName();
       EAudioComponentType      GetType();
       bool                     Init(IAudioPluginMgr *pMgr, IStatus **ppStatus);
       bool                     Done();
};
```

Exposed by

- any audio component

Function Name	Description
GetName	Returns the name of the component, e.g. “PCM wav file”
GetType	Application should first call this method to determine the type of the component and after that query the appropriate interface.
Init	This method is called after all the components are enumerated. It should perform the actual initialization and store pMgr for later use to access other components or the plug-in manager itself.
Done	

6.3.4. IIdentifiable

This interface **must be exposed** by **every** factory or enumerator component in the plug-in!

```
interface IIdentifiable : public IUnknown
{
	virtual void		GetID(GUID *pGUID);
};
```

Exposed by

- IAudioSourceFactory
- IAudioTargetFactory
- IConvertorFactory
- IComponentEnum

Function Name	Description
GetID	Retrieves the GUID of the component.

6.3.5. IComponentEnum

Enumerates audio plug-in components. Exposed by the central plug-in manager and by all the audio plug-in DLLs that need to incapsulate more than one component.

```
interface IComponentEnum : public IUnknown
{
	int			GetCount();
	bool			GetComponent(int iNum, IAudioComponent **pComp);
};
```

Exposed by

- Component enumerator object
- Audio Plug-in Manager

Function Name	Description
GetCount	Retrieves the number of components.
GetComponent	Retrieves a component by index number.

6.3.6. IAudioTargetFactory

```
interface IAudioTargetFactory
{
	bool			EditSettings(IUnknown **ppTgt, int iCount);
	bool			CanEditSettings(IUnknown **ppTgt, int iCount);
};
```

Base class for

- IURLAudioTargetFactory
- other target factories

Function Name	Description
EditSettings	Returns true if settings are changed, otherwise returns false. If ppTgt and iCount are equal to 0 plug-in will edit global settings.
CanEditSettings	Application should first call this function to determine if this factory can edit settings for these items.

6.3.7. IURLSupport

Creates target audio items at URLs.

```
interface IURLSupport : public IUnknown
{
	EURLType	GetSupportedURLTypes();
};
```

Exposed by

- all the objects which need URL type support.

Base class for

- IURLAudioSourceFactory
- IURLAudioTargetFactory

Function Name	Description
GetSupportedURLTypes	Return the type of URLs that this object supports (e.g. local files).

6.3.8. IAudioSourceFactory

```
interface IAudioSourceFactory
{
       virtual EAuxFlags     GetAuxFlags()
};
```

Base class for

- IURLAudioSourceFactory

Function Name	Description
GetAuxFlags	This function is used to return the source specific flags. GetAuxFlags exists both in source factories and source objects. In some cases the return values can vary. For example, in general files in the WMA format are accepted to be burned on CSs, but if the file is protected by digital rights management and is not allowed to be burned on CD, the respective flag can be disabled. If a particular prohibited action is requested, the Plug-in Manager will report that the file could not be opened.

6.3.9. IURLAudioSourceFactory

Opens source audio items by URLs.

```
interface IURLAudioSourceFactory : public IAudioSourceFactory  IURLSupport
{
       bool        Open(     const char        *szURL,
                             IUnknown          **ppSrc,
                             EAuxFlags         flagsInclude,
                             EAuxFlags         flagsExclude,
                             IStatus           **ppStatus);
};
```

Exposed by

- Audio URL source factory

Function Name	Description
Open	When the application opens an audio file it normally knows what this item is being opened for: playing, reading, etc. The plug-in which opens the item must be able to operate according to the flagsInclude and flagsExclude parameters, i.e. all the flags specified by flagsInclude must be supported and all the flags specified by flagsExclude must not be supported.

6.3.10. IURLAudioTargetFactory

```
interface IURLAudioTargetFactory : public IAudioTargetFactory, IURLSupport
{
	bool   Create(IUnknown **ppTgt, const SWavFormat  &formatSrc, IStatus, **ppStatus);
};
```

Exposed by

- Audio URL target factory

Function Name	Description
Create	Create a target item.

6.3.11. IURLHolder

Allows the owner object to store, change, retrieve and determine the type of URL.

```
interface IURLHolder : public IUnknown
{
	void                    SetURL(const char *szURL);
	const char *            GetURL();
	EURLType                GetType();
};
```

Exposed by

- source and target items

(This interface is generic, and in the future might be exposed by other objects besides the ones mentioned.)

Function Name	Description
SetURL	Sets a new URL.
GetURL	Retrieves the current URL.
GetType	Retrieves the type of URL, e.g. "local file".

6.3.12. IProcess

IProcess is a generic interface for process control.

```
interface IProcess : public IUnknown
{
	bool  Start(IStatus **ppStatus);
	bool  End(IStatus **ppStatus);
	bool  IsInProcess();
};
```

Function Name	Description
Start	Starts the execution of a process.
End	Ends the execution of a process.
IsInProcess	Return true if process is currently being executed.

6.3.13. IAudioItem

```
interface IAudioItem : public IUnknown
{
	bool			GetCreator(IAudioComponent **ppCreator);
	EAuxFlags		GetAuxFlags();
	void			SetAuxFlags(EAuxFlags flags);
};
```

Function Name	Description
GetCreator	Retrieves a pointer to the object that created the item.
GetAuxFlags	Bitmask of auxiliary flags. GetAuxFlags is normally called from source items.
SetAuxFlags	SetAuxFlags is normally called from target items.

6.3.14. IAudioSource

Provides general information about audio source items.

```
interface IAudioSource : public IUnknown
{
       const char *             GetInfo();
       ULONGLONG                GetDuration();
};
```

Function Name	Description
GetInfo	Returns free-form text string about the item.
GetDuration	Returns the file duration in milliseconds.

6.3.15. IAudioRawWriter

Allows to write audio RAW data into an object.

```
interface IAudioRawWriter : public IUnknown
{
       bool              RawWrite(   BYTE *pData,
                                     int iNumberOfBytesToWrite,
                                     EAudioRawState &state,
                                     IStatus **ppStatus);
       SWavFormat        GetRawFormat();
};
```

Function Name	Description
RawWrite	Write a number of bytes from a buffer.
GetRawFormat	Since the object is created, this function can return different values depending on output target settings. These cannot be changed during the writing process so the application/aggregator calls this function immediately before the process starts.

6.3.16. IInfoReader

All the methods of IInfoReader retrieve the data in ASCII and Multibyte zero terminated strings. No specific format is defined for the content.

```
interface IInfoReader : public IUnknown
{
       const char *             GetTitle();
       const char *             GetArtist();
       const char *             GetAlbum();
       const char *             GetYear();
       const char *             GetGenre();
};
```

Exposed by

- URL audio source object

Function Name	Description
GetTitle	Can return NULL for no data indication
GetArtist	Can return NULL for no data indication
GetAlbum	Can return NULL for no data indication
GetYear	Can return NULL for no data indication
GetGenre	Can return NULL for no data indication

6.3.17. IInfoWriter

All the methods of IInfoWriter receive the data in ASCII and Multibyte zero terminated strings. No specific format is defined for the content.

```
interface IInfoWriter : public IUnknown
{
	bool			SetTitle(const char *szTitle);
	bool			SetArtist(const char *szArtist);
	bool			SetAlbum(const char *szAlbum);
	bool			SetYear(const char *szYear);
	bool			SetGenre(const char *szGenre);
};
```

Exposed by

- URL audio source object

Function Name	Description
SetTitle	Set title information.
SetArtist	Set artist information.
SetAlbum	Set album information.
SetYear	Set information about the year of recording.
SetGenre	Set information about the genre.

6.3.18. ISrcInfoCallback

An application can create custom buttons in the plug-in source information dialogs. Handling of those buttons is accordingly implemented by the application. In order to do so, a source item supporting this functionality should expose the ISrcInfoViewerEditor interface which will give the application access to this data, so it can be changed. The plug-in calls the application's GetCustomButtonCount to get the number of those buttons and then calls GetControl to retrieve the individual buttons.

```
interface ISrcInfoCallback : public IUnknown
{
	int			GetCustomButtonCount();
	bool			GetControl(int iNum, IControl **ppControl);
	void			OnModified(bool b);
	void			OnCustomButton(int iID);
	bool			OnSave();
	bool			OnClose();
};
```

Exposed by

- Information callback object

Function Name	Description
GetCustomButtonCount	Return the number of custom buttons.
GetControl	Return an individual button.
OnModified	Called when some information has been changed by the user.
OnCustomButton	Called if a custom button is pressed.
OnSave	Called before the info dialog performs the actual saving. If it returns true, the saving will be performed, otherwise not.
OnClose	Called before the info dialog closes. If returns true, the dialog will be closed, otherwise not.

6.3.19. IAudioRawBlockInfo

Provides information about an audio source in blocks (current position, size of block, size of the source in blocks).

```
interface IAudioRawBlockInfo : public IUnknown
{
	ULONGLONG		GetPos();
	ULONGLONG		GetBlockSize();
	virtual ULONGLONG	GetDataLength();
};
```

Function Name	Description
GetPos	Retrieve current position.
GetBlockSize	Retrieve block size.
GetDataLength	If the audio source is infinite like in Internet streams, the return value will be NO_LENGTH. Otherwise this method returns the length of RAW data in blocks.

6.3.20. ISrcInfoViewerEditor

This interface provides the application with a flexible way to display the properties of opened media files. The application can add self-handled buttons to a dialog, which is actually implemented on the plug-in side.

The ISrcInfoViewerEditor interface is implemented by media source objects. If is not implemented, the application should just disable the appropriate menu/toolbar or any other control items.

When the application wants to display the information, it queries the source object for this interface and passes a pointer to the ISrcInfoCallback interface of the callback object implemented on the application side. The callback object actually feeds the necessary information about the custom buttons to the plug-in - this is optional and the application can omit this.

GetCustomControlCount returns the number of custom controls that the application intends to add to the source information dialog - currently only buttons are supported. Then the plug-in calls the GetControl method from the callback interface as many times as custom buttons are to be added. Each time the application should return an IControl interface of the object implemented on the application side which provides some details on the control such as title, ID and window class name (which should normally be "BUTTON"). The ID will be passed to the OnCustomButton method of the callback interface when a button is pressed.

In some cases it is possible to change the artist or title information on the fly - for example in mp3 files where the ID3 tag is just added to the end of the file and it is possible to rewrite it without restructuring the file. In such cases edit boxes let the user enter new values.

The application can then call the SaveDialogToObject and SaveObjectToFile methods from the "Save" button handler. Previously the application must ascertain through CanSaveObjectToFile that this action is permitted - most formats do not allow this. If saving is successful the application receives the OnSave notification.

```
interface ISrcInfoViewerEditor : public IUnknown
{
	bool		GetCallback(ISrcInfoCallback **ppCB);
	void		SetCallback(ISrcInfoCallback *pNewCallback);
	bool		DoModal(IStatus **ppStatus);
	bool		CloseModal(IStatus **ppStatus);
	bool		SaveDialogToObject(IStatus **ppStatus);
	bool		SaveObjectToFile(IStatus **ppStatus);
	bool		CanSaveObjectToFile();
};
```

Exposed by

- URL audio source object
- URL audio target object

Function Name	Description
GetCallback	Retrieve the callback pointer.
SetCallback	Set the callback pointer.
DoModal	Open the Info dialog.
CloseModal	Close the Info dialog.
SaveDialogToObject	Save the dialog to an object.
SaveObjectToFile	Save the object to a file.
CanSaveObjectToFile	Returns true if saving to a file is allowed.

6.3.21. IConvertorFactory

Creates instances of certain type raw convertor objects.

```
interface IConvertorFactory : public IUnknown
{
	bool		IsConfigurable();
	bool		Configure();
	EMediaType	GetSupportedMediaTypes();
	bool		CreateConvertor(	SWavFormat &src,
						SWavFormat &tgt,
						IConvertor **ppConv,
						IStatus **ppStatus);
};
```

Exposed by

- Audio raw convertor factory

Function Name	Description
IsConfigurable	Returns true if the conversion is configurable, otherwise returns false.
Configure	Allows the convertor to configure itself, normally it pops up a dialog which allows the user to select conversion methods, options, etc. If it returns true, it means that the configuration is changed, otherwise the return value will be false.
GetSupportedMediaTypes	None, Basic Audio or Extended Audio.
CreateConvertor	Creates an instance of a convertor object and returns its IConvertor interface.

6.3.22. IConvertor

```
interface IConvertor : public IUnknown
{
	bool		Convert(BYTE *pData, int iNumberOfBytes, IStatus **ppStatus);
	int		GetOutputBufSize();
	BYTE *	GetOutputBuf();
};
```

Exposed by

- Audio raw convertor

Function Name	Description
Convert	pData points to the buffer containing the source data, iNumberOfBytes contains the amount of source data in bytes. This function allocates an approriate output buffer and converts source data into that buffer. If the operation succeeded it returns true, otherwise (e.g. when memory allocation fails) it returns false.
GetOutputBufSize	Retrieve the size of the buffer that was allocated for conversion.
GetOutputBuf	Retrieve a pointer to the buffer.

6.3.23. ILanguage

Changes the current language for all the windows shown by plug-ins. The Plug-in Manager uses it to store the current language, and since all the components can reach the Plug-in Manager, they can obtain and use it whenever is necessary.

```
interface ILanguage : public IUnknown
{
	bool			SetLanguage(LANGID id, IStatus **ppStatus);
	LANGID			GetLanguage();
};
```

Exposed by

- Audio Plug-in Manager

Function Name	Description
SetLanguage	Change the language ID.
GetLanguage	Retrieve current language settings.

6.3.24. IExtEnum

```
interface IExtEnum : public IUnknown
{
	int			GetCount();
	const char *		GetExt(int iNum);
};
```

Exposed by

- Audio URL source factory
- Audio URL target factory

Function Name	Description
GetCount	Get number of supported items.
GetExt	The returned value can’t be stored for later use. The application must copy it.

6.3.25. IProfile

IProfile provides and interface to the application name. Plug-ins can use this information (along with the "vendor") to store the application specific data. For example if the same plug-in is used from two different applications it is easy for the plug-in to decide where (for example in registry) to store the settings so that they are saved independently.

```
interface IProfile : public IUnknown
{
	bool			SetVendor(const char *szVendor, IStatus **pStatus);
	const char *		GetVendor();
	bool			SetProduct(const char *szProduct, IStatus **pStatus);
	const char *		GetProduct();
};
```

Exposed by

- Audio Plug-in Manager

Function Name	Description
SetVendor	Set the vendor name for the current application.
GetVendor	Retrieve the vendor name.
SetProduct	Set the product name for the current application.
GetProduct	Retrieve the product name.

6.3.26. ILimit

This interface gives access to information about possible limitations of the plug-in, e.g. an allowed number of actions before the trial version of a plug-in needs to be registered.

GetDescription returns a free form text explaining the limitation. The application receives the information from a plug-in and displays it to the user.
Since there can be several types of restrictions (by time, by usage number, etc.) the application can decide to what to display to the user.

```
interface ILimit : public IUnknown
{
	virtual bool		IsUnlimited();
	virtual const char *	GetDescription();
};
```

Exposed by

- Audio URL source factory
- Audio URL target factory

(This interface is generic, and in the future might be exposed by other objects besides the ones mentioned.)

Function Name	Description
IsUnlimited	Indicates whether any restriction apply to the use of the plug-in.
GetDescription	Information about the type of restriction.

6.3.27. ISeekable

ISeekable is a generic interface for positioning inside streams.

```
interface ISeekable : public IUnknown
{
	bool		Seek(ULONGLONG uiNewPos, IStatus **ppStatus);
};
```

Function Name	Description
Seek	Moves the current position in the stream. UiNewPos contains the new position in blocks.

6.3.28. IAggregatable

Allows the object to be aggregated by outer objects in order to override some functionality. For example outer object will aggregate IAudioRawReader interfaces in order to perform conversion from plug-in format to the format requested by the application.

```
interface IAggregatable : public IUnknown
{
	bool				SetAggregator(IUnknown *pAggregator);
	bool				GetAggregator(IUnknown **ppAggregator);
	HRESULT STDMETHODCALLTYPE	InnerQueryInterface(REFIID riid, void *ppObj);
	ULONG STDMETHODCALLTYPE		InnerAddRef();
	ULONG STDMETHODCALLTYPE		InnerRelease();
};
```

Base class for

- IAudioItem
- IAudioComponent

Function Name	Description
SetAggregator	Set the aggregator pointer.
GetAggregator	Return the aggregator pointer.
InnerQueryInterface	Object's real IUnknown, if the object is not aggregated, it must call this functions itself from real IUnknown overridables.
InnerAddRef	Add one to the reference counter.
InnerRelease	Decrease the reference counter by one.

6.3.29. IAudioRawReader

Audio items expose this interface when they can extract audio RAW data.

This interface corresponds to the EAF_Readable auxiliary audio item flag.

```
interface IAudioRawReader : public IUnknown
{
	bool		GetRawFormat(SWavFormat &format, IStatus **ppStatus);
	bool		RawRead(BYTE *pBuf, int iBufSize, int *piRead, IStatus **ppStatus);
	ULONGLONG	GetRawLen();
};
```

Function Name	Description
GetRawFormat	Return information about the RAW format.
RawRead	iBufSize must contain the size in bytes of the buffer pointed by pBuf.
GetRawLen	Returns RAW data length in bytes.

6.3.30. IControl

Provides information about a screen control (button, check box, etc.).

```
interface IControl : public IUnknown
{
	const char *		GetClassName();
	const char *		GetTitle();
	int			GetID();
};
```

Function Name	Description
GetClassName	Return the Class Name property of the control.
GetTitle	Return the Title property of the control.
GetID	Return the ID property of the control.

6.3.31. IEventReceiver

Can be implemented by any audio component or application side object in order to receive events. The object, besides exposing this interface, also has to be subscribed to through the plug-in manager.
This interface is analogue to the standard COM IAdviseSink interface.

The plug-in manager dispatches general purpose notifications to all the subscribed objects. Usually these subscribers are implemented in the plug-ins, however they can be implemented anywhere (in the manager itself, in an application, etc.).

For example when application changes the language of the plug-in manager, it sends a notification, and all the working user interface parts are translated on the fly.

```
interface IEventReceiver : public IUnknown
{
	void		ReceiveEvent(EEvent event, IUnknown *pSource);
};
```

Function Name	Description
ReceiveEvent	Called if an event occurs.

6.3.32. IConfigurable

Any object can expose this interface in order to become configurable.

Configure usually drops out a dialog (usually) and lets the user change the plug-in configuration. If the configuration is changed (the user made some real changes and, for example, clicked OK), Configure returns true, otherwise false. Thus the application does not know anything about the nature of the altered plug-in settings.

```
interface IConfigure
{
	bool		Configure();
};
```

Function Name	Description
Configure	Initiate the configuration.

6.3.33. IVendorInfo

This interface can provide information about the vendor of a plug-in, and the plug-in itself.
The information that IVendorInfo presents can be displayed by Nero 5.5.10.14 and later versions.

```
interface IVendorInfo
{
	const char*	GetVendorName()
	bool		CanDisplayAboutBox();
	void		DisplayAboutBox()
};
```

Function Name	Description
GetVendorName	Return the vendor's name.
CanDisplayAboutBox	Return true if the plug-in is capable of displaying an About box.
DisplayAboutBox	If available, display the About box.

6.4. Types

6.4.1. EAudioComponentType

Available types of audio components. Values, which are not listed here, are reserved and may not be used!

```
enum EAudioComponentType
{
	ACT_None                              = 0,
	ACT_PluginMgr                         = 1,
	ACT_ComponentEnumerator               = 2,
	ACT_AudioURLSourceFactory             = 3,
	ACT_AudioURLTargetFactory             = 4,
	ACT_ConvertorFactory                  = 5,
	ACT_Downloader                        = 6
};
```

6.4.2. EURLType

Types of processable URLs. Values, which are not listed here, are reserved and may not be used!

```
enum EURLType
{
	URL_None                              = 0x00000000,
	URL_LocalFile                         = 0x00000001,
	URL_Http                              = 0x00000002,
	URL_Ftp                               = 0x00000004
};
```

6.4.3. EAuxFlags

Auxilliary flags. Values, which are not listed here, are reserved and may not be used!

```
enum EAuxFlags
{
	EAF_None                              = 0x00000000,
	EAF_CanRecOnAudioCD                   = 0x00000010,
	EAF_CanRecOnDataCD                    = 0x00000020,
	EAF_CanCopy                           = 0x00000040,
	EAF_CanDLToMp3PlayerOrPortabDvc       = 0x00000080,
	EAF_CanConvert                        = 0x00000100,
	EAF_Readable                          = 0x00001000,
	EAF_Playable                          = 0x00002000,
	EAF_SrcCanSaveAddInfo                 = 0x00010000,
	EAF_RealTimeOnly                      = 0x00020000,
	EAF_LightOpen                         = 0x00040000
};
```

Enumerator	Description
EAF_None	No restrictions.
EAF_CanReacOnAudioCD	Can be burnt on an Audio CD.
EAF_CanRecOnDataCD	Can be burnt on Data CD.
EAF_CanCopy	Copy protection.
EAF_CanDLToMp3PlayerOrPortabDvc	Can be downloaded to an mp3 player or a portable device.
EAF_CanConvert	Can be converted to another format.
EAF_Readable	Indicates that RAW data can be read from the item.
EAF_Playable	Indicates that the source can play itself (also pause, resume, etc.).
EAF_SrcCanSaveAddInfo	Indicates that the source can edit the additional information such as artist and title name, and store it back to the source file.
EAF_RealTimeOnly	Some sources can read/write data at a speed of 1.0 x only (internet streams, CD players and others). Must be specified with EAF_BadReadable or EAF_Readable.
EAF_LightOpen	This flag is declared by the plug-in and can be requested by applications in order to open sources in light version i.e. with minimal information and minimal resource waste (time, memory, connnections and others.). This can be useful for internet radio, big files, etc.

6.4.4. EMainScreenMode

Values, which are not listed here, are reserved and may not be used!

```
enum EMainScreenMode
{
	MSM_None				= 0,
	MSM_Normal				= 1,
	MSM_Shade				= 2,
	MSM_Minimized				= 3
};
```

6.4.5. EAppearMethod

Values, which are not listed here, are reserved and may not be used!

```
enum EAppearMethod
{
	EAM_None				= 0,
	EAM_Photoplate				= 1,
	EAM_Stars				= 2,
	EAM_HorizontalJalousie			= 3,
	EAM_VerticalJalousie			= 4,
	EAM_Compass				= 5,
	EAM_ClockFace				= 6
};
```

6.4.6. EEvent

Events that can be processed by IEventReceivers. Values, which are not listed here, are reserved and may not be used!

```
#define EEvent DWORD
```

Constant	Value	Description
EE_None	0	
EE_AudioSourceStart	100	
EE_MainWndModeChange	200	
EE_MainWndMoving	201	
EE_MainWndMoved	202	
EE_MainWndChanged	203	
EE_SkinChange	300	
EE_AppearMethodChange	301	
EE_SkinFolderChange	302	
EE_LanguageChange	400	
EE_ProfileChange	500	
EE_AppIdle	600	This event is being sent periodically to announce all the architecture components that the main application is in the idle state.
	700	700 - 799 reserved
EE_EqualizerHide	800	The plug-in wants to hide the equalizer
EE_EqualizerShow	801	The plug-in wants to hide the equalizer
EE_EqualizerIsHidden	850	Sent to plug-ins when the application hides the equalizer
EE_EqualizerIsShown	851	Sent to plug-ins when the application shows the equalizer
EE_DownloadStart	900	
EE_DownloadFinish	901	
EE_DownloadProgress	901	

6.4.7. EWndMode

Values, which are not listed here, are reserved and may not be used!

```
enum EWndMode
{
	ESM_Undefined			= 0x00000000,
	ESM_Minimized			= 0x00000001,
	ESM_Maximized			= 0x00000002,
	ESM_Shaded			= 0x00000004,
	ESM_Visible			= 0x00000008
};
```

6.4.8. EAudioRawState

Values, which are not listed here, are reserved and may not be used!

```
enum EAudioRawState
{
	ERS_None			= 0x00000000,
	ERS_FirstAfterSeek		= 0x00000001,
	ERS_EndOfFile			= 0x00000002
};
```

6.4.9. EMediaType

Values, which are not listed here, are reserved and may not be used!

```
enum EMediaType
{
	EMT_None			= 0x00000000,
	EMT_BasicAudio			= 0x00000001,
	EMT_ExtAudio			= 0x00000002
};
```

Enumerator	Description
EMT_None	
EMT_BasicAudio	8 and 16 bit, whole frequency range, 1 or 2 channels.
EMT_ExtAudio	8, 16, 24, 32 bit, whole frequency range, 1 or 2 channels.

6.4.10. ELimitType

Values, which are not listed here, are reserved and may not be used!

```
enum ELimitType
{
	ERT_NoRestrictions			= 0,
	ERT_LimitedTimes			= 1,
	ERT_Expired				= 3,
	ERT_PartiallyLimited			= 4
};
```

Enumerator	Description
ERT_PartiallyLimited	One part has no restrictions, the other is limited. For example, MP3 fully working but mp3PRO is limited.

6.4.11. EConvFactoryMode

Values, which are not listed here, are reserved and may not be used!

```
enum EConvFactoryMode
{
	ECFM_Playback,
	ECFM_Other
};
```

6.5. The Plug-in Utility Files

The plug-in utility files *NeroPluginUtil.h* and *NeroPluginUtil.cpp* offer functions for storing and retrieving plug-in parameters. The Windows registry is used to store this information.

If szGroup is not NULL, the existence of an additional subgroup is assumed. This parameter can be NULL for no additional groups indication.

6.5.1. reg_SetPluginParam

This function is used to store DWORD or string values.

DWORD

```
bool   reg_SetPluginParam(   const char *szVendor,
                             const char *szProduct,
                             const char *szPlugin,
                             const char *szGroup,
                             const char *szParam,
                             DWORD dwValue);
```

String

```
bool   reg_SetPluginParam(   const char *szVendor,
                             const char *szProduct,
                             const char *szPlugin,
                             const char *szGroup,
                             const char *szParam,
                             const char *szValue);
```

6.5.2. reg_GetPluginParam

This function is used to retrieve values.

DWORD

```
bool   reg_GetPluginParam(   const char *szVendor,
                             const char *szProduct,
                             const char *szPlugin,
                             const char *szGroup,
                             const char *szParam,
                             DWORD &dwValue);
```

String

```
CString reg_GetPluginParam(  const char *szVendor,
                             const char *szProduct,
                             const char *szPlugin,
                             const char *szGroup,
                             const char *szParam);
```

6.5.3. file_DoesExist

Returns true if szFile exists.

```
bool   file_DoesExist(const char *szFile);
```

6.5.4. file_GetSize

Returns the size of szFile.

```
DWORD  file_GetSize(const char *szFile);
```

6.5.5. file_GetTime

Returns the time of last access to the file, which is, actually, the time of last reading/writing to the file.

```
bool   file_GetTime(const char *szFile, CTime &time);
```

6.5.6. IsURLLocal

Returns true if the URL points to a local file.

```
bool   IsURLLocal(const char *szURL);
```

6.5.7. EAudioErrorMapSysErrorToAudio

Maps a system error to EAudioError.

```
EAudioError MapSysErrorToAudio(DWORD dwErr);
```

6.6. Files For Localization

It is not obligatory to use the following method for localization. It is one possibility that has proven to be feasible.

So if you prefer another way to implement localization, do not feel obliged to use the CTranslator class.

Independent of how many languages you intend to support, please make sure that at least English is available.

6.6.1. NeroPluginNls.h And NeroPluginNls.cpp

Declaration and implementation of the CTranslator class. CTranslator translates windows and separate expressions.

```
class CTranslator
{
public:
       CTranslator(WORD wResID, const LOGFONT *pLF, CString &rcsLang);

public:
       void   TranslateWindow(HWND hMainWnd);
       void   GetWordTranslation(const char *szWord, CString *pcs);

protected:

       LOGFONT        m_LogFont;
       CString        m_csLang;
       CStringArray   m_strarEng,    m_strarTgt;
};

CTranslator *  CreateTranslator(WORD wResID, const LOGFONT *pLF, CString &rcsLang);
void           DestroyTranslator(CTranslator *pTranslator);
```

6.6.2. Translation File Format

Every plug-in contains a single text file with translations of all expressions used in the user interface of the plug-in.

```
[language code]
<number><space><expression>
<number><space><expression>
<number><space><expression>

[language code]
<number><space><expression>
<number><space><expression>
<number><space><expression>
```

6.6.3. Translation File Example

```
[ENG]
1 Frequency
2 Bits
3 Channels
4 Mono
5 Stereo
6 OK
7 Cancel

[HUN]
1 Frekvencia:
2 bitek
3 Csatornák:
4 Mono
5 Sztereo
6 OK
7 Mégse
```

The plug-in creates an object of the CTranslator class which parses this file. Then if the contents of a window need to be translated the CTranslator::TranslateWindow() function is called and the handle of the window being translated is passed. TranslateWindow() recursively scans the window and translates every subwindow's text. If those windows are combo or list boxes, also their content. For other types of container controls such us tree controls, additional content translation must be implemented.

If a separate expression should be translated, the CTranslator::GetWordTranslation() function is called.

7. Error Handling

7.1. EAudioError

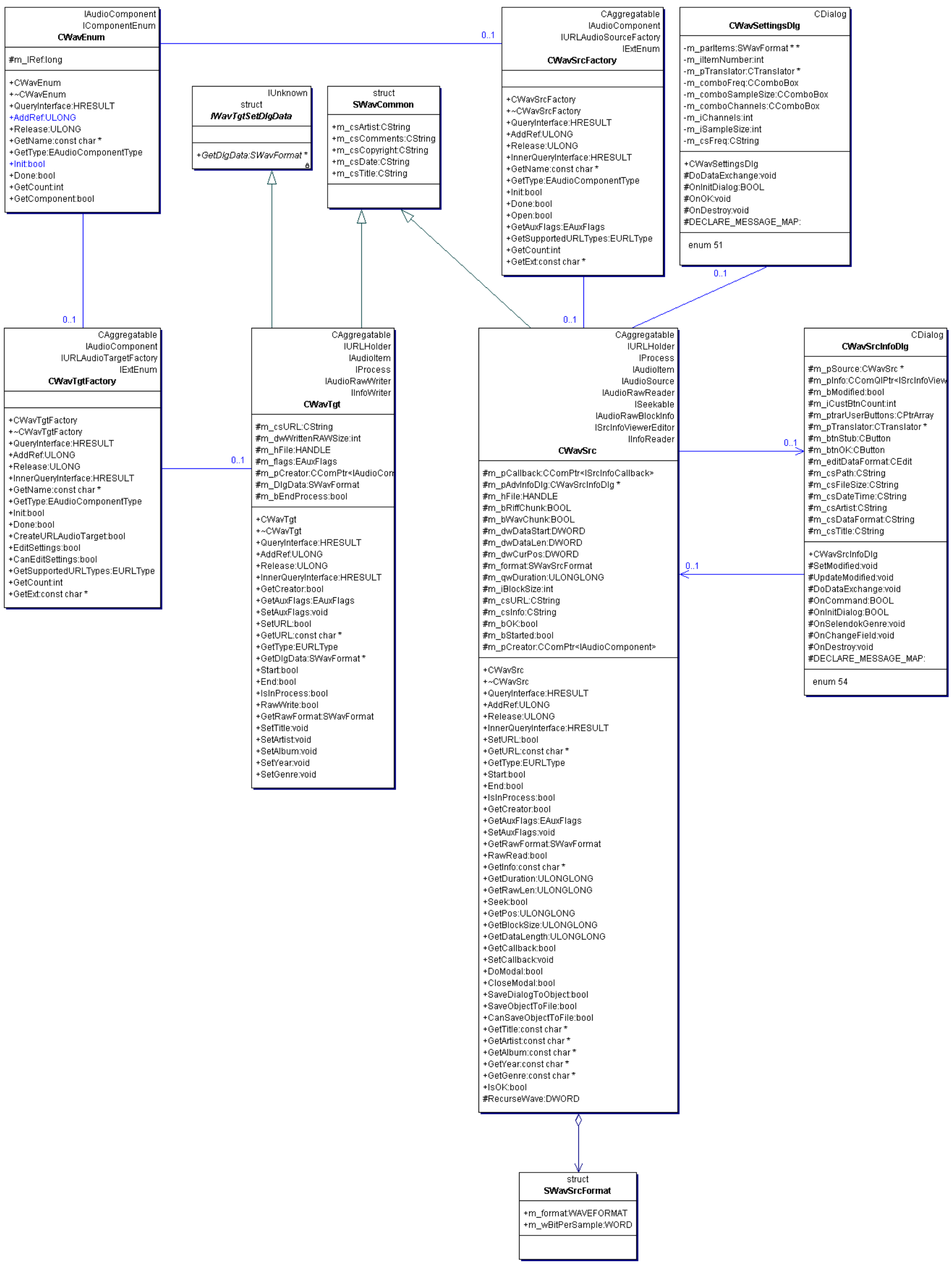
All audio errors are defined in the EAudioError enumeration. Values, which are not listed here, are reserved and may not be used!

```
enum EAudioError
{
    AE_Undefined                 = 0x00000000,
    AE_Failure                   = 0x80000000,
    AE_InvalidParameter          = 0x80000001,
    AE_InsufficientBuffer        = 0x80000002,
    AE_BadPathName               = 0x80000003,
    AE_FileInvalid               = 0x80000004,
    AE_CallNotImplemented        = 0x80000005,
    AE_EOF                       = 0x80000006,
    AE_OpenFailed                = 0x80000007,
    AE_SeekFailed                = 0x80000008,
    AE_BadFormat                 = 0x80000009,
    AE_ServiceNotActive          = 0x80000010,
    AE_ReadFault                 = 0x80000011,
    AE_WriteFault                = 0x80000012,
    AE_WrongCall                 = 0x80000013,
    AE_FileNotFound              = 0x80000014,
    AE_Aborted                   = 0x80000015,
    AE_Skip                      = 0x80000016,
    AE_STUB
};
```

8. Creating A Wave Plug-in

On the following pages we will guide you through creating a Nero Audio Plug-in that is capable of reading and writing data in Wave format.

We have used Visual C++ 6.0 to create and build this example. But of course, every compiler and programming language that is able to produce COM objects can be used to build your own plug-ins.



Class overview of the plug-in

8.1. Module Overview

The following table describes the modules that we need to create to implement the desired functionality.

File	Description
dlg_WavSettings.cpp	Implementation of CWavSettingsDlg, an MFC dialog class that lets the user change target settings like sample rate, bit resolution or number of channels.
dlg_WavSettings.h	Header File for CWavSettingsDlg.
dlg_WavSrcInfo.cpp	Implementation of CWavSrcInfoDlg, an information window that provides details about the currently selected source file.
dlg_WavSrcInfo.h	Header File for CWavSrcInfoDlg.
FileHelper.cpp	Implementation of helper functions for file access.
FileHelper.h	Header file for the file helper functions.
myWav.cpp	The main file of the plug-in. Contains the implementation of the NERO_PLUGIN_GetPrimaryAudioObject and NERO_PLUGIN_ReadyToFinish functions.
Wav.def	Definition file for exported functions.
WavEnum.cpp	Implementation of CWavEnum, a factory enumerator class.
WavEnum.h	Header file for CWavEnum.
WavFormat.h	Wave format definitions and structs, including RAW data format and flags.
WavPluginNlsData.txt	Translation file for multi-language support.
WavSrc.cpp	Implementation of CWavSrc, a wave source class which is capable of reading wave data from a file and presenting it in RAW format.
WavSrc.h	Header file for CWavSrc.
WavSrcFactory.cpp	Implementation of CWavSrcFatory, the wave source factory which can create instances of the CWavSrc class.
WavSrcFactory.h	Header file for CWavSrcFactory.
WavTgt.cpp	Implementation of CWavTgt, the wave target class, which can receive wave data in RAW format and write it to a file in wave format.
WavTgt.h	Header file for CWavTgt.
WavTgtFactory.cpp	Implementation of CWavTgtFactory, the wave source factory which can create instances of the CWavTgt class.
WavTgtFactory.h	Header file for CWavTgtFactory.

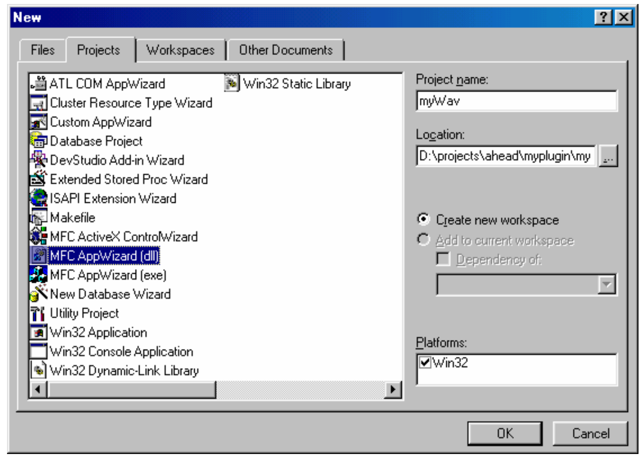
As you can see from the diagram and the table, there is a large number of classes and functions involved. Therefore we will not go through every implementation detail but focus on the key aspects – creating the project's resources, implementing the enumerator and the factories. Complete listings of all files can be found at the end of this document.

8.2. A New Project

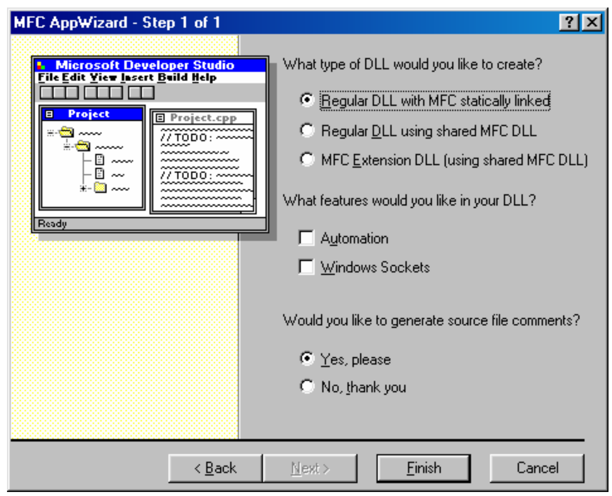
The first step towards the plug-in is the creation of a new Microsoft Visual C++ project.

Select File/New from the menu, then choose MFC Application Wizard (dll) from the Projects tab and name the project “MyWav”.

Click the OK button.



In the next and last Wizard step choose “Regular DLL with MFC statically linked”. Click the “Finish” button.

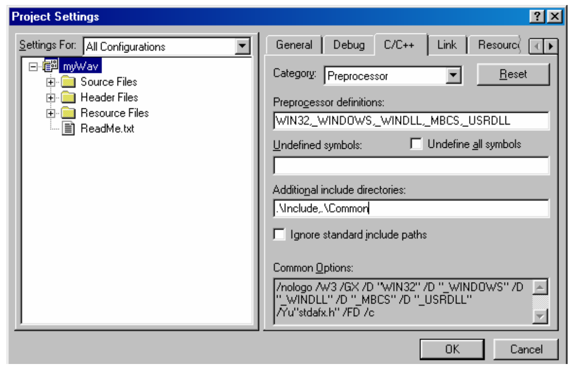


8.3. Adding Include Directories

Make the Common and Include directories of the Audio Plug-in Manager files known to your application. The easiest way is to use Windows Explorer to copy both directories in to the myWav directory.

Then open Visual Studio’s project settings dialog and choose “All Configurations” from the “Settings For” listbox. Open the C/C++ Tab and select the “Preprocessor” category.

Under “Additional include directories” type: **.\Include,.\Common**.

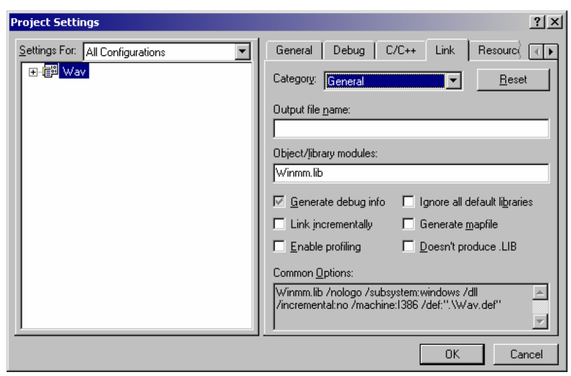


8.4. Linking with the Windows Multimedia Library

Windows offers convenient ways of reading audio data from files. The required functions are packed together in the Windows Multimedia Library, which needs to be linked to our project.

Open the “Link” Tab and under “Object/library modules” add “Winmm.lib”.

Close the settings dialog.



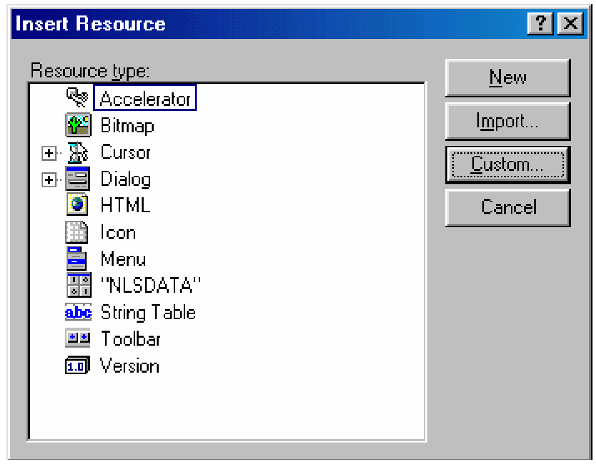
8.5. Adding An NLS Resource

In this plug-in we intend to offer the user multi-language support. The first step towards supporting multiple languages is the creation of a translation file that could look like the following. This file supports English and Dutch:

```
[ENG]
1 Frequency
2 Bits
3 Channels
4 Mono
5 Stereo
6 OK
7 Cancel
8 File details
9 Path
10 File size
11 Date/Time
12 Title
13 Artist
14 Data format
15 Close
16 Format
17 Samplerate
18 BitPerSample

[NLD]
1 Frequentie
2 bits
3 Kanalen
4 Mono
5 Stereo
6 OK
7 Annuleren
8 Details bestand
9 Pad
10 Bestandsgrootte
11 Datum/tijd
12 Titel
13 Kunstenaar
14 Format data
15 Sluiten
16 Indeling
17 Samplesnelheid
18 BitPerSample
```

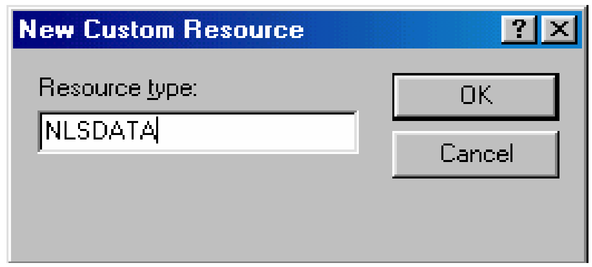
Create a new text file with the Microsoft Visual C++ editor or any other editor that you prefer. Add the text from the example above and save it as “WavPluginNlsData.txt”.

Now the translation file must be added as a resource. First we have to create a new resource Type "NLSDATA": Change from ClassView to ResourceView inside the Workspace window and right click on the myWav Resources folder icon. Select "Insert…" from the context menu and click the "Custom..." button.



In the following dialog type "NLSDATA" and click OK.

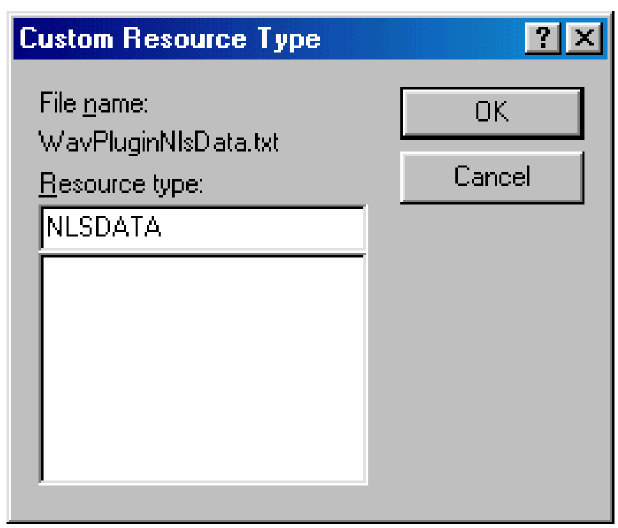
You will now see a new folder "NLSDATA" and a new resource that has been added "IDR_NLSDATA1". Rename "IDR_NLSDATA1" to "IDR_PLUGIN_NLSDATA".



Right click "IDR_PLUGIN_NLSDATA" and select "Import…". In the following file selection dialog change the file type to "All Files (*.*)" and pick the formerly created "WavePluginNlsData.txt".

Click "Open".

Visual Studio asks you to select the proper resource type from custom resources. Currently there should only be one available – "NLSDATA". Pick that and click "OK". Your translation file is now available as a resource and will be displayed in a binary view.



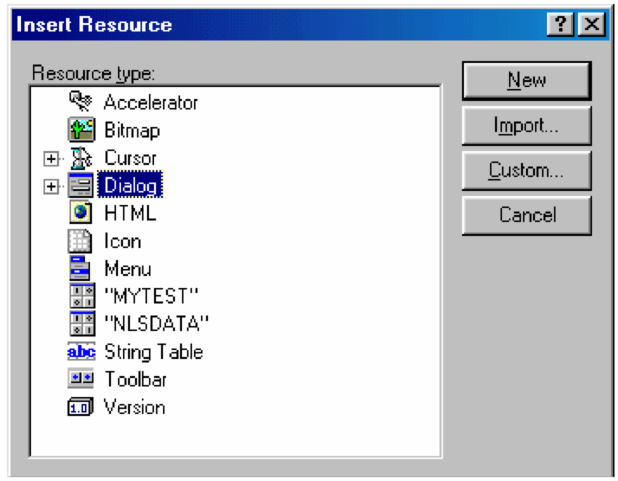
8.6. Wave Format Definitions And Structs

With File/New create a C/C++ header file and name it “WavFormat.h”. In this file we provide the necessary information about the wave format. It is required for both the source and target items.

```
#ifndef _WAV_FORMATD_
# define _WAV_FORMATD_

#if _MSC_VER > 1000
# pragma once
#endif // _MSC_VER > 1000


// Headers

// length of the name part
const int WAV_CHUNK_NAME_LEN = 4;

// the number of supported subchunks of the LIST chunk
const int LIST_CHUNK_ITEMS = 5;

// source flags
const EAuxFlags SRC_FLAGS = ((EAuxFlags) (EAF_LightOpen|
                                          EAF_Readable|
                                          EAF_CanRecOnAudioCD|EAF_CanRecOnDataCD|
                                          EAF_CanCopy|
                                          EAF_CanDLToMp3PlayerOrPortabDvc|
                                          EAF_CanConvert));

// SWavCommon

// A base structure for both source and target item classes.
// Incapsulates the additional information.
struct SWavCommon
{
// Data

// the order of placement of this fields MUST not
// be changed or MUST be arranged with the second
// switch of Recurse Wave function

  CString   m_csArtist;
  CString   m_csComments;
  CString   m_csCopyright;
  CString   m_csDate;
  CString   m_csTitle;
};

#endif // _WAV_FORMATD_
```

8.7. The Wave Target Settings Dialog

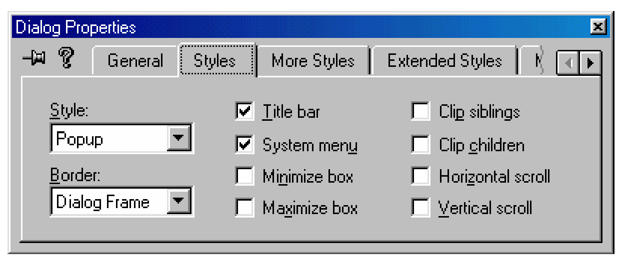
Another required resource is the Settings Dialog that lets the user change the format of the Wave data.



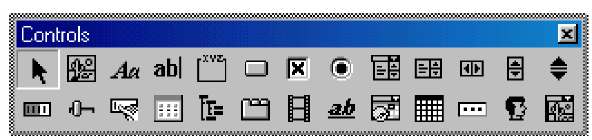
Inside the "Insert Resource" dialog, double click the "Dialog" icon. This will create a standard Dialog box that already contains an OK and Cancel button.

Right click over the new dialog and select "Properties". In the "General" tab change the ID to "IDD_WAV_SETTINGS".

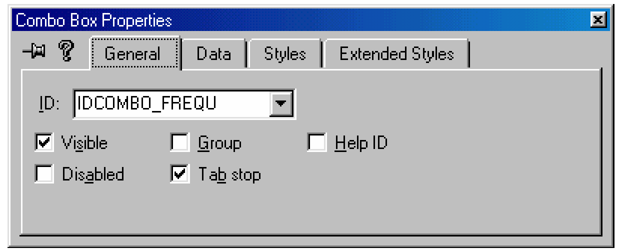
Inside the "Styles" tab the proper settings already should have been made by Visual Studio.



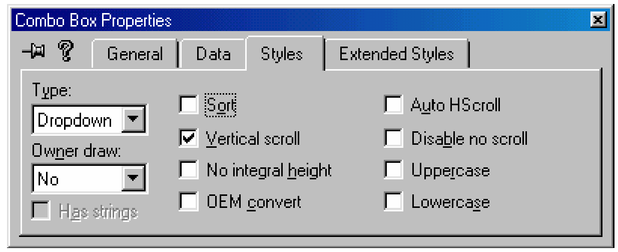
Make sure that you can see the "Controls" window because the dialog still needs a few controls.



Select a ComboBox control control and place it on the dialog. Change the ID to "IDCOMBO_FREQU" in the properties window.

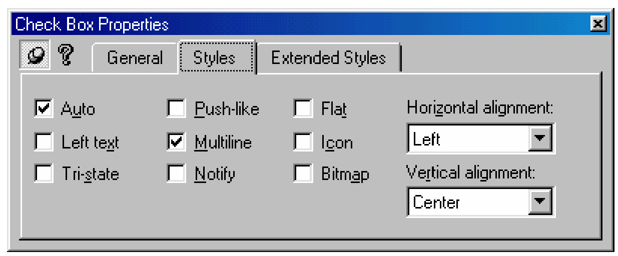


Uncheck the "Sort" option in the "Styles" tab.



Repeat the ComboBox procedure to create two other dropdown listboxes "IDCOMBO_SAMPLE_SIZE" and "IDCOMBO_CHANNELS".

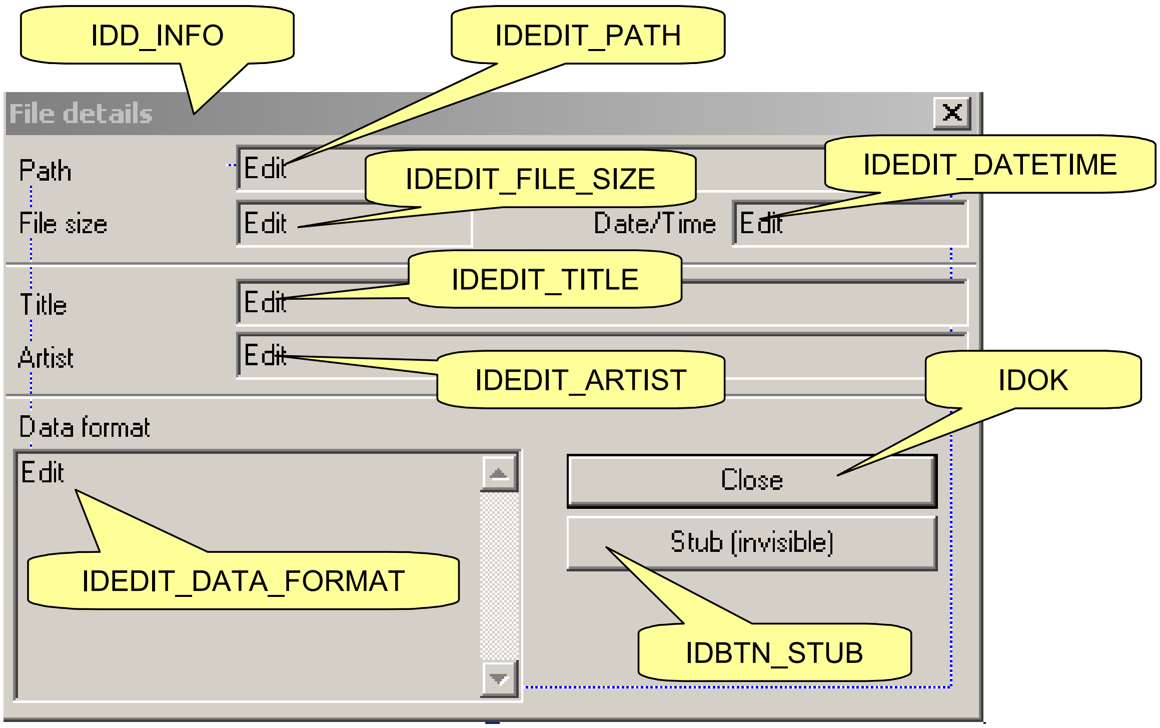
Now add a Check Box control and change the ID to "IDCHECK_REMEMBER_OPT". Go to the "Styles" tab and make it "Multiline", also changing the alignment to "Left" and "Center"



Finally change the dialog heading to "myWav" and add labels with captions that show "Frequency", "Bits" and "Channels". Change the Check Box caption to "Remember these options for later use".

8.8. The File Details Dialog

Create this dialog just like the Wave Target Settings Dialog. Of course the layout and the IDs differ, as you can see in the picture below.



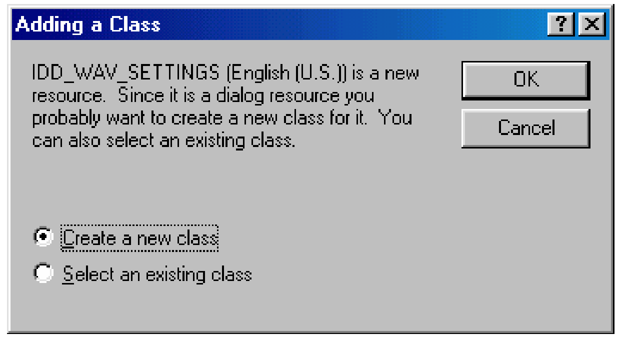
The creation of this dialog concludes work on resources.

8.9. Adding The Wave Target Settings Dialog Class

In the next step the wave target settings dialog resource needs to be connected to a C++ class.

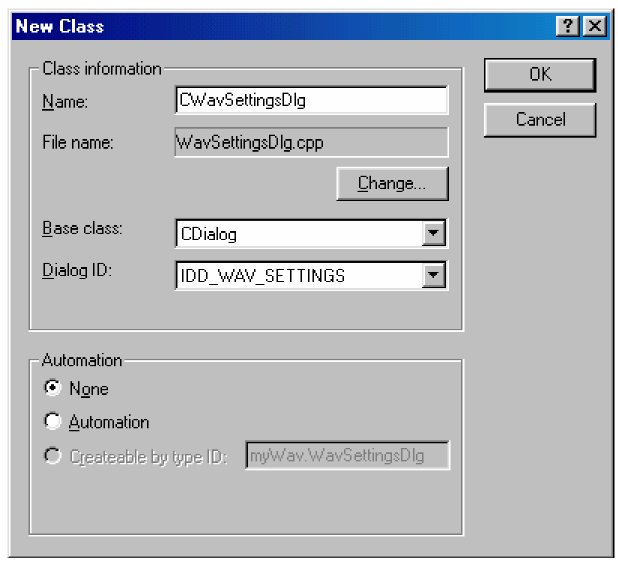
Select the IDD_WAV_SETTINGS dialog and open the Class Wizard by pressing “Ctrl + W”.

The Class Wizard will prompt you to connect the dialog resource to a class. Select “Create a new class” and click OK.

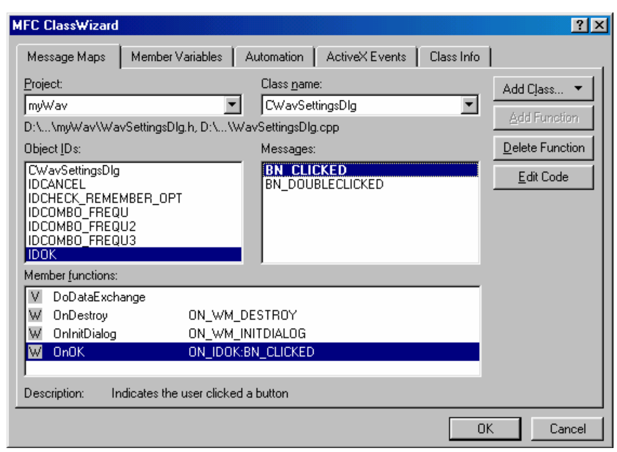


In the following New Class dialog name the class “CWavSettingsDlg” and make sure the settings look like the screenshot on the right.

Click OK.



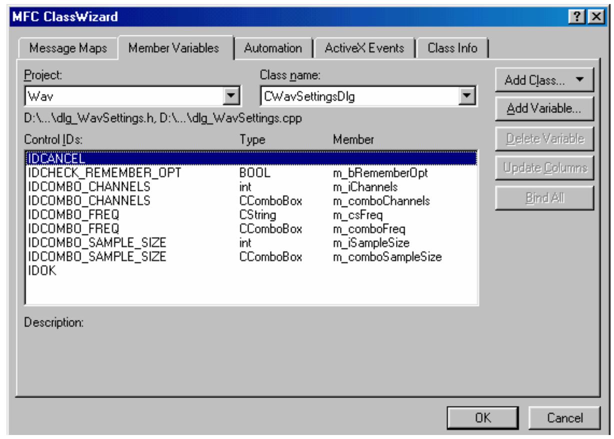
A few member functions to handle Windows events still need to be created. Add functions for the “ON_WM_DESTROY”, “ON_WM_INITDIALOG” and “ON_IDOK_BN_CLICKED” messages for the CWaveSettingsDlg class.



In addition to these functions the dialog needs a few member variables for the dialog's controls.

Change to the "Member Variables" tab and add variables to resemble the list shown on the right.

Close the Class Wizard by clicking OK.



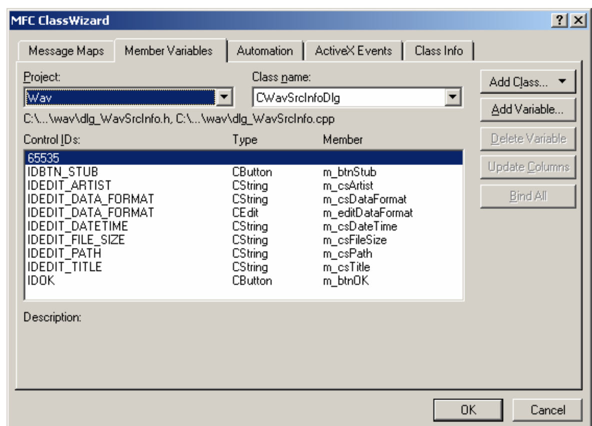
The complete implementation of CWavSettingsDlg is listed at the end of this document.

8.10. Adding The File Details Dialog Class

Now we will assign the File Details dialog to a class.

Open the Class Wizard again, and perform the steps required to create the File Details Dialog class, just like you did for CWavSettingsDlg.

Name the new class CWavSrcInfoDlg, and connect the Controls as shown in the screenshot on the right.



8.11. The Wave Target Dialog Class

This class, is relatively large, so its listing can be found at the end of this document.

The one important fact about CWavSettingsDlg is that it is instanciated by CWavTgtFactory, the wave target factory class.

8.12. The Wave Source Class

CWavSrc inherits from a number of classes and interfaces:

- CAggregatable
- IURLHolder
- IProcess
- IAudioItem
- IAudioSource
- IAudioRawReader
- ISeekable
- IAudioRawBlockInfo
- ISrcInfoViewerEditor
- IInfoReader
- SWavCommon

The class definition and implementation are extensive, and the handling of data is very specific to the wave format. Therefore the listings of WavSrc.h and WavSrc.cpp are not included here, but rather can be found at the end of this document.

The behavior of both source and target classes is shown in the diagram in chapter **5.3 Transfer Data**.

8.13. The Wave Target Class

The CWavTgt class inherits these classes and interfaces:

- IURLHolder
- IAudioItem,
- IProcess,
- IAudioRawWriter,
- IWavTgtSetDlgData,
- IInfoWriter,
- SWavCommon

Similar to the Wave Source class this is a large class. We will not discuss it in detail. The listings for WavTgt.h and WavTgt.cpp can be found at the end of this document.

8.14. The Wave Source Factory Class

The Source Factory class is capable of creating instances of the CWavSrc class.

It inherits from

- IIdentifiable
- CAggregatable,
- IAudioComponent,
- IURLAudioSourceFactory,
- IExtEnum
- IVendorInfo

Create a header file WavSrcFactory.h and an implementation file WavSrcFactory.cpp.

8.14.1. WavSrcFactory.h

The header file contains declarations for the implemented interfaces and the aggregation map.

AudioBase.h contains the definitions of CAggregatable, and includes AudioPluginEnv.h, where the necessary interface definitions can be found:

```
// CStatus and CAggregatable
#include "AudioBase.h"
```

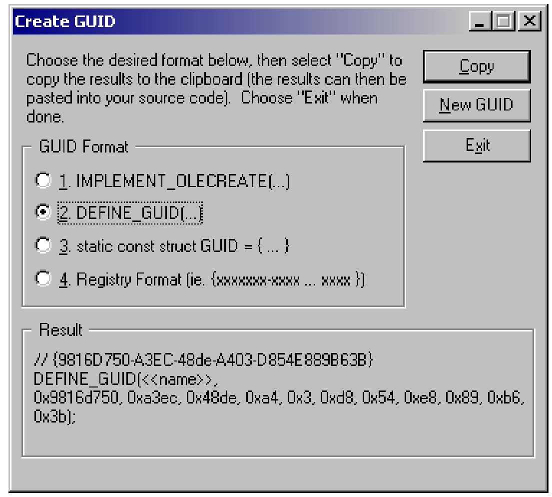
The rest of the declaration is straightforward:

```
class CWavSrcFactory :  public IIdentifiable,
                        public CAggregatable,
                        public IAudioComponent,
                        public IURLAudioSourceFactory,
                        public IExtEnum,
                        public IVendorInfo
{
// Construction/Destruction
public:
  CWavSrcFactory()
  : CAggregatable (AGGFLAG_NODELETE)
  {}

  virtual ~CWavSrcFactory() {};

// Operations
public:
  virtual long GetRefCount();
```

```
  AGGREGATABLE_INTERFACE_MAP_BEGIN
    INTERFACE_ENTRY(IIdentifiable)
    INTERFACE_ENTRY(IAudioComponent)
    INTERFACE_ENTRY(IURLAudioSourceFactory)
    INTERFACE_ENTRY(IExtEnum)
    INTERFACE_ENTRY(IVendorInfo)
  AGGREGATABLE_INTERFACE_MAP_END

  // IIdentifiable
  void GetID(GUID *pGUID);

  // IAudioComponent
  virtual const char* GetName();
  virtual EAudioComponentType GetType();
  virtual bool Init(IAudioPluginMgr* pMgr, IStatus** ppStatus);
  virtual bool Done();
  virtual bool
  Open( const char*   szURL,
        IUnknown**    ppSrc,
        EAuxFlags     flagsInclude,
        EAuxFlags     flagsExclude,
        IStatus**     ppStatus);

  virtual EAuxFlags GetAuxFlags();

  virtual EURLType GetSupportedURLTypes();

  // IExtEnum
  int GetCount();

  // The returned value can't be stored for later use.
  // The application must copy it.
  const char* GetExt(int iNum);

  // IVendorInfo
  const char * GetVendorName();
  bool CanDisplayAboutBox();
  void DisplayAboutBox();
};
```

8.14.2. WavSrcFactory.cpp

This class requires that a GUID (globally unique identifier) is created. Please run GUIDGEN.exe, and select the "DEFINE_GUID{…}" format to create a GUID. Then use the copy button and paste it into your source code!



The implementation file must include WavSrc.h because it will instanciate CWavSrc objects.

```
#include "stdafx.h"
#include "WavSrcFactory.h"
#include "WavSrc.h"
```

Change the output of GUIDGEN to this format:

```
EXTERN_GUID(COMPID_SrcFactory, 0x459215ca, 0x50f5, 0x42cc, 0xbf, 0x51, 0x84, 0x38, 0x11,
0x9d, 0xee, 0x6c);
```

A pointer to the Plug-in Manager itself is stored in a global variable, which is defined in the main module. The external reference will be resolved when the project is linked.

```
extern IAudioPluginMgr* g_pPluginMgr;
```

GetID implements the mandatory IIdentifiable interface.

```
void CWavSrcFactory::GetID(GUID *pGUID)
{
  ASSERT(pGUID);
  if(pGUID) memcpy(pGUID, &COMPID_SrcFactory, sizeof(GUID));
}
```

GetName returns the name of the supported format as string.

```
const char* CWavSrcFactory::GetName()
{
  return "PCM Wav file";
}
```

GetType returns the type of the component, in this case ACT_AudioURLSourceFactory.

```
EAudioComponentType CWavSrcFactory::GetType()
{
  return ACT_AudioURLSourceFactory;
}
```

The Init method receives a pointer to the Audio Manager object. If the global pointer has been stored during a call to another component (the component enumerator), we make sure that the pointers global pointer equals the parameter. If the Audio Manager pointer has not been set so far, g_pPluginManager is set for the first time here.

```
bool CWavSrcFactory::Init(IAudioPluginMgr* pMgr, IStatus** ppStatus)
{
  if(NULL != ppStatus)
  {
    *ppStatus = NULL;
  }

  if(NULL != g_pPluginMgr)
  {
    ASSERT(g_pPluginMgr == pMgr);
  }
  else
  {
    g_pPluginMgr = pMgr;
  }

  // success
  return true;
}
```

The Done method merely returns true because no additional cleanup is required. In general the combination of Init/Done can be used to start and stop any software engine that might be required for proper operation of a particular Plug-in.

```
bool CWavSrcFactory::Done()
{
  return true;
}
```

GetAuxFlags returns the source flags that have been defined in WavFormat.h. WavFormat.h is know through inclusion in WavSrc.h.

```
EAuxFlags CWavSrcFactory::GetAuxFlags()
{
  return SRC_FLAGS;
}
```

We only support local files.

```
EURLType CWavSrcFactory::GetSupportedURLTypes()
{
  return URL_LocalFile;
}
```

GetCount is a member of IExtEnum and returns the number of supported extensions. We support two.

```
int CWavSrcFactory::GetCount()
{
  return 2;
}
```

Depending on the iNum parameter one of the supported extensions is returned. The returned value cannot be stored for later use. The application must copy it.

```
const char* CWavSrcFactory::GetExt(int iNum)
{
  if(0 == iNum)
  {
    return "wav";
  }
  else if(1 == iNum)
  {
    return "wave";
  }

  ASSERT(FALSE);

  return NULL;
}
```

The Open method creates a new CWavSrc object, providing the file name, and the this-pointer for later reference to the parent object.

```
bool CWavSrcFactory::Open(const char* szURL,
                          IUnknown**  ppSrc,
                          EAuxFlags flagsInclude,
                          EAuxFlags flagsExclude,
                          IStatus** ppStatus)
{
  if(ppStatus)
  {
    *ppStatus = NULL;
  }

  CWavSrc* pSrc = new CWavSrc(szURL, (IAudioComponent*)this, ppStatus);

  bool bOK = pSrc->IsOK();

  if(bOK)
  {
    *ppSrc = static_cast<IAggregatable*>(pSrc);
```

```
    (*ppSrc)->AddRef();
  }
  else
  {
    delete pSrc;

    *ppSrc = NULL;
  }

  return bOK;
}
```

GetRefCount returns the reference counter, which can be used to determine whether or not the object is still in use by any other object.

```
long CWavSrcFactory::GetRefCount()
{
  return m_lRef;
}
```

The following three functions implement the IVendorInfo interface.
Get VendorName will return a string “Example Vendor”, CanDisplayAboutBox will tell the Plug-in Manager that an About Box is available, and DisplayAboutBox simply creates a Message Box which displays "Nero Example PCM WAV plug-in."

```
const char *CWavSrcFactory::GetVendorName()
{
  return "Example Vendor";
}

bool CWavSrcFactory::CanDisplayAboutBox()
{
  return true;
}

void CWavSrcFactory::DisplayAboutBox()
{
  AfxMessageBox("Nero Example PCM WAV plug-in.");
}
```

8.15. The Wave Target Factory Class

The Source Factory class is capable of creating instances of the CWavTgt class.

It inherits from

- IIdentifiable
- CAggregatable,
- IAudioComponent,
- IURLAudioTargetFactory,
- IExtEnum

Create a header file WavTgtFactory.h and an implementation file WavTgtFactory.cpp.

8.15.1. WavTgtFactory.h

AudioBase.h contains the definitions of CAggregatable, and includes AudioPluginEnv.h, where the necessary interface definitions can be found.

```
// CStatus and CAggregatable
#include "AudioBase.h"

class CWavTgtFactory :  public  IIdentifiable,
                        public  CAggregatable,
                        public  IAudioComponent,
                        IURLAudioTargetFactory,
                        IExtEnum
{
// Construction/Destruction
public:
  CWavTgtFactory()
  : CAggregatable (AGGFLAG_NODELETE)
  {}

  virtual ~CWavTgtFactory() {};

// Operations
public:
  virtual long GetRefCount();

  AGGREGATABLE_INTERFACE_MAP_BEGIN
    INTERFACE_ENTRY(IIdentifiable)
    INTERFACE_ENTRY(IAudioComponent)
    INTERFACE_ENTRY(IURLAudioTargetFactory)
    INTERFACE_ENTRY(IExtEnum)
  AGGREGATABLE_INTERFACE_MAP_END

  // IIdentifiable
  void GetID(GUID *pGUID);

  // IAudioComponent
```

```
  virtual const char* GetName();
  virtual EAudioComponentType GetType();
  virtual bool Init(IAudioPluginMgr* pMgr, IStatus** ppStatus);
  virtual bool Done();

  // IURLAudioTargetFactory
  virtual bool CreateURLAudioTarget(IUnknown** ppTgt,
                                    const SWavFormat& formatSrc,
                                    IStatus** ppStatus);

  virtual bool EditSettings(IUnknown** ppTgt, int iCount);

  // Application should first call this function to determine if this
  // factory can edit settings for items.
  virtual bool CanEditSettings();
  virtual EURLType GetSupportedURLTypes();

  // IExtEnum
  int GetCount();

  // The returned value can't be stored for later use.
  // The application must copy it.
  const char* GetExt(int iNum);
};
```

8.15.2. WavTgtFactory.cpp

Just like CWavSrcFactory, this class requires that a GUID is created. Please run GUIDGEN.exe, and select the “DEFINE_GUID{…}” format to create a GUID. Then use the copy button and paste it into your source code!

The implementation file must include WavTgt.h because it will instanciate CWavSrc objects. Also the Wave Settings dialog header file must be included because the settings dialog is called by the factory.

```
#include "stdafx.h"

#include "WavTgtFactory.h"

// {9E424B23-3D6A-48ca-A41D-B65927057499}
EXTERN_GUID(COMPID_TgtFactory , 0x9e424b23, 0x3d6a, 0x48ca, 0xa4, 0x1d, 0xb6, 0x59, 0x27,
0x5, 0x74, 0x99);

// Wave Target class
#include "WavTgt.h"

// Wave Settings dialog
#include "dlg_WavSettings.h"

extern IAudioPluginMgr* g_pPluginMgr;
```

The Target Factory’s GetID function implements the mandatory IIdentifiable interface, which is required for every source and target factory.

```
void CWavTgtFactory::GetID(GUID *pGUID)
{
       ASSERT(pGUID);
       if(pGUID) memcpy(pGUID, &COMPID_TgtFactory, sizeof(GUID));
}
```

GetName returns the name of the supported format.

```
const char * CWavTgtFactory::GetName()
{
  return "PCM Wav file";
}
```

This component is an ACT_AudioURLTargetFactory.

```
EAudioComponentType CWavTgtFactory::GetType()
{
  return ACT_AudioURLTargetFactory;
}
```

Initialization is the same as for CWavSrcFactory.

```
bool CWavTgtFactory::Init(IAudioPluginMgr *pMgr, IStatus **ppStatus)
{
  if(ppStatus)
    *ppStatus = NULL;

  if(g_pPluginMgr)
  {
    ASSERT(g_pPluginMgr == pMgr);
  }
  else
    g_pPluginMgr = pMgr;

  // success
  return true;
}
```

Done does nothing apart from returning true.

```
bool CWavTgtFactory::Done()
{
  return true;
}
```

CreateURLAudioTarget creates a CWavTgt instance.

```
bool CWavTgtFactory::CreateURLAudioTarget(IUnknown**        ppTgt,
                                          const SWavFormat& formatSrc,
                                          IStatus**         ppStatus)
{
  if(ppStatus)
  {
    *ppStatus = NULL;
  }

  *ppTgt = static_cast<IAggregatable*>
```

```
            (new CWavTgt(formatSrc, (IAudioComponent*)this));
  if(*ppTgt)
  {
    (*ppTgt)->AddRef();
  }

  return true;
}
```

EditSettings creates an instance of the SWavFormat struct and provides this as parameter for the constructor of the CWavSettingsDlg class, wich represents the wave format settings dialog.

```
bool CWavTgtFactory::EditSettings(IUnknown **ppTgt, int iCount)
{
  SWavFormat** ppDlgData = new SWavFormat*[iCount];

  if(!(ppTgt && iCount && ppDlgData))
  {
    ASSERT(FALSE);
    return false;
  }

  AFX_MANAGE_STATE(AfxGetStaticModuleState());

  for(int i = 0; i < iCount; i++)
  {
    CComQIPtr<IWavTgtSetDlgData> pDlgData = ppTgt[i];

    if(!pDlgData)
    {
      // This is an object of not a valid type because we cannot
      // retrieve our internal interface from it.
      ASSERT(FALSE);
      return false;
    }

    ppDlgData[i] = pDlgData->GetDlgData();
  }

  CWavSettingsDlg dlg(ppDlgData, iCount);

  int iDlgRet = dlg.DoModal();

  delete ppDlgData;

  return (iDlgRet == IDOK);
}
```

The application should first call the CanEditSettings function to determine if this factory can edit settings for items.

```
bool CWavTgtFactory::CanEditSettings()
{
  return true;
}
```

We will only support local files.

```
EURLType CWavTgtFactory::GetSupportedURLTypes()
{
  return URL_LocalFile;
}
```

GetCount returns the number of supported extensions.

```
int CWavTgtFactory::GetCount()
{
  return 2;
}
```

The GetExt implementation is identical to that of CWavSrcFactory.

```
const char* CWavTgtFactory::GetExt(int iNum)
{
  if(0 == iNum)
  {
    return "wav";
  }
  else if(1 == iNum)
  {
    return "wave";
  }

  ASSERT(FALSE);

  return NULL;
}
```

The GetRefCount implementation is the same for CWavEnum, CWavSrcFactory and CWavTgtFactory, it returns the value of the reference counter member variable.

```
long CWavTgtFactory::GetRefCount()
{
  return m_lRef;
}
```

8.16. The Component Enumerator Class

The component enumerator class is required because we have more than one audio component, namely the source and target factories.

8.16.1. WavEnum.h

CWavEnum implements IIdentifiable, IAudioComponent and IComponentEnum.

```
// CStatus and CAggregatable
#include "AudioBase.h"


class CWavEnum : public IIdentifiable, public IAudioComponent, public IComponentEnum
{
// Construction/Destruction
public:
  CWavEnum() : m_lRef(0) {};
  virtual ~CWavEnum() {}

// Operations
public:
  virtual long GetRefCount();

  // IUnknown
  STDMETHODIMP QueryInterface(REFIID riid, void __RPC_FAR* __RPC_FAR* ppObj);
  STDMETHODIMP_(ULONG) AddRef();
  STDMETHODIMP_(ULONG) Release();

  // IIdentifiable
  virtual void GetID(GUID *pGUID);

  // IAudioComponent
  virtual const char* GetName();
  virtual EAudioComponentType GetType();
  virtual bool Init(IAudioPluginMgr* pMgr, IStatus** ppStatus);
  virtual bool Done();

  // IComponentEnum
  virtual int GetCount();
  virtual bool GetComponent(int iNum, IAudioComponent** ppComp);

// Data
protected:
  long    m_lRef;
};
```

8.16.2. WavEnum.cpp

We include the headers for the factories and provide extern declarations for globally defined variables.

```
#include "stdafx.h"
#include "WavEnum.h"

// wave target factory
#include "WavTgtFactory.h"

// wave source factory
#include "WavSrcFactory.h"

// globals, defined in the main module
extern IAudioPluginMgr* g_pPluginMgr;
extern CWavTgtFactory   g_TgtFactory;
extern CWavSrcFactory   g_SrcFactory;
```

Like the Factory classes, this class requires that a GUID is created. Please run GUIDGEN.exe, and select the “DEFINE_GUID{…}” format to create a GUID. Then use the copy button and paste it into your source code!

```
// {D8E76911-E1D2-4560-8E28-F953D6E1BC56}
EXTERN_GUID(COMPID_WavEnum, 0xd8e76911, 0xe1d2, 0x4560, 0x8e, 0x28, 0xf9, 0x53, 0xd6,
0xe1, 0xbc, 0x56);
```

QueryInterface returns either IUnknown, IIdentifiable, IAudioComponent or IComponentEnum. The implementation is required because CWavEnum is not derived from CAggregatable, which creates the implementation for all components that are aggregatable.

```
STDMETHODIMP CWavEnum::QueryInterface(REFIID riid, void __RPC_FAR *__RPC_FAR *ppObj)
{
  if(NULL == ppObj)
  {
    ASSERT(FALSE);

    return E_INVALIDARG;
  }

  *ppObj = NULL;

  if(IsEqualIID(riid, IID_IUnknown))
    *ppObj = static_cast<IUnknown*>(static_cast<IAudioComponent*>(this));
  else
  if(IsEqualIID(riid, IID_IIdentifiable))
    *ppObj = static_cast<IIdentifiable*>(this);
  else
  if(IsEqualIID(riid, IID_IAudioComponent))
    *ppObj = static_cast<IAudioComponent*>(this);
  else
  if(IsEqualIID(riid, IID_IComponentEnum))
```

```
    *ppObj = static_cast<IComponentEnum*>(this);

  if(*ppObj)
    AddRef();

  if (NULL != *ppObj)
  {
    return S_OK;
  }
  else
  {
    return E_NOINTERFACE;
  }
}
```

Increase the reference counter and return the number of references.

```
STDMETHODIMP_(ULONG) CWavEnum::AddRef()
{
  InterlockedIncrement(&m_lRef);

  return m_lRef;
}
```

Decrease the reference counter and return the number of references.

```
STDMETHODIMP_(ULONG) CWavEnum::Release()
{
  InterlockedDecrement(&m_lRef);

  return m_lRef;
}
```

Return the globally unique identifier of the class (GUID).

```
void CWavEnum::GetID(GUID *pGUID)
{
   ASSERT(pGUID);
   if(pGUID)
      memcpy(pGUID, &COMPID_WavEnum, sizeof(GUID));
}
```

GetName returns the name of the component enumerator.

```
const char* CWavEnum::GetName()
{
  return "wav enum";
}
```

The type of this component is ACT_ComponentEnumerator.

```
EAudioComponentType CWavEnum::GetType()
{
  return ACT_ComponentEnumerator;
}
```

The wave enumerator object is the component that is responsible for storing the pointer to the plug-in manager class. This is done during a call to the Init function.

```
bool CWavEnum::Init(IAudioPluginMgr* pMgr, IStatus** ppStatus)
{
  if(ppStatus)
  {
    *ppStatus = NULL;
  }

  // set the global pointer to the Audio Plug-in Manager
  g_pPluginMgr = pMgr;

  return true;
}
```

Just return true in the Done method because no additional cleanup is required.

```
bool CWavEnum::Done()
{
  return true;
}
```

Return the number of supported components.

```
int CWavEnum::GetCount()
{
  // PCM reader, ADPCM reader and PCM writer
  return 3;
}
```

GetComponent returns a pointer to one of the global factory objects, depending on iNum.

```
bool CWavEnum::GetComponent(int iNum, IAudioComponent** ppComp)
{
  *ppComp = NULL;

  if(iNum == 0)
  {
    *ppComp = &g_SrcFactory;
  }
  else if(iNum == 1)
  {
    *ppComp = &g_TgtFactory;
  }
  else if(iNum == 2)
  {
    // not implemented
    //  *ppComp = &g_ADPCMSrcFactory;
  }

  if(*ppComp)
  {
    (*ppComp)->AddRef();
  }
```

```
  if (NULL != *ppComp)
  {
    return true;
  }
  else
  {
    return false;
  }
}
```

Return the value of the reference counter.

```
long CWavEnum::GetRefCount()
{
  return m_lRef;
}
```

8.17. Implementation of exported functions

After the targets, sources, factories and the enumerator have been implemented, the remaining task is to fill the body of the main module .

We have to include the header files for the enumerator and the target factories because those will be created statically and referenced by global variables.

```
#include "stdafx.h"
#include "myWav.h"

// component enumerator
#include "WavEnum.h"

// wave target factory
#include "WavTgtFactory.h"

// wave source factory
#include "WavSrcFactory.h"
```

8.17.1. Global Definitions

Next we define global variables for the Plug-in Manager, the enumerator and the factories.

```
// reference to the Audio Plug-in Manager
// this pointer will be provided during initialization by the
// Audio Plug-in Manager itself.
IAudioPluginMgr* g_pPluginMgr = NULL;


// instances of the component enumerator and the factories
// will be kept alive until the DLL is unloaded
CWavEnum g_enum;
CWavTgtFactory g_TgtFactory;
CWavSrcFactory g_SrcFactory;
```

8.17.2. NERO_PLUGIN_GetPrimaryAudioObject

The Audio Plug-in Manager will call this function during initialization.

```
bool NERO_PLUGIN_GetPrimaryAudioObject(IAudioComponent** pAC)
{
  // the pointer must be different from NULL
  ASSERT(pAC);

  if(NULL == pAC)
  {
    return false;
  }

  // our primary audio object is the component enumerator
```

```
  *pAC = &g_enum;

  if(NULL != *pAC)
  {
    (*pAC)->AddRef();
  }

  return true;
}
```

8.17.3. NERO_PLUGIN_ReadyToFinish

This function will be called when *Nero* is closed.

```
bool NERO_PLUGIN_ReadyToFinish()
{
  return  (g_TgtFactory.GetRefCount() <= 1) && (g_SrcFactory.GetRefCount() <= 1) &&
        (g_enum.GetRefCount() <= 1);
}
```

8.18. Additional Includes In stdafx.h

We also need to make sure that the stdafx.h file includes all required headers, for example to obtain access to Windows multimedia type definitions:

```
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

// Exclude rarely-used stuff from Windows headers
#define VC_EXTRALEAN

// MFC core and standard components
#include <afxwin.h>

// ATL
#include "atlbase.h"

// Windows multimedia
#include "mmsystem.h"
```

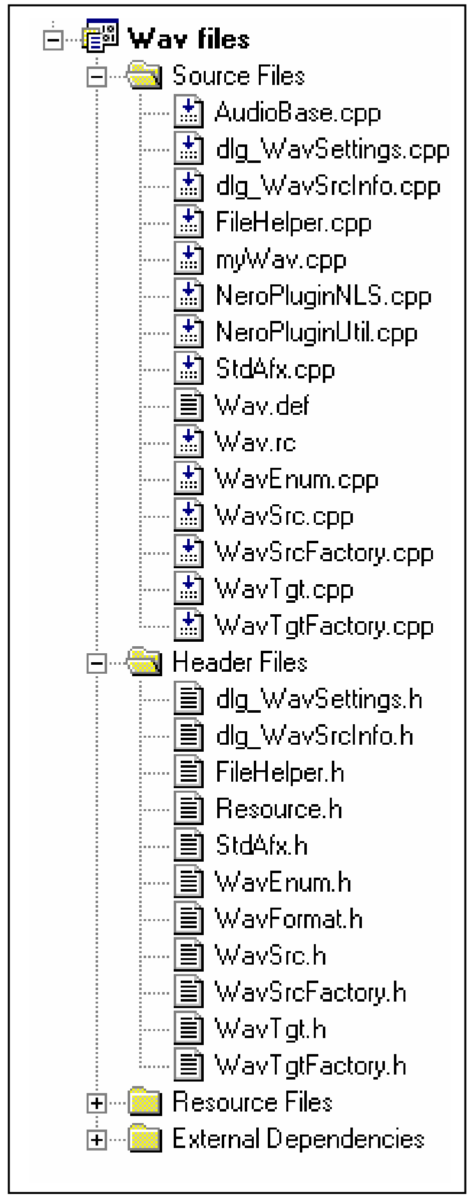
8.19. Defining Exported Functions

Finally we need to determine which functions will be exported. Open “myWav.def” and add the function names as shown here:

```
; Wav.def : Declares the module parameters for the DLL.

LIBRARY      "nxMyWav"
DESCRIPTION  'Wav Windows Dynamic Link Library'

EXPORTS
    NERO_PLUGIN_GetPrimaryAudioObject
    NERO_PLUGIN_ReadyToFinish
```

8.20. Files In The Project

You have to make sure that all required files have been added to the project. The treeview on the right shows the files for this project.



Some of those files can be found in the Common and Include directories of the Audio Plug-in Manager (AudioBase.cpp, NeroPluginNLS.cpp, NeroPluginUtil.cpp).

8.21. Building The Project

*To comply with the naming conventions for custom Plug-ins, make sure that your build target name is prefixed by "nx". On the link tab of your project settings, under "Output file name", you should enter "**nx**MyWav.dll" for both Debug and Release versions.*

Currently, Nero and NeroMIX still support Plug-ins without "nx" prefix, but this will change with one of the next versions!

From the menu select "Build/Build nxMyWav.dll". Visual C++ should build the DLL without any errors or warnings.

Of course the example that we created here is available as Visual C++ project, ready to be built.

Here, we just wanted to give you an idea of what steps have to be followed to come to a functioning Audio Manager Plug-in DLL.

For your owns plug-ins, the main module, the enumerator, the factories will be similar to *MyWav*, while the implementation of the target and source will be completely different – apart from the interfaces they adhere to.

8.22. Copying And Installing The DLL

Use the Windows Explorer to copy nxMyWav.dll to the *ahead\Shared\AudioPlugins* directory.

Note: You might want to automate the copy process by adding a post build step in the project settings:

```
copy Debug\nxMyWav.dll c:\program files\ahead\Shared\AudioPlugins\nxMyWav.dll
```

9. Bibliography

9.1. C++ Programming Books

From the inventor of C++. Not for the faint of heart.

Bjarne Stroustrup: The C++ Programming Language

The author's favourite author. ;-) Read it if you think you know C++ inside and out.

James Coplien: Advanced C++ Programming Styles and Idioms

9.2. C++ Online Resources

Valencia Community College C++ programming course

http://m2tech.net/cppclass/

Intended for C users who want to make the transition to C++

http://www.icce.rug.nl/docs/cplusplus/cplusplus.html

A very good site

http://www.codeproject.com

9.3. COM books

This book has been **the** COM book for quite a while

Don Box: Essential COM

9.4. General CD/CD-ROM Online Resources

Glossary of CD-ROM and DVD technologies

http://www.sigcat.org/resource/gloss697.htm

9.5. Audio CD Online Resources

Digital Audio on CD

http://www.disctronics.co.uk/cdref/cdaudio/cdaudio.htm

10. Listings

10.1. dlg_WavSettings.h

```
/******************************************************************************
|* THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF
|* ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO
|* THE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
|* PARTICULAR PURPOSE.
|*
|* Copyright 2002-2005 Nero AG. All Rights Reserved.
|*-----------------------------------------------------------------------------
|* PROJECT: Nero Plug-in Manager Example
|*
|* FILE: dlg_WavSettings.h
|*
|* PURPOSE: Definition of target item settings editor dialog class.
******************************************************************************/

#if !defined(AFX_DLG_WAVSETTINGS_H__23F58034_73E7_4DEB_B9E5_0ECA1DE88FF3__INCLUDED_)
# define AFX_DLG_WAVSETTINGS_H__23F58034_73E7_4DEB_B9E5_0ECA1DE88FF3__INCLUDED_

#if _MSC_VER > 1000
# pragma once
#endif // _MSC_VER > 1000


// main symbols
#include "resource.h"

// translator class definitions
#include "NeroPluginNLS.h"


struct SWavFormat;

// Target item settings editor dialog class.
class CWavSettingsDlg : public CDialog
{
// Construction
public:
  CWavSettingsDlg(SWavFormat** parItems, int iItemNum);

// Data
private:
  enum {MUL_VAL = 0xffffffff};

  SWavFormat**  m_parItems;
  int           m_iItemNumber;

  CTranslator*  m_pTranslator;

// Dialog Data
  //{{AFX_DATA(CWavSettingsDlg)
  enum { IDD = IDD_WAV_SETTINGS };
  CComboBox m_comboFreq;
  CComboBox m_comboSampleSize;
```

```
  CComboBox m_comboChannels;
  int   m_iChannels;
  int   m_iSampleSize;
  CString m_csFreq;
  //}}AFX_DATA

// Overrides
  // ClassWizard generated virtual function overrides
  //{{AFX_VIRTUAL(CWavSettingsDlg)
  protected:
  virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
  //}}AFX_VIRTUAL

// Implementation
protected:

  // Generated message map functions
  //{{AFX_MSG(CWavSettingsDlg)
  virtual BOOL OnInitDialog();
  virtual void OnOK();
  afx_msg void OnDestroy();
  //}}AFX_MSG
  DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous
line.

#endif // !defined(AFX_DLG_WAVSETTINGS_H__23F58034_73E7_4DEB_B9E5_0ECA1DE88FF3__INCLUDED_)
```

10.2. dlg_WavSettings.cpp

```
/******************************************************************************
|* THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF
|* ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO
|* THE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
|* PARTICULAR PURPOSE.
|*
|* Copyright 2002-2005 Nero AG. All Rights Reserved.
|*-----------------------------------------------------------------------------
|* PROJECT: Nero Plug-in Manager Example
|*
|* FILE: dlg_WavSettings.cpp
|*
|* PURPOSE: Implementation of the target item settings editor dialog.
******************************************************************************/


#include "stdafx.h"
#include "dlg_WavSettings.h"
#include "AudioPluginEnv.h"


#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
```

```
extern IAudioPluginMgr* g_pPluginMgr;

// This class doesn't have a default constructor, it must be initialized
// by this one, it receives an array of item handles, font and the language.
CWavSettingsDlg::CWavSettingsDlg(SWavFormat** parItems, int iItemNum)
: CDialog(CWavSettingsDlg::IDD, NULL),
  m_parItems    (parItems),
  m_iItemNumber (iItemNum),
  m_pTranslator (NULL)
{
  //{{AFX_DATA_INIT(CWavSettingsDlg)
  m_iChannels = -1;
  m_iSampleSize = -1;
  m_csFreq = _T("");
  //}}AFX_DATA_INIT
}

void CWavSettingsDlg::DoDataExchange(CDataExchange* pDX)
{
  CDialog::DoDataExchange(pDX);
  //{{AFX_DATA_MAP(CWavSettingsDlg)
  DDX_Control(pDX, IDCOMBO_FREQ, m_comboFreq);
  DDX_Control(pDX, IDCOMBO_SAMPLE_SIZE, m_comboSampleSize);
  DDX_Control(pDX, IDCOMBO_CHANNELS, m_comboChannels);
  DDX_CBIndex(pDX, IDCOMBO_CHANNELS, m_iChannels);
  DDX_CBIndex(pDX, IDCOMBO_SAMPLE_SIZE, m_iSampleSize);
  DDX_CBString(pDX, IDCOMBO_FREQ, m_csFreq);
  //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CWavSettingsDlg, CDialog)
  //{{AFX_MSG_MAP(CWavSettingsDlg)
  ON_WM_DESTROY()
  //}}AFX_MSG_MAP
END_MESSAGE_MAP()

// Merges up the fields from received items and initializes controls.
BOOL CWavSettingsDlg::OnInitDialog()
{
  CDialog::OnInitDialog();

  if(!(m_parItems && m_iItemNumber))
  {
    ASSERT(FALSE);

    EndDialog(IDCANCEL);

    return TRUE;
  }

  SWavFormat mi;
  memset(&mi, 0xff, sizeof(SWavFormat));

  // Finding equal and different fields in
  // items received. Fields which are different
  // in mi will have value of MUL_VAL

  for(int iCurField = 0; iCurField < (sizeof(SWavFormat) / sizeof(DWORD));
```

```
  iCurField++)
{
  DWORD dwCurValue = MUL_VAL;

  for(int iCurItem = 0; iCurItem < m_iItemNumber; iCurItem++)
  {
    DWORD* pItem = (DWORD*)m_parItems[iCurItem];

    if(pItem)
    {
      DWORD dwCurItemField = pItem[iCurField];

      if (dwCurValue == MUL_VAL)
        dwCurValue = dwCurItemField;
      else
      {
        if (dwCurValue != dwCurItemField)
          break;
      }
    }
    else
    {
      ASSERT(FALSE);
    }
  }

  // all the items have the same value
  if (iCurItem == m_iItemNumber)
    ((DWORD*)&mi)[iCurField] = dwCurValue;
}

// Initializing controls from mi
// Filling the frequency combo

int iarSampleRates[]  =
{
  6000, 8000, 11025,  12000,  16000,  22050,
  32000,  32075,  36000,  44100,  48000
};

for(int i = 0; i < sizeof(iarSampleRates) / sizeof(int); i++)
{
  int iValue = iarSampleRates[i];

  CString csTemp;
  csTemp.Format("%d", iValue);

  m_comboFreq.SetItemData(m_comboFreq.AddString(csTemp), iValue);
}

if(mi.m_iSamplesPerSecond != MUL_VAL)
  m_csFreq.Format("%d", mi.m_iSamplesPerSecond);

int iarSampleSizes[] =
{
  BITS_PER_BYTE,
  BITS_PER_WORD,
  BITS_PER_3BYTE,
  BITS_PER_DWORD
```

```
  };

  for(i = 0; i < sizeof(iarSampleSizes) / sizeof(int); i++)
  {
    m_comboSampleSize.SetItemData(i, iarSampleSizes[i]);

    if(mi.m_iBitsPerSample == iarSampleSizes[i])
      m_iSampleSize = i;
  }

  int iarChannelsNumber[] =
  {
    1,
    2
  };

  for(i = 0; i < sizeof(iarChannelsNumber) / sizeof(int); i++)
  {
    m_comboChannels.SetItemData(i, iarChannelsNumber[i]);

    if (mi.m_iChannels == iarChannelsNumber[i])
      m_iChannels = i;
  }

  UpdateData(FALSE);

  CComQIPtr<ILanguage> pLang = g_pPluginMgr;

  ASSERT(pLang);
  if(pLang)
  {
    m_pTranslator = CreateTranslator(IDR_PLUGIN_NLSDATA,
                      pLang->GetLanguage());
    m_pTranslator->TranslateWindow(m_hWnd);
  }

  return TRUE;
}

// Stores settings from controls in items and closes the dialog
void CWavSettingsDlg::OnOK()
{
  UpdateData(TRUE);

  DWORD dwarDataFromScr[sizeof(SWavFormat) / sizeof(DWORD)] =
  {
    m_csFreq.IsEmpty()? MUL_VAL: atoi(m_csFreq),
    m_comboSampleSize.GetItemData(m_iSampleSize),
    m_comboChannels.GetItemData(m_iChannels)
  };

  for(int iCurItem = 0; iCurItem < m_iItemNumber; iCurItem++)
  {
    DWORD* pItem = (DWORD*)m_parItems[iCurItem];

    for(int iField = 0, iFieldNum = sizeof(SWavFormat) / sizeof(DWORD);
      iField < iFieldNum; iField++)
    {
      DWORD dwField = dwarDataFromScr[iField];
```

```
      if(dwField != MUL_VAL)
        pItem[iField] = dwField;
    }
  }

  CDialog::OnOK();
}

// Destroys the translator after the dialog has been closed
void CWavSettingsDlg::OnDestroy()
{
  CDialog::OnDestroy();
  DestroyTranslator(m_pTranslator);
}
```

10.3. dlg_WavSrcInfo.h

```
/******************************************************************************
|* THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF
|* ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO
|* THE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
|* PARTICULAR PURPOSE.
|*
|* Copyright 2002-2005 Nero AG. All Rights Reserved.
|*----------------------------------------------------------------------------
|* PROJECT: Nero Plug-in Manager Example
|*
|* FILE: dlg_WavSrcInfo.h
|*
|* PURPOSE: Declaration of the source info dialog.
******************************************************************************/


#if !defined(AFX_DLG_WAVSRCINFO_H__E95FB7B0_430A_45E6_8970_876A7EBC49A9__INCLUDED_)
# define AFX_DLG_WAVSRCINFO_H__E95FB7B0_430A_45E6_8970_876A7EBC49A9__INCLUDED_

#if _MSC_VER > 1000
# pragma once
#endif // _MSC_VER > 1000


// main symbols
#include "resource.h"

// translator class definitions
#include "NeroPluginNLS.h"

// interface definitions
#include "AudioPluginEnv.h"


class CWavSrc;

class CWavSrcInfoDlg : public CDialog
{
// Construction
public:
```

```
  CWavSrcInfoDlg(CWavSrc* pSource);

// Operations
protected:

  void  SetModified(bool b);
  void  UpdateModified();

// Data
protected:

  CWavSrc*                            m_pSource;
  CComQIPtr<ISrcInfoViewerEditor>     m_pInfo;
  bool                                m_bModified;
  int                                 m_iCustBtnCount;
  CPtrArray                           m_ptrarUserButtons;
  CTranslator*                        m_pTranslator;

// Dialog Data
  //{{AFX_DATA(CWavSrcInfoDlg)
  enum { IDD = IDD_INFO };
  CButton m_btnStub;
  CButton m_btnOK;
  CEdit m_editDataFormat;
  CString m_csPath;
  CString m_csFileSize;
  CString m_csDateTime;
  CString m_csArtist;
  CString m_csDataFormat;
  CString m_csTitle;
  //}}AFX_DATA

// Overrides
  // ClassWizard generated virtual function overrides
  //{{AFX_VIRTUAL(CWavSrcInfoDlg)
  protected:
  virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
  virtual BOOL OnCommand(WPARAM wParam, LPARAM lParam);
  //}}AFX_VIRTUAL

// Implementation
protected:

  // Generated message map functions
  //{{AFX_MSG(CWavSrcInfoDlg)
  virtual BOOL OnInitDialog();
  afx_msg void OnSelendokGenre();
  afx_msg void OnChangeField();
  afx_msg void OnDestroy();
  //}}AFX_MSG
  DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous
line.

#endif // !defined(AFX_DLG_WAVSRCINFO_H__E95FB7B0_430A_45E6_8970_876A7EBC49A9__INCLUDED_)
```

10.4. dlg_WavSrcInfo.cpp

```
/******************************************************************************
|* THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF
|* ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO
|* THE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
|* PARTICULAR PURPOSE.
|*
|* Copyright 2002-2005 Nero AG. All Rights Reserved.
|*-----------------------------------------------------------------------------
|* PROJECT: Nero Plug-in Manager Example
|*
|* FILE: dlg_WavSrcInfos.cpp
|*
|* PURPOSE: Implementation of the source info dialog.
******************************************************************************/


#include "stdafx.h"
#include "dlg_WavSrcInfo.h"

#include "WavSrc.h"


#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

extern IAudioPluginMgr* g_pPluginMgr;


// CWavSrcInfoDlg

CWavSrcInfoDlg::CWavSrcInfoDlg(CWavSrc* pSource)
: CDialog     (CWavSrcInfoDlg::IDD, NULL),
  m_pSource   (pSource),
  m_bModified   (0),
  m_iCustBtnCount (0)
{
  if(pSource)
  {
    m_pInfo = static_cast<IAggregatable*>(pSource);

    ASSERT(m_pInfo);
  }

  //{{AFX_DATA_INIT(CWavSrcInfoDlg)
  m_csPath = _T("");
  m_csFileSize = _T("");
  m_csDateTime = _T("");
  m_csArtist = _T("");
  m_csDataFormat = _T("");
  m_csTitle = _T("");
  //}}AFX_DATA_INIT
}
```

```
void CWavSrcInfoDlg::DoDataExchange(CDataExchange* pDX)
{
  CDialog::DoDataExchange(pDX);
  //{{AFX_DATA_MAP(CWavSrcInfoDlg)
  DDX_Control(pDX, IDBTN_STUB, m_btnStub);
  DDX_Control(pDX, IDOK, m_btnOK);
  DDX_Control(pDX, IDEDIT_DATA_FORMAT, m_editDataFormat);
  DDX_Text(pDX, IDEDIT_PATH, m_csPath);
  DDX_Text(pDX, IDEDIT_FILE_SIZE, m_csFileSize);
  DDX_Text(pDX, IDEDIT_DATETIME, m_csDateTime);
  DDX_Text(pDX, IDEDIT_ARTIST, m_csArtist);
  DDX_Text(pDX, IDEDIT_DATA_FORMAT, m_csDataFormat);
  DDX_Text(pDX, IDEDIT_TITLE, m_csTitle);
  //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CWavSrcInfoDlg, CDialog)
  //{{AFX_MSG_MAP(CWavSrcInfoDlg)
  ON_CBN_SELENDOK(IDCOMBO_GENRE, OnSelendokGenre)
  ON_EN_CHANGE(IDEDIT_ARTIST, OnChangeField)
  ON_EN_CHANGE(IDEDIT_DATETIME, OnChangeField)
  ON_EN_CHANGE(IDEDIT_TITLE, OnChangeField)
  ON_WM_DESTROY()
  //}}AFX_MSG_MAP
END_MESSAGE_MAP()

//#define EXTERNAL_GENRE
#include "GenreStrings.inc"



BOOL CWavSrcInfoDlg::OnInitDialog()
{
  CDialog::OnInitDialog();

  // Obtainig the language information from the plug-in manager in order to
  // translate this window.
  CComQIPtr<ILanguage> pLang = g_pPluginMgr;

  ASSERT(pLang);
  if(pLang)
  {
    m_pTranslator = CreateTranslator(IDR_PLUGIN_NLSDATA, pLang->GetLanguage());
    m_pTranslator->TranslateWindow(m_hWnd);
  }

  if(!(m_pSource && m_pInfo))
  {
    ASSERT(FALSE);

    EndDialog(IDCANCEL);

    return TRUE;
  }
  // file path
  CComQIPtr<IURLHolder> pURLHolder = m_pSource;

  if(pURLHolder)
  {
```

```
    m_csPath = pURLHolder->GetURL();
  }

  // file size
  DWORD dwSize = GetFileSize(m_pSource->GetFileHandle(), NULL);

  m_csFileSize.Format("%u", dwSize);

  for(int i = m_csFileSize.GetLength() - 3; i > 0; i -= 3)
  {
    m_csFileSize.Insert(i, ".");
  }

  // date / time
  CFile file;
  file.GetFilePath();

  FILETIME ft;
  GetFileTime(m_pSource->GetFileHandle(), NULL, NULL, &ft);

  CTime time(ft);
  m_csDateTime = time.Format("%d.%m.%Y  %H:%M");

  // additional info values
  const char* szValue = NULL;

  CComQIPtr<IInfoReader> pInfoReader = m_pInfo;

  if(pInfoReader)
  {
    szValue = pInfoReader->GetArtist();
    if(szValue)
    {
      m_csArtist = szValue;
    }

    szValue = pInfoReader->GetTitle();
    if(szValue)
    {
      m_csTitle = szValue;
    }
  }

  // data format
  // 80 is the rough position of the second column
  m_editDataFormat.SetTabStops(80);

  m_csDataFormat.Format("Format:\tPCM\r\n"
              "Samplerate:\t%d\r\nChannels:\t%d\r\n"
              "BitPerSample:\t%d",
              m_pSource->GetWavSrcFormat().wf.nSamplesPerSec,
              m_pSource->GetWavSrcFormat().wf.nChannels,
              m_pSource->GetWavSrcFormat().wBitsPerSample);

  // application buttons
  CComPtr<ISrcInfoCallback> pCB;

  if(m_pInfo->GetCallback(&pCB) && pCB)
  {
```

```
    CRect rOK,
        rSave,
        rThisClient;

    GetClientRect(&rThisClient);

    m_btnOK.GetWindowRect(&rOK);
    m_btnStub.GetWindowRect(&rSave);
    ScreenToClient(&rOK);
    ScreenToClient(&rSave);

    int iDiff = rSave.top - rOK.top;

    m_iCustBtnCount = pCB->GetCustomControlCount();

    int iLastButtonBottom = 0;

    for(int i = 0; i < m_iCustBtnCount; i++)
    {
      CComPtr<IControl> pControl;
      if(!pCB->GetControl(i, &pControl) ||
          strcmpi(pControl->GetClassName(), "BUTTON"))
        continue;

      int iHInc = iDiff * i;

      CRect rNew(rOK.left, rSave.top + iHInc,
            rOK.right, rSave.bottom + iHInc);

      iLastButtonBottom = rNew.bottom;

      CButton* pBtn = new CButton;
      pBtn->Create(pControl->GetTitle(),
              GetWindowLong(m_btnStub.m_hWnd, GWL_STYLE),
              rNew, this, IDBTN_CUSTOM_FIRST +
              pControl->GetID());

      pBtn->SetFont(m_btnOK.GetFont());

      pBtn->ShowWindow(SW_SHOW);

      m_ptrarUserButtons.Add(pBtn);
    }

    // Resizind the dialog in order our new buttons to fit in it.
    iLastButtonBottom += (iDiff - rOK.Height());

    if(iLastButtonBottom > rThisClient.Height())
    {
      CRect r(0, 0, rThisClient.Width(), iLastButtonBottom);

      AdjustWindowRect(&r, GetWindowLong(m_hWnd, GWL_STYLE), FALSE);

      SetWindowPos(NULL, 0, 0, r.Width(), r.Height(),
              SWP_NOZORDER|SWP_NOMOVE);
    }
  }

  // update and validate
```

```
  UpdateData(FALSE);

  UpdateModified();

  return TRUE;
}

void CWavSrcInfoDlg::SetModified(bool b)
{
  bool bModifiedChanged = (b == m_bModified);

  m_bModified = b;

  UpdateModified();

  CComPtr<ISrcInfoCallback> pCB;

  if(m_pSource->GetCallback(&pCB) && pCB)
  {
    pCB->OnModified(b);
  }
}

void CWavSrcInfoDlg::UpdateModified()
{
//  m_btnSaveToID3Tag.EnableWindow(m_bModified);
}

void CWavSrcInfoDlg::OnChangeField()
{
  SetModified(true);
}

void CWavSrcInfoDlg::OnSelendokGenre()
{
  SetModified(true);
}

BOOL CWavSrcInfoDlg::OnCommand(WPARAM wParam, LPARAM lParam)
{
  int iID = LOWORD(wParam) - IDBTN_CUSTOM_FIRST;

  if(iID >= 0 && iID < m_iCustBtnCount)
  {
    CComPtr<ISrcInfoCallback> pCB;
    if(m_pSource->GetCallback(&pCB) && pCB)
    {
      pCB->OnCustomButton(iID);
    }
    else
    {
      // this must be not NULL, otherwise where from did
      // we get our custom buttons?
      ASSERT(pCB);
    }
    return TRUE;
  }

  return CDialog::OnCommand(wParam, lParam);
```

```
}

// Destroys the translator after the dialog has been closed
void CWavSrcInfoDlg::OnDestroy()
{
  CDialog::OnDestroy();

  for(int i = 0; i < m_ptrarUserButtons.GetSize(); i++)
  {
    delete ((CButton*)m_ptrarUserButtons[i]);
  }

  DestroyTranslator(m_pTranslator);
}
```

10.5. FileHelper.h

```
/******************************************************************************
|* THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF
|* ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO
|* THE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
|* PARTICULAR PURPOSE.
|*
|* Copyright 2002-2005 Nero AG. All Rights Reserved.
|*-----------------------------------------------------------------------------
|* PROJECT: Nero Plug-in Manager Example
|*
|* FILE: FileHelper.h
|*
|* PURPOSE: Declaration of helper functions for file access
******************************************************************************/

#ifndef _FILE_HELPER_
# define _FILE_HELPER_

#if _MSC_VER > 1000
# pragma once
#endif // _MSC_VER > 1000


// Returns the current position in the file.
DWORD GetFilePointer(HANDLE hFile);


// Reads a DWORD from a file, if it's not possible throws an exception. This
// allows to scan the file with multiple read without checking after every read.
void  ReadDWORD(void* pObject, HANDLE hFile);


// Reads a WORD from a file, if it's not possible throws an exception. This
// allows to scan the file with multiple read without checking after every read.
void  ReadWORD(void* pObject, HANDLE hFile);


// Reads a DWORD from the file and aligns it to a 2-byte border since the chunk
// length must be aligned to word size.
DWORD ReadChunkLen(HANDLE hFile);
```

```
// Writes a byte in the file.
void  WriteZeroByte(HANDLE hFile);


#endif // _FILE_HELPER_
```

10.6. FileHelper.cpp

```
/******************************************************************************
|* THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF
|* ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO
|* THE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
|* PARTICULAR PURPOSE.
|*
|* Copyright 2002-2005 Nero AG. All Rights Reserved.
|*-----------------------------------------------------------------------------
|* PROJECT: Nero Plug-in Manager Example
|*
|* FILE: FileHelper.cpp
|*
|* PURPOSE: Implementation of helper functions for file access
******************************************************************************/

#include "stdafx.h"

#include "FileHelper.h"

#ifdef _DEBUG
# define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif


// Returns the current position in the file.

DWORD GetFilePointer(HANDLE hFile)
{
  ASSERT(hFile != INVALID_HANDLE_VALUE);

  return SetFilePointer(hFile, 0, 0, FILE_CURRENT);
}


// Reads a DWORD from a file, if it's not possible throws an exception. This
// allows to scan the file with multiple read without checking after every read.

void ReadDWORD(void* pObject, HANDLE hFile)
{
  if(!(pObject && hFile))
  {
    ASSERT(FALSE);

    throw FALSE;
  }

  DWORD dwRead = 0;
```

```
  if (!(ReadFile(hFile, pObject, sizeof(DWORD), &dwRead, NULL) &&
    dwRead == sizeof(DWORD)))
  {
    throw FALSE;
  }
}


// Reads a WORD from a file, if it's not possible throws an exception. This
// allows to scan the file with multiple read without checking after every read.

void ReadWORD(void* pObject, HANDLE hFile)
{
  if(!(pObject && hFile))
  {
    ASSERT(FALSE);

    throw FALSE;
  }

  DWORD dwRead = 0;

  if (!(ReadFile(hFile, pObject, sizeof(WORD), &dwRead, NULL) &&
    dwRead == sizeof(WORD)))
  {
    throw FALSE;
  }
}


// Reads a DWORD from the file and aligns it to a 2-byte border since the chunk
// length must be aligned to word size.

DWORD ReadChunkLen(HANDLE hFile)
{
  ASSERT(hFile);

  DWORD dwLen = 0;

  ReadDWORD(&dwLen, hFile);

  if (dwLen % 2)
  {
    dwLen++;
  }

  return dwLen;
}


// Writes a byte in the file.

void WriteZeroByte(HANDLE hFile)
{
  if(hFile)
  {
    BYTE  bt      = 0;
    DWORD dwWritten = 0;
```

```
    WriteFile(hFile, &bt, 1, &dwWritten, NULL);
  }
  else
  {
    ASSERT(FALSE);
  }
}
```

10.7. myWav.h

```
/******************************************************************************
|* THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF
|* ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO
|* THE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
|* PARTICULAR PURPOSE.
|*
|* Copyright 2002-2005 Nero AG. All Rights Reserved.
|*-----------------------------------------------------------------------------
|* PROJECT: Nero Plug-in Manager Example
|*
|* FILE: myWav.h
|*
|* PURPOSE: CWavApp declaration.
******************************************************************************/

#if !defined(AFX_MYWAV_H__E5459AD3_C7F4_11D6_BA0E_00A0D2171523__INCLUDED_)
#define AFX_MYWAV_H__E5459AD3_C7F4_11D6_BA0E_00A0D2171523__INCLUDED_


#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#ifndef __AFXWIN_H__
    #error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"            // main symbols

// Application class. Stores and retrieves last parameters of settings editor
// dialog in registry.
class CWavApp : public CWinApp
{
public:
    CWavApp();

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CWavApp)
    public:
    //}}AFX_VIRTUAL

    //{{AFX_MSG(CWavApp)
            // NOTE - the ClassWizard will add and remove member functions here.
            //    DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
```

```
};


//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before the previous
line.

#endif // !defined(AFX_MYWAV_H__E5459AD3_C7F4_11D6_BA0E_00A0D2171523__INCLUDED_)
```

10.8. myWav.cpp

```
/******************************************************************************
|* THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF
|* ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO
|* THE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
|* PARTICULAR PURPOSE.
|*
|* Copyright 2002-2005 Nero AG. All Rights Reserved.
|*-----------------------------------------------------------------------------
|* PROJECT: Nero Plug-in Manager Example
|*
|* FILE: myWav.cpp
|*
|* PURPOSE: Global variables and implementation of the published DLL functions.
******************************************************************************/


#include "stdafx.h"
#include "myWav.h"

// component enumerator
#include "WavEnum.h"

// wave target factory
#include "WavTgtFactory.h"

// wave source factory
#include "WavSrcFactory.h"

#ifdef _DEBUG
# define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

BEGIN_MESSAGE_MAP(CWavApp, CWinApp)
    //{{AFX_MSG_MAP(CWavApp)
            // NOTE - the ClassWizard will add and remove mapping macros here.
            //    DO NOT EDIT what you see in these blocks of generated code!
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

CWavApp::CWavApp() {}

CWavApp theApp;

// Global data

// reference to the Audio Plug-in Manager
```

```
// this pointer will be provided during initialization by the
// Audio Plug-in Manager itself.
IAudioPluginMgr* g_pPluginMgr = NULL;


// instances of the component enumerator and the factories
// will be kept alive until the DLL is unloaded
CWavEnum g_enum;
CWavTgtFactory g_TgtFactory;
CWavSrcFactory g_SrcFactory;


// The Audio Plug-in Manager will call this function during initialization
bool NERO_PLUGIN_GetPrimaryAudioObject(IAudioComponent** pAC)
{
  // the pointer must be different from NULL
  ASSERT(pAC);

  if(NULL == pAC)
  {
    return false;
  }

  // our primary audio object is the component enumerator
  *pAC = &g_enum;

  if(NULL != *pAC)
  {
    (*pAC)->AddRef();
  }

  return true;
}

// This function will be called when the Nero application is closed
bool NERO_PLUGIN_ReadyToFinish()
{
  return  (g_TgtFactory.GetRefCount() <= 1) && (g_SrcFactory.GetRefCount() <= 1) &&
        (g_enum.GetRefCount() <= 1);
}
```

10.9. WavEnum.h

```
/******************************************************************************
|* THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF
|* ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO
|* THE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
|* PARTICULAR PURPOSE.
|*
|* Copyright 2002-2005 Nero AG. All Rights Reserved.
|*-----------------------------------------------------------------------------
|* PROJECT: Nero Plug-in Manager Example
|*
|* FILE: WavEnum.h
|*
|* PURPOSE: Declaration file for the factory enumerator class.
******************************************************************************/

#ifndef _WAV_ENUM_
# define _WAV_ENUM_

#if _MSC_VER > 1000
# pragma once
#endif // _MSC_VER > 1000


// CStatus and CAggregatable
#include "AudioBase.h"


class CWavEnum : public IIdentifiable, public IAudioComponent, public IComponentEnum
{
// Construction/Destruction
public:
  CWavEnum() : m_lRef(0) {};
  virtual ~CWavEnum() {}

// Operations
public:
  virtual long GetRefCount();

  // IUnknown
  STDMETHODIMP QueryInterface(REFIID riid, void __RPC_FAR* __RPC_FAR* ppObj);
  STDMETHODIMP_(ULONG) AddRef();
  STDMETHODIMP_(ULONG) Release();

  // IIdentifiable
  virtual void GetID(GUID *pGUID);

  // IAudioComponent
  virtual const char* GetName();
  virtual EAudioComponentType GetType();
  virtual bool Init(IAudioPluginMgr* pMgr, IStatus** ppStatus);
  virtual bool Done();

  // IComponentEnum
  virtual int GetCount();
  virtual bool GetComponent(int iNum, IAudioComponent** ppComp);
```

```
// Data
protected:
  long    m_lRef;
};


#endif // _WAV_ENUM_
```

10.10. WavEnum.cpp

```
/******************************************************************************
|* THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF
|* ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO
|* THE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
|* PARTICULAR PURPOSE.
|*
|* Copyright 2002-2005 Nero AG. All Rights Reserved.
|*-----------------------------------------------------------------------------
|* PROJECT: Nero Plug-in Manager Example
|*
|* FILE: WavEnum.cpp
|*
|* PURPOSE: Implementation of the factory enumerator class
******************************************************************************/


#include "stdafx.h"
#include "WavEnum.h"

// wave target factory
#include "WavTgtFactory.h"

// wave source factory
#include "WavSrcFactory.h"

// globals, defined in the main module
extern IAudioPluginMgr* g_pPluginMgr;
extern CWavTgtFactory   g_TgtFactory;
extern CWavSrcFactory   g_SrcFactory;

// {D8E76911-E1D2-4560-8E28-F953D6E1BC56}
EXTERN_GUID(COMPID_WavEnum, 0xd8e76911, 0xe1d2, 0x4560, 0x8e, 0x28, 0xf9, 0x53, 0xd6, 0xe1,
0xbc, 0x56);

// QueryInterface returns either IUnknown, IAudioComponent or IComponentEnum
STDMETHODIMP CWavEnum::QueryInterface(REFIID riid, void __RPC_FAR *__RPC_FAR *ppObj)
{
  // ppObj must not be NULL
  if(NULL == ppObj)
  {
    ASSERT(FALSE);

    return E_INVALIDARG;
  }

  *ppObj = NULL;

  if(IsEqualIID(riid, IID_IUnknown))
```

```
    *ppObj = static_cast<IUnknown*>(static_cast<IAudioComponent*>(this));
  else
  if(IsEqualIID(riid, IID_IIdentifiable))
    *ppObj = static_cast<IIdentifiable*>(this);
  else
  if(IsEqualIID(riid, IID_IAudioComponent))
    *ppObj = static_cast<IAudioComponent*>(this);
  else
  if(IsEqualIID(riid, IID_IComponentEnum))
    *ppObj = static_cast<IComponentEnum*>(this);

  if(*ppObj)
    AddRef();

  if (NULL != *ppObj)
  {
    return S_OK;
  }
  else
  {
    return E_NOINTERFACE;
  }
}

// Increase the reference counter and return the number of references
STDMETHODIMP_(ULONG) CWavEnum::AddRef()
{
  InterlockedIncrement(&m_lRef);

  return m_lRef;
}

// Decrease the reference counter and return the number of references
STDMETHODIMP_(ULONG) CWavEnum::Release()
{
  InterlockedDecrement(&m_lRef);

  return m_lRef;
}

void CWavEnum::GetID(GUID *pGUID)
{
   ASSERT(pGUID);
   if(pGUID)
      memcpy(pGUID, &COMPID_WavEnum, sizeof(GUID));
}

// IAudioComponent
const char* CWavEnum::GetName()
{
  return "wav enum";
}

EAudioComponentType CWavEnum::GetType()
{
  return ACT_ComponentEnumerator;
}

bool CWavEnum::Init(IAudioPluginMgr* pMgr, IStatus** ppStatus)
```

```
{
  if(ppStatus)
  {
    *ppStatus = NULL;
  }

  // set the global pointer to the Audio Plug-in Manager
  g_pPluginMgr = pMgr;

  return true;
}

bool CWavEnum::Done()
{
  return true;
}

// IComponentEnum

// Return the number of supported components
int CWavEnum::GetCount()
{
  // PCM reader, ADPCM reader and PCM writer
  return 3;
}

// return a pointer to a factory object
bool CWavEnum::GetComponent(int iNum, IAudioComponent** ppComp)
{
  *ppComp = NULL;

  if(iNum == 0)
  {
    *ppComp = &g_SrcFactory;
  }
  else if(iNum == 1)
  {
    *ppComp = &g_TgtFactory;
  }
  else if(iNum == 2)
  {
    // not implemented
    //  *ppComp = &g_ADPCMSrcFactory;
  }

  if(*ppComp)
  {
    (*ppComp)->AddRef();
  }

  if (NULL != *ppComp)
  {
    return true;
  }
  else
  {
    return false;
  }
}
```

```
long CWavEnum::GetRefCount()
{
  return m_lRef;
}
```

10.11. WavSrc.h

```
/******************************************************************************
|* THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF
|* ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO
|* THE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
|* PARTICULAR PURPOSE.
|*
|* Copyright 2002-2005 Nero AG. All Rights Reserved.
|*-----------------------------------------------------------------------------
|* PROJECT: Nero Plug-in Manager Example
|*
|* FILE: WavTgt.h
|*
|* PURPOSE: Declaration file for the wave target class
******************************************************************************/

#ifndef _WAV_SRC_
# define _WAV_SRC_

#if _MSC_VER > 1000
# pragma once
#endif // _MSC_VER > 1000


// CStatus and CAggregatable
#include "AudioBase.h"

// wave format definitions
#include "WavFormat.h"

// source info dialog
#include "dlg_WavSrcInfo.h"


class CWavSrc : public  CAggregatable,
                public  IURLHolder,
                public  IProcess,
                public  IAudioItem,
                public  IAudioSource,
                public  IAudioRawReader,
                public  ISeekable,
                public  IAudioRawBlockInfo,
                public  ISrcInfoViewerEditor,
                public  IInfoReader,
                public  SWavCommon
{
// Construction/Destruction
public:

  CWavSrc(const char* szURL, IAudioComponent* pCreator, IStatus** ppStatus);
  virtual ~CWavSrc();
```

```
// Operations
public:

  AGGREGATABLE_INTERFACE_MAP_BEGIN
    INTERFACE_ENTRY(IURLHolder)
    INTERFACE_ENTRY(IAudioItem)
    INTERFACE_ENTRY(IProcess)
    INTERFACE_ENTRY(IAudioSource)
    INTERFACE_ENTRY(IAudioRawReader)
    INTERFACE_ENTRY(ISeekable)
    INTERFACE_ENTRY(IAudioRawBlockInfo)
    INTERFACE_ENTRY(ISrcInfoViewerEditor)
    INTERFACE_ENTRY(IInfoReader)
  AGGREGATABLE_INTERFACE_MAP_END


  // IURLHolder
  virtual bool SetURL(const char* szURL, IStatus** ppStatus);
  virtual const char* GetURL();
  virtual EURLType GetType();

  // IProcess
  virtual bool Start(IStatus** ppStatus);
  virtual bool End(IStatus** ppStatus);
  virtual bool IsInProcess();

  // IAudioItem
  virtual bool GetCreator(IAudioComponent** pCreator);

  virtual EAuxFlags GetAuxFlags();

  virtual void SetAuxFlags(EAuxFlags flags);

  // IAudioSource
  virtual SWavFormat GetRawFormat();

  // iBufSize must contain the size in bytes of the buffer pointed by pBuf.
  virtual bool RawRead( BYTE* pBuf, int iBufSize, int* piRead,
                        EAudioRawState &state,
                        IStatus** ppStatus);

  // Returns free-form text string about the item.
  virtual const char * GetInfo();

  // Returns the file duration in milliseconds.
  virtual ULONGLONG GetDuration();

  // Returns RAW data length in bytes.
  virtual ULONGLONG GetRawLen();

  // ISeekable
  virtual bool Seek(ULONGLONG pos, IStatus** ppStatus);
  virtual ULONGLONG GetPos();
  virtual ULONGLONG GetBlockSize();
  virtual ULONGLONG GetDataLength();

  // ISrcInfoViewerEditor
  virtual bool GetCallback(ISrcInfoCallback** ppCB);
```

```
  virtual void SetCallback(ISrcInfoCallback* pNewCallback);
  virtual bool DoModal(IStatus** ppStatus);
  virtual bool CloseModal(IStatus** ppStatus);
  virtual bool SaveDialogToObject(IStatus** ppStatus);
  virtual bool SaveObjectToFile(IStatus** ppStatus);
  virtual bool CanSaveObjectToFile();

  // IInfoReader
  virtual const char* GetTitle();
  virtual const char* GetArtist();
  virtual const char* GetAlbum();
  virtual const char* GetYear();
  virtual const char* GetGenre();

  virtual HRESULT ReadMMIO();

// others
public:
  virtual PCMWAVEFORMAT GetWavSrcFormat();
  virtual HANDLE GetFileHandle();
  bool    IsOK() { return m_bOK; };

// Data
protected:
  CComPtr<ISrcInfoCallback>   m_pCallback;
  CWavSrcInfoDlg*             m_pAdvInfoDlg;

  // the object
  HANDLE  m_hFile;

  bool    m_bRiffChunk;
  bool    m_bWavChunk;

  DWORD   m_dwDataStart;

  // size of RAW in bytes
  DWORD   m_dwDataLen;

  // current position in bytes
  DWORD   m_dwCurPos;

  PCMWAVEFORMAT   m_format;
  ULONGLONG       m_qwDuration;
  int             m_iBlockSize;

  CString         m_csURL;

  // free-form text info
  CString         m_csInfo;

  bool            m_bOK;
  bool            m_bStarted;

  CComPtr<IAudioComponent>  m_pCreator;
};


#endif //_WAV_SRC_
```

10.12. WavSrc.cpp

```
/******************************************************************************
|* THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF
|* ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO
|* THE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
|* PARTICULAR PURPOSE.
|*
|* Copyright 2002-2005 Nero AG. All Rights Reserved.
|*-----------------------------------------------------------------------------
|* PROJECT: Nero Plug-in Manager Example
|*
|* FILE: WavSrc.cpp
|*
|* PURPOSE: Implementation of the wave source class
******************************************************************************/


#include "stdafx.h"
#include "WavSrc.h"

// helper functions for file access
#include "FileHelper.h"

// IURLHolder implementation

bool CWavSrc::SetURL(const char* szURL, IStatus** ppStatus)
{
  RETURN_ERROR(AE_CallNotImplemented);
}

const char* CWavSrc::GetURL()
{
  return m_csURL;
}

EURLType CWavSrc::GetType()
{
  return URL_LocalFile;
}

// IProcess implementation

bool CWavSrc::Start(IStatus** ppStatus)
{
  if(NULL != ppStatus)
  {
    *ppStatus = NULL;
  }

  m_dwCurPos = 0;

  if(SetFilePointer(m_hFile, m_dwDataStart, 0, FILE_BEGIN) !=
                              m_dwDataStart)
  {
    RETURN_ERROR(GetLastError());
  }
```

```
  m_bStarted = true;

  return true;
}

bool CWavSrc::End(IStatus** ppStatus)
{
  if(NULL != ppStatus)
  {
    *ppStatus = NULL;
  }

  m_bStarted = false;

  return true;
}

bool CWavSrc::IsInProcess()
{
  return m_bStarted;
}

// IAudioItem implementation

bool CWavSrc::GetCreator(IAudioComponent** pCreator)
{
  if(!pCreator)
  {
    ASSERT(FALSE);
    return false;
  }

  *pCreator = m_pCreator;

  (*pCreator)->AddRef();

  return true;
}

EAuxFlags CWavSrc::GetAuxFlags()
{
  return SRC_FLAGS;
}

void CWavSrc::SetAuxFlags(EAuxFlags flags)
{
  // Nothing to do here
}

// IAudioSource implementation

SWavFormat CWavSrc::GetRawFormat()
{
  SWavFormat format;

  format.m_iSamplesPerSecond= m_format.wf.nSamplesPerSec;
  format.m_iBitsPerSample       = m_format.wBitsPerSample;
  format.m_iChannels            = m_format.wf.nChannels;
```

```
  return format;
}

// iBufSize must contain the size in bytes of the buffer pointed by pBuf.

bool CWavSrc::RawRead(BYTE* pBuf, int iBufSize, int* piRead,
            EAudioRawState& state,
            IStatus** ppStatus)
{
  if(!(pBuf && iBufSize && piRead))
  {
    RETURN_ERROR(AE_InvalidParameter);
  }

  if(NULL != ppStatus)
  {
    *ppStatus = NULL;
  }

  state = ERS_None;

  if(m_dwCurPos >= m_dwDataLen)
  {
    state = ERS_EndOfFile;

    RETURN_ERROR(AE_EOF);
  }

  DWORD dwRead = 0;

  DWORD dwRemaining = m_dwDataLen - m_dwCurPos;

  if(iBufSize > (int)dwRemaining)
  {
    iBufSize = dwRemaining;
    state = ERS_EndOfFile;
  }

  if (!(ReadFile(m_hFile, pBuf, iBufSize, (DWORD*)piRead, NULL) && *piRead))
  {

    // That means that the file is over or
    // something is wrong, in any case
    // stopping the process

    m_dwCurPos = m_dwDataLen;
  }
  else
  {
    m_dwCurPos += *piRead;
  }

  return true;
}

// Returns free-form text string about the item.

const char* CWavSrc::GetInfo()
```

```
{
  if(m_csInfo.IsEmpty())
  {
    m_csInfo.Format("PCM Wav file. Format: %d,  Channels: %d,"
              " Sample rate: %d, Bits per sample: %d",
              m_format.wf.wFormatTag,
              m_format.wf.nChannels,
              m_format.wf.nSamplesPerSec,
              m_format.wBitsPerSample);

    CString* pcsStrings[] =
    { &m_csArtist, &m_csTitle, &m_csCopyright, &m_csDate, &m_csComments };

    char* szarNames[5] =
    { "Artist", "Title", "Copyright", "Date", "Comments" };

    for(int i = 0; i < 5; i++)
    {
      CString* pcs = pcsStrings[i];

      if (pcs->IsEmpty())
        continue;

      m_csInfo += ",\r\n";

      CString csTemp;
      csTemp.Format("%s: %s", szarNames[i], (LPCTSTR)(*pcs));

      m_csInfo += csTemp;
    }
  }

  return m_csInfo;
}

// Returns the file duration in milliseconds.

ULONGLONG CWavSrc::GetDuration()
{
  return m_qwDuration;
}

// Returns RAW data length in bytes.

ULONGLONG CWavSrc::GetRawLen()
{
  return m_dwDataLen;
}

// ISeekable implementation

bool CWavSrc::Seek(ULONGLONG pos, IStatus** ppStatus)
{
  if(NULL == ppStatus)
  {
    *ppStatus = NULL;
  }

  DWORD dwLastPos = SetFilePointer(m_hFile, 0, 0, FILE_CURRENT);
```

```
  DWORD dwNewPos  = (DWORD)(m_dwDataStart + pos * m_iBlockSize);

  bool bRetCode = false;

  if(SetFilePointer(m_hFile, dwNewPos, 0, FILE_BEGIN) != dwNewPos)
  {
    SetFilePointer(m_hFile, dwLastPos, 0, FILE_BEGIN);
  }
  else
  {
    m_dwCurPos = (DWORD)(pos * m_iBlockSize);

    bRetCode = true;
  }

  if((false == bRetCode) && (NULL != ppStatus))
  {
    *ppStatus = new CStatus(AE_SeekFailed);
  }

  return bRetCode;
}

ULONGLONG CWavSrc::GetPos()
{
  return (m_dwCurPos / m_iBlockSize);
}

ULONGLONG CWavSrc::GetBlockSize()
{
  return m_iBlockSize;
}

ULONGLONG CWavSrc::GetDataLength()
{
  return (m_dwDataLen / m_iBlockSize);
}

// ISrcInfoViewerEditor implementation

bool CWavSrc::GetCallback(ISrcInfoCallback** ppCB)
{
  if(NULL == ppCB)
  {
    ASSERT(FALSE);
    return false;
  }

  *ppCB = m_pCallback;

  if(NULL != *ppCB)
  {
    (*ppCB)->AddRef();
  }

  return true;
}

void CWavSrc::SetCallback(ISrcInfoCallback* pNewCallback)
```

```
{
  m_pCallback = pNewCallback;
}

bool CWavSrc::DoModal(IStatus** ppStatus)
{
  CWavSrcInfoDlg dlg(this);
  m_pAdvInfoDlg = &dlg;
  dlg.DoModal();
  m_pAdvInfoDlg = NULL;

  return true;
}

bool CWavSrc::CloseModal(IStatus** ppStatus)
{
  if(!(m_pAdvInfoDlg && IsWindow(m_pAdvInfoDlg->m_hWnd)))
  {
    return false;
  }

  m_pAdvInfoDlg->EndDialog(IDCANCEL);

  if (0 != IsWindow(m_pAdvInfoDlg->m_hWnd))
  {
    return true;
  }
  else
  {
    return false;
  }
}

bool CWavSrc::SaveDialogToObject(IStatus** ppStatus)
{
  return false;
}

bool CWavSrc::SaveObjectToFile(IStatus** ppStatus)
{
  return false;
}

bool CWavSrc::CanSaveObjectToFile()
{
  return false;
}

// IInfoReader implementation

const char* CWavSrc::GetTitle()
{
  return m_csTitle;
}

const char* CWavSrc::GetArtist()
{
  return m_csArtist;
}
```

```
const char* CWavSrc::GetAlbum()
{
  return NULL;
}

const char* CWavSrc::GetYear()
{
  return NULL;
}

const char* CWavSrc::GetGenre()
{
  return NULL;
}


CWavSrc::CWavSrc(const char* szURL, IAudioComponent* pCreator,
          IStatus** ppStatus)

          : CAggregatable (0),
            m_pCreator    (pCreator),
            m_hFile       (INVALID_HANDLE_VALUE),
            m_bRiffChunk  (FALSE),
            m_bWavChunk   (FALSE),
            m_dwDataStart (0),
            m_dwDataLen   (0),
            m_dwCurPos    (0),
            m_iBlockSize  (0),
            m_bOK         (false),
            m_bStarted    (false)
{
  try
  {
    m_csURL = szURL;

    // CreateFile - create or open an object: Consoles, Communications resources
    //                          Directories (open only), Files etc.
    //
    //  LPCTSTR lpFileName -  file name
    // DWORD dwDesiredAccess - access mode
    // DWORD dwShareMode - share mode
    // LPSECURITY_ATTRIBUTES lpSecurityAttributes - [in] Pointer to a SECURITY_ATTRIBUTES
    //                                              structure that determines whether the
returned
    //                                              handle can be inherited by child
processes.
    //                                              If lpSecurityAttributes is NULL, the
handle cannot
    //                                              be inherited.
    // DWORD dwCreationDisposition - how to create
    // DWORD dwFlagsAndAttributes - file attributes
    // HANDLE hTemplateFile - handle to template file

    m_hFile = CreateFile(szURL, GENERIC_READ, FILE_SHARE_READ,
              NULL, OPEN_EXISTING, FILE_ATTRIBUTE_READONLY, NULL);

    if(INVALID_HANDLE_VALUE == m_hFile )
    {
      if(NULL != ppStatus)
```

```
    {
      *ppStatus = new CStatus(GetLastError());
    }

    throw false;
  }


  // Trying to recognize the file

  DWORD dwRead    = 0,
  dwFileSize  = 0;

  try
  {
    ReadMMIO();
  }
  catch(BOOL )
  {
    // If we're here - something is wrong in this file
    // Sometimes files can contain some wrong information at the end,
    // but if the rest is OK we can accept such files.
  }

  // make sure that we are dealing with a valid file format

  if(false == ((INVALID_HANDLE_VALUE != m_hFile)  &&
              (true == m_bRiffChunk) &&
              (true == m_bWavChunk) &&
              (0 != m_dwDataStart) &&
              (0 != m_dwDataLen)))
  {
    if(NULL != ppStatus)
    {
      *ppStatus = new CStatus(AE_BadFormat);
    }

    throw false;
  }

  // make sure that we are dealing with a valid wave format

  if(false == (((8 == m_format.wBitsPerSample) ||
               (16 == m_format.wBitsPerSample)) &&

              ((1 == m_format.wf.nChannels) ||
               (2 == m_format.wf.nChannels)) &&

              ((m_format.wf.nSamplesPerSec) >= 1000 &&
               (m_format.wf.nSamplesPerSec <= 100000))))
  {
    if(NULL != ppStatus)
    {
      *ppStatus = new CStatus(AE_BadFormat);
    }

    throw false;
  }
```

```
    // Calculating the file duration

    m_iBlockSize = (m_format.wBitsPerSample / BITS_PER_BYTE) *
                        m_format.wf.nChannels;

    m_qwDuration = (ULONGLONG)m_dwDataLen * (ULONGLONG)1000 /
                  m_format.wf.nAvgBytesPerSec;
  }
  catch(bool )
  {
    // Error occured

    return;
  }

  m_bOK = true;
}

CWavSrc::~CWavSrc()
{
  if(m_hFile != INVALID_HANDLE_VALUE)
  {
    CloseHandle(m_hFile);
  }
}

HANDLE CWavSrc::GetFileHandle()
{
  return m_hFile;
}

PCMWAVEFORMAT CWavSrc::GetWavSrcFormat()
{
  return m_format;
}

HRESULT CWavSrc::ReadMMIO()
{
  // Convert the CString URL to LPTSTR

  LPTSTR lpsz = new TCHAR[m_csURL.GetLength()+1];
  _tcscpy(lpsz, m_csURL);


  // MM I/O handle for the WAVE
  HMMIO         hmmio;

  // Open the file with the appropriate multimedia function

  hmmio = mmioOpen(lpsz, NULL, MMIO_ALLOCBUF | MMIO_READ );

  // Free the temporay pointer

  delete lpsz;

  // Make sure that the file could be opened

  if( NULL != hmmio )
  {
```

```
    // Use in opening a WAVE file
    MMCKINFO      ckRiff;

    // Use for subchunks
    MMCKINFO      ck;

    // Search for the first chunk
    if( ( 0 != mmioDescend( hmmio, &ckRiff, NULL, 0 ) ) )
    {
      return E_FAIL;
    }

    // Make sure this is a valid wave file

    if( (ckRiff.ckid != FOURCC_RIFF) || (ckRiff.fccType != mmioFOURCC('W', 'A', 'V', 'E')))
    {
      return E_FAIL;
    }

    // If we got here, we can be sure that there is a riff chunk, and the type is WAVE

    m_bRiffChunk = true;
    m_bWavChunk = true;

    // Search the input file for for the 'fmt ' chunk.

    ck.ckid = mmioFOURCC('f', 'm', 't', ' ');
    if( 0 != mmioDescend( hmmio, &ck, &ckRiff, MMIO_FINDCHUNK))
    {
      return E_FAIL;
    }

    // Expect the 'fmt' chunk to be at least as large as <PCMWAVEFORMAT>;
    // if there are extra parameters at the end, we'll ignore them

    if( ck.cksize < (LONG) sizeof(PCMWAVEFORMAT))
    {
     return E_FAIL;
    }

    // Read the 'fmt ' chunk into m_format

    if( mmioRead( hmmio, (HPSTR) &m_format, sizeof(PCMWAVEFORMAT)) != sizeof(PCMWAVEFORMAT))
    {
      return E_FAIL;
    }

    if( m_format.wf.wFormatTag == WAVE_FORMAT_PCM )
    {
      // Seek to the data

      if( -1 == mmioSeek( hmmio, ckRiff.dwDataOffset + sizeof(FOURCC), SEEK_SET))
      {
          return E_FAIL;
      }

      // Search the input file for the 'data' chunk.

      ck.ckid = mmioFOURCC('d', 'a', 't', 'a');
```

```
      if( 0 != mmioDescend( hmmio, &ck, &ckRiff, MMIO_FINDCHUNK))
      {
        return E_FAIL;
      }

      // store offset and size of the data

      m_dwDataStart = ck.dwDataOffset;
      m_dwDataLen = ck.cksize;
    }
    else
    {
      // Wrong format

      return E_FAIL;
    }

    // Close the file

    mmioClose( hmmio, 0 );

    return S_OK;
  }
  else
  {
    // File could not be openend
    return E_FAIL;
  }
}
```

10.13. WavSrcFactory.h

```
/******************************************************************************
|* THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF
|* ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO
|* THE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
|* PARTICULAR PURPOSE.
|*
|* Copyright 2002-2005 Nero AG. All Rights Reserved.
|*----------------------------------------------------------------------------
|* PROJECT: Nero Plug-in Manager Example
|*
|* FILE: WavSrcFactory.h
|*
|* PURPOSE: Declaration file for the wave sourc factory class.
******************************************************************************/

#ifndef _WAV_SRC_FACTORY_
# define _WAV_SRC_FACTORY_

#if _MSC_VER > 1000
# pragma once
#endif // _MSC_VER > 1000

// CStatus and CAggregatable
#include "AudioBase.h"
```

```
class CWavSrcFactory :  public IIdentifiable,
                        public CAggregatable,
                        public IAudioComponent,
                        public IURLAudioSourceFactory,
                        public IExtEnum,
                                         public IVendorInfo
{
// Construction/Destruction
public:
  CWavSrcFactory()
  : CAggregatable (AGGFLAG_NODELETE)
  {}

  virtual ~CWavSrcFactory() {};

// Operations
public:
  virtual long GetRefCount();

  AGGREGATABLE_INTERFACE_MAP_BEGIN
    INTERFACE_ENTRY(IIdentifiable)
    INTERFACE_ENTRY(IAudioComponent)
    INTERFACE_ENTRY(IURLAudioSourceFactory)
    INTERFACE_ENTRY(IExtEnum)
    INTERFACE_ENTRY(IVendorInfo)
  AGGREGATABLE_INTERFACE_MAP_END

  // IIdentifiable
  void GetID(GUID *pGUID);

  // IAudioComponent
  virtual const char* GetName();
  virtual EAudioComponentType GetType();
  virtual bool Init(IAudioPluginMgr* pMgr, IStatus** ppStatus);
  virtual bool Done();
  virtual bool
  Open( const char*   szURL,
        IUnknown**    ppSrc,
        EAuxFlags     flagsInclude,
        EAuxFlags     flagsExclude,
        IStatus**     ppStatus);

  virtual EAuxFlags GetAuxFlags();

  virtual EURLType GetSupportedURLTypes();

  // IExtEnum
  int GetCount();

  // The returned value can't be stored for later use.
  // The application must copy it.
  const char* GetExt(int iNum);

  // IVendorInfo
  const char * GetVendorName();
  bool CanDisplayAboutBox();
  void DisplayAboutBox();
};
#endif // _WAV_SRC_FACTORY_
```

10.14. WavSrcFactory.cpp

```
/******************************************************************************
|* THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF
|* ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO
|* THE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
|* PARTICULAR PURPOSE.
|*
|* Copyright 2002-2005 Nero AG. All Rights Reserved.
|*-----------------------------------------------------------------------------
|* PROJECT: Nero Plug-in Manager Example
|*
|* FILE: WavSrcFactory.cpp
|*
|* PURPOSE: Implementation of the wave source factory
******************************************************************************/


#include "stdafx.h"

#include "WavSrcFactory.h"

#include "WavSrc.h"


EXTERN_GUID(COMPID_SrcFactory, 0x459215ca, 0x50f5, 0x42cc, 0xbf, 0x51, 0x84, 0x38, 0x11, 0x9d,
0xee, 0x6c);

extern IAudioPluginMgr* g_pPluginMgr;


void CWavSrcFactory::GetID(GUID *pGUID)
{
    ASSERT(pGUID);
    if(pGUID)
    memcpy(pGUID, &COMPID_SrcFactory, sizeof(GUID));
}

const char* CWavSrcFactory::GetName()
{
  return "PCM Wav file";
}

EAudioComponentType CWavSrcFactory::GetType()
{
  return ACT_AudioURLSourceFactory;
}

bool CWavSrcFactory::Init(IAudioPluginMgr* pMgr, IStatus** ppStatus)
{
  if(NULL != ppStatus)
  {
    *ppStatus = NULL;
  }

  if(NULL != g_pPluginMgr)
  {
    ASSERT(g_pPluginMgr == pMgr);
  }
```

```
  else
  {
    g_pPluginMgr = pMgr;
  }

  // success
  return true;
}

bool CWavSrcFactory::Done()
{
  return true;
}

EAuxFlags CWavSrcFactory::GetAuxFlags()
{
  return SRC_FLAGS;
}

EURLType CWavSrcFactory::GetSupportedURLTypes()
{
  return URL_LocalFile;
}

// IExtEnum implementation
int CWavSrcFactory::GetCount()
{
  return 2;
}

// The returned value can't be stored for later use.
// The application must copy it.
const char* CWavSrcFactory::GetExt(int iNum)
{
  if(0 == iNum)
  {
    return "wav";
  }
  else if(1 == iNum)
  {
    return "wave";
  }

  ASSERT(FALSE);

  return NULL;
}

bool CWavSrcFactory::Open(const char* szURL,
                          IUnknown**  ppSrc,
                          EAuxFlags flagsInclude,
                          EAuxFlags flagsExclude,
                          IStatus** ppStatus)
{
  if(ppStatus)
  {
    *ppStatus = NULL;
  }
```

```
  CWavSrc* pSrc = new CWavSrc(szURL, (IAudioComponent*)this, ppStatus);

  bool bOK = pSrc->IsOK();

  if(bOK)
  {
    *ppSrc = static_cast<IAggregatable*>(pSrc);

    (*ppSrc)->AddRef();
  }
  else
  {
    delete pSrc;

    *ppSrc = NULL;
  }

  return bOK;
}

long CWavSrcFactory::GetRefCount()
{
  return m_lRef;
}

const char *CWavSrcFactory::GetVendorName()
{
  return "Example Vendor";
}

bool CWavSrcFactory::CanDisplayAboutBox()
{
  return true;
}

void CWavSrcFactory::DisplayAboutBox()
{
  AfxMessageBox("Nero Example PCM WAV plug-in.");
}
```

10.15. WavTgt.h

```
/******************************************************************************
|* THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF
|* ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO
|* THE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
|* PARTICULAR PURPOSE.
|*
|* Copyright 2002-2005 Nero AG. All Rights Reserved.
|*-----------------------------------------------------------------------------
|* PROJECT: Nero Plug-in Manager Example
|*
|* FILE: WavTgt.h
|*
|* PURPOSE: Declaration file for the wave target class
******************************************************************************/

#ifndef _WAV_TGT_
# define _WAV_TGT_

#if _MSC_VER > 1000
# pragma once
#endif // _MSC_VER > 1000

// CStatus and CAggregatable
#include "AudioBase.h"

// wave format definitions
#include "WavFormat.h"


// Internal interfaces

// {48960729-B5C6-48fb-B6A9-AF3E428F9E2B}
EXTERN_GUID(IID_IWavTgtSetDlgData,
                  0x48960729, 0xb5c6, 0x48fb,
                  0xb6, 0xa9, 0xaf, 0x3e, 0x42, 0x8f, 0x9e, 0x2b);

struct __declspec(uuid("48960729-B5C6-48fb-B6A9-AF3E428F9E2B"))
                              IWavTgtSetDlgData;

interface IWavTgtSetDlgData : public IUnknown
{
  virtual SWavFormat* GetDlgData()  = 0;
};


// CWavTgt - the wave target class

class CWavTgt : public CAggregatable,
                public IURLHolder,
                public IAudioItem,
                public IProcess,
                public IAudioRawWriter,
                public IWavTgtSetDlgData,
                public IInfoWriter,
                public SWavCommon
```

```
{
// Construction/Destruction
public:
  CWavTgt(const SWavFormat &formatSrc, IAudioComponent *pCreator)
  : CAggregatable   (0),
    m_dwWrittenRAWSize  (0),
    m_hFile       (INVALID_HANDLE_VALUE),
    m_flags       (EAuxFlags(0)),
    m_pCreator      (pCreator),
    m_DlgData     (formatSrc),
    m_bEndProcess   (false)
  {};

  virtual ~CWavTgt() {};

// Operations
public:

  AGGREGATABLE_INTERFACE_MAP_BEGIN
    INTERFACE_ENTRY(IAudioItem)
    INTERFACE_ENTRY(IURLHolder)
    INTERFACE_ENTRY(IProcess)
    INTERFACE_ENTRY(IAudioRawWriter)
    INTERFACE_ENTRY(IWavTgtSetDlgData)
    INTERFACE_ENTRY(IInfoWriter)
  AGGREGATABLE_INTERFACE_MAP_END

  // IAudioItem

  virtual bool GetCreator(IAudioComponent** pCreator);

  // Bitmask of auxiliary flags
  virtual EAuxFlags GetAuxFlags();
  virtual void SetAuxFlags(EAuxFlags flags);

  // IURLHolder
  virtual bool SetURL(const char* szURL, IStatus** ppStatus);
  virtual const char* GetURL();
  virtual EURLType GetType();

  // IWavTgtSetDlgData
  SWavFormat* GetDlgData();

  // IProcess
  virtual bool Start(IStatus** ppStatus);
  virtual bool End(IStatus** ppStatus);
  virtual bool IsInProcess();

  // IAudioTarget
  virtual bool RawWrite(BYTE* pData, int iNumberOfBytesToWrite,
            EAudioRawState& state,
            IStatus** ppStatus);

  virtual SWavFormat GetRawFormat();

  // IInfoWriter
  virtual void SetTitle(const char* szTitle);
  virtual void SetArtist(const char* szArtist);
  virtual void SetAlbum(const char* szAlbum);
```

```
  virtual void SetYear(const char* szYear);
  virtual void SetGenre(const char* szGenre);

// Data
protected:
  CString                   m_csURL;
  int                       m_dwWrittenRAWSize;
  HANDLE                    m_hFile;
  EAuxFlags                 m_flags;
  CComPtr<IAudioComponent>  m_pCreator;
  SWavFormat                m_DlgData;
  bool                      m_bEndProcess;
};

#endif // _WAV_TGT
```

10.16. WavTgt.cpp

```
/******************************************************************************
|* THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF
|* ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO
|* THE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
|* PARTICULAR PURPOSE.
|*
|* Copyright 2002-2005 Nero AG. All Rights Reserved.
|*-----------------------------------------------------------------------------
|* PROJECT: Nero Plug-in Manager Example
|*
|* FILE: WavTgt.cpp
|*
|* PURPOSE: Implementation of the wave target class
******************************************************************************/


#include "stdafx.h"
#include "WavTgt.h"

// helper functions for file access
#include "FileHelper.h"

#include "NeroPluginUtil.h"


// IAudioItem implementation

// return the object that created the wave target
bool CWavTgt::GetCreator(IAudioComponent** pCreator)
{
  if(!pCreator)
  {
    ASSERT(FALSE);
    return false;
  }

  *pCreator = m_pCreator;

  (*pCreator)->AddRef();
```

```
  return true;
}

EAuxFlags CWavTgt::GetAuxFlags()
{
  return m_flags;
}

void CWavTgt::SetAuxFlags(EAuxFlags flags)
{
  m_flags = flags;
}

bool CWavTgt::SetURL(const char* szURL, IStatus** ppStatus)
{
  // only local URLs are allowed for this target
  if(false == IsURLLocal(szURL))
  {
    RETURN_ERROR(AE_BadPathName);
  }

  m_csURL = szURL;

  return true;
}

const char * CWavTgt::GetURL()
{
  return  m_csURL;
}

EURLType CWavTgt::GetType()
{
  return URL_LocalFile;
}

// IWavTgtSetDlgData implementation

SWavFormat * CWavTgt::GetDlgData()
{
  return &m_DlgData;
}

// IProcess implementation

bool CWavTgt::Start(IStatus **ppStatus)
{
  if(ppStatus)
  {
    *ppStatus = NULL;
  }

  m_hFile = CreateFile(m_csURL, GENERIC_WRITE, 0, NULL, CREATE_ALWAYS,
                       FILE_ATTRIBUTE_NORMAL, NULL);

  DWORD dwError = GetLastError();

  bool bRes = (m_hFile != INVALID_HANDLE_VALUE);
```

```
  if(false == bRes)
  {
    RETURN_ERROR(dwError);
  }

  int iSamplesPerSec = m_DlgData.m_iSamplesPerSecond;


  // Starting the target

  m_bEndProcess = FALSE;

  DWORD dwWritten   = 0;

  // Starting header

  char* szHeader    = "RIFF****WAVEfmt ";
  WriteFile(m_hFile, szHeader, strlen(szHeader), &dwWritten, NULL);

  // WAVE chunk size

  DWORD dw = 16;
  WriteFile(m_hFile, &dw, sizeof(DWORD), &dwWritten, NULL);

  // Sampling format

  PCMWAVEFORMAT format =
  {
    WAVE_FORMAT_PCM,
    m_DlgData.m_iChannels,
    iSamplesPerSec,
    (m_DlgData.m_iChannels * iSamplesPerSec *
            m_DlgData.m_iBitsPerSample) / BITS_PER_BYTE,
    (m_DlgData.m_iChannels * m_DlgData.m_iBitsPerSample) /
                                BITS_PER_BYTE,
    m_DlgData.m_iBitsPerSample
  };

  WriteFile(m_hFile, &format, sizeof(PCMWAVEFORMAT), &dwWritten, NULL);

  // Data chunk header

  char *szData = "data****";
  WriteFile(m_hFile, szData, strlen(szData), &dwWritten, NULL);

  m_dwWrittenRAWSize = 0;

  // The remaining information will be saved as soon as RAW data is received
  // and the artist, copyright and other additional information will be
  // save in EndTarget, after the whole RAW chunk is ready.

  return bRes;
}

bool CWavTgt::End(IStatus** ppStatus)
{
  ASSERT(m_hFile != INVALID_HANDLE_VALUE);

  if(ppStatus)
```

```
{
  *ppStatus = NULL;
}

// align to the word size the RAW chunk
if(m_dwWrittenRAWSize % 2)
{
  WriteZeroByte(m_hFile);
}

// List chunk
DWORD dwListChunkSizePos  = 0,
    dwWritten     = 0;

CString *pcsCurField    = &m_csArtist;

BOOL  bHeaderIsWritten  = FALSE;

DWORD dwListChunkSize   = 0;

char *szarListSubTitles[] =
{
  "IART",   // Artist
  "ICMT",   // Comments
  "ICOP",   // Copyright
  "ICRD",   // Creation date
  "INAM"    // Name (the title)
};

for(int i = 0; i < LIST_CHUNK_ITEMS; i++, pcsCurField++)
{
  if(!pcsCurField->IsEmpty())
  {
    // the size of list item chunk size

    dwListChunkSize += 8;

    if (!bHeaderIsWritten)
    {
      // the INFO word

      dwListChunkSize += WAV_CHUNK_NAME_LEN;

      dwListChunkSizePos =
            GetFilePointer(m_hFile) + WAV_CHUNK_NAME_LEN;

      char *szListHeader = "LIST****INFO";
      WriteFile(m_hFile, szListHeader, strlen(szListHeader),
            &dwWritten, NULL);

      bHeaderIsWritten = TRUE;
    }

    // Subchunk title

    WriteFile(m_hFile, szarListSubTitles[i], WAV_CHUNK_NAME_LEN,
          &dwWritten, NULL);

    DWORD dwDataLen = pcsCurField->GetLength() + 1;
```

```
    dwListChunkSize += dwDataLen;

    // The size of this subchunk

    WriteFile(m_hFile, &dwDataLen, sizeof(DWORD),
          &dwWritten, NULL);

    // The subchunk content

    WriteFile(m_hFile, (LPCTSTR)(*pcsCurField), dwDataLen,
          &dwWritten, NULL);

    // align the chunk's size to the word size
    if(dwDataLen % 2)
    {
      WriteZeroByte(m_hFile);
      dwListChunkSize++;
    }
  }
}

// Fill the size field of the LIST chunk

if(bHeaderIsWritten)
{
  SetFilePointer(m_hFile, dwListChunkSizePos, 0, FILE_BEGIN);

  WriteFile(m_hFile, &dwListChunkSize, sizeof(DWORD),
        &dwWritten, NULL);

  // align the list chunk's size to the word size

  if(dwListChunkSize % 2)
    WriteZeroByte(m_hFile);
}

// Filling the RIFF chunk's size

DWORD dwFileSize = GetFileSize(m_hFile, NULL);

if (dwFileSize % 2)
{
  // Going to the end of the file to align
  // the file's size to size of word

  SetFilePointer(m_hFile, 0, 0, FILE_END);
  WriteZeroByte(m_hFile);

  dwFileSize++;
}

SetFilePointer(m_hFile, WAV_CHUNK_NAME_LEN, 0, FILE_BEGIN);

// The RIFF header size

dwFileSize -= 8;

WriteFile(m_hFile, &dwFileSize, sizeof(DWORD), &dwWritten, NULL);
```

```
  // Writing the "data" chunk's size

  SetFilePointer(m_hFile, 40, 0, FILE_BEGIN);

  WriteFile(m_hFile, &m_dwWrittenRAWSize, sizeof(DWORD), &dwWritten,
        NULL);

  CloseHandle(m_hFile);
  m_hFile = INVALID_HANDLE_VALUE;

  return true;
}

bool CWavTgt::IsInProcess()
{
  return (m_hFile != INVALID_HANDLE_VALUE);
}

// IAudioTarget implementation

bool CWavTgt::RawWrite(BYTE *pData, int iNumberOfBytesToWrite,
            EAudioRawState &state,
            IStatus **ppStatus)
{
  if(false == (pData && iNumberOfBytesToWrite))
  {
    RETURN_ERROR(AE_InvalidParameter);
  }

  ASSERT(m_hFile != INVALID_HANDLE_VALUE);

  if(ppStatus)
  {
    *ppStatus = NULL;
  }

  DWORD dwWritten = 0;

  // WriteFile returns zero for failure
  int iRes = WriteFile(m_hFile, pData, iNumberOfBytesToWrite, &dwWritten, NULL);

  if(0 == iRes)
  {
    DWORD dwError = GetLastError();
    RETURN_ERROR(dwError);
  }

  m_dwWrittenRAWSize += iNumberOfBytesToWrite;

  return true;
}

SWavFormat CWavTgt::GetRawFormat()
{
  return m_DlgData;
}

// IInfoWriter implementation
```

```
void CWavTgt::SetTitle(const char *szTitle)
{
  m_csTitle = szTitle;
}

void CWavTgt::SetArtist(const char *szArtist)
{
  m_csArtist = szArtist;
}

void CWavTgt::SetAlbum(const char *szAlbum)
{
  // not implemented
}

void CWavTgt::SetYear(const char *szYear)
{
  // not implemented
}

void CWavTgt::SetGenre(const char *szGenre)
{
  // not implemented
}
```

10.17. WavTgtFactory.h

```
/******************************************************************************
|* THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF
|* ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO
|* THE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
|* PARTICULAR PURPOSE.
|*
|* Copyright 2002-2005 Nero AG. All Rights Reserved.
|*-----------------------------------------------------------------------------
|* PROJECT: Nero Plug-in Manager Example
|*
|* FILE: WavTgtFactory.h
|*
|* PURPOSE: Declaration file for the wave target factory class.
******************************************************************************/

#ifndef _WAV_TGT_FACTORY_
# define _WAV_TGT_FACTORY_

#if _MSC_VER > 1000
# pragma once
#endif // _MSC_VER > 1000


// CStatus and CAggregatable
#include "AudioBase.h"


class CWavTgtFactory :  public  IIdentifiable,
                        public  CAggregatable,
                        public  IAudioComponent,
```

```
                        IURLAudioTargetFactory,
                        IExtEnum
{
// Construction/Destruction
public:
  CWavTgtFactory()
  : CAggregatable (AGGFLAG_NODELETE)
  {}

  virtual ~CWavTgtFactory() {};

// Operations
public:
  virtual long GetRefCount();

  AGGREGATABLE_INTERFACE_MAP_BEGIN
    INTERFACE_ENTRY(IIdentifiable)
    INTERFACE_ENTRY(IAudioComponent)
    INTERFACE_ENTRY(IURLAudioTargetFactory)
    INTERFACE_ENTRY(IExtEnum)
  AGGREGATABLE_INTERFACE_MAP_END

  // IIdentifiable
  void GetID(GUID *pGUID);

  // IAudioComponent
  virtual const char* GetName();
  virtual EAudioComponentType GetType();
  virtual bool Init(IAudioPluginMgr* pMgr, IStatus** ppStatus);
  virtual bool Done();

  // IURLAudioTargetFactory
  virtual bool CreateURLAudioTarget(IUnknown** ppTgt,
                                    const SWavFormat& formatSrc,
                                    IStatus** ppStatus);

  virtual bool EditSettings(IUnknown** ppTgt, int iCount);

  // Application should first call this function to determine if this
  // factory can edit settings for items.
  virtual bool CanEditSettings();
  virtual EURLType GetSupportedURLTypes();

  // IExtEnum
  int GetCount();

  // The returned value can't be stored for later use.
  // The application must copy it.
  const char* GetExt(int iNum);
};


#endif _WAV_TGT_FACTORY
```

10.18. WavTgtFactory.cpp

```
/******************************************************************************
|* THIS CODE AND INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF
|* ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO
|* THE IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A
|* PARTICULAR PURPOSE.
|*
|* Copyright 2002-2005 Nero AG. All Rights Reserved.
|*-----------------------------------------------------------------------------
|* PROJECT: Nero Plug-in Manager Example
|*
|* FILE: WavTgtFactory.cpp
|*
|* PURPOSE: Implementation of the wave target factory
******************************************************************************/


#include "stdafx.h"

#include "WavTgtFactory.h"

// {9E424B23-3D6A-48ca-A41D-B65927057499}
EXTERN_GUID(COMPID_TgtFactory , 0x9e424b23, 0x3d6a, 0x48ca, 0xa4, 0x1d, 0xb6, 0x59, 0x27, 0x5,
0x74, 0x99);

// Wave Target class
#include "WavTgt.h"

// Wave Settings dialog
#include "dlg_WavSettings.h"


extern IAudioPluginMgr* g_pPluginMgr;


void CWavTgtFactory::GetID(GUID *pGUID)
{
    ASSERT(pGUID);
    if(pGUID)
       memcpy(pGUID, &COMPID_TgtFactory, sizeof(GUID));
}

const char* CWavTgtFactory::GetName()
{
  return "PCM Wav file";
}

EAudioComponentType CWavTgtFactory::GetType()
{
  return ACT_AudioURLTargetFactory;
}

bool CWavTgtFactory::Init(IAudioPluginMgr*  pMgr, IStatus** ppStatus)
{
  if(ppStatus)
  {
    *ppStatus = NULL;
```

```
  }

  if(g_pPluginMgr)
  {
    ASSERT(g_pPluginMgr == pMgr);
  }
  else
  {
    g_pPluginMgr = pMgr;
  }

  // success
  return true;
}

bool CWavTgtFactory::Done()
{
  return true;
}

// IURLAudioTargetFactory

bool CWavTgtFactory::CreateURLAudioTarget(IUnknown**        ppTgt,
                                          const SWavFormat& formatSrc,
                                          IStatus**         ppStatus)
{
  if(ppStatus)
  {
    *ppStatus = NULL;
  }

  *ppTgt = static_cast<IAggregatable*>
            (new CWavTgt(formatSrc, (IAudioComponent*)this));
  if(*ppTgt)
  {
    (*ppTgt)->AddRef();
  }

  return true;
}

bool CWavTgtFactory::EditSettings(IUnknown **ppTgt, int iCount)
{
  SWavFormat **ppDlgData = new SWavFormat*[iCount];

  if(!(ppTgt && iCount && ppDlgData))
  {
    ASSERT(FALSE);
    return false;
  }

  AFX_MANAGE_STATE(AfxGetStaticModuleState());

  for(int i = 0; i < iCount; i++)
  {
    CComQIPtr<IWavTgtSetDlgData> pDlgData = ppTgt[i];

    if(!pDlgData)
    {
```

```
      // This is an object of not a valid type because we cannot
      // retrieve our internal interface from it.
      ASSERT(FALSE);
      return false;
    }

    ppDlgData[i] = pDlgData->GetDlgData();
  }

  CWavSettingsDlg dlg(ppDlgData, iCount);

  int iDlgRet = dlg.DoModal();

  delete ppDlgData;

  return (iDlgRet == IDOK);
}

// Application should first call this function to determine if this
// factory can edit settings for items.
bool CWavTgtFactory::CanEditSettings()
{
  return true;
}

EURLType CWavTgtFactory::GetSupportedURLTypes()
{
  return URL_LocalFile;
}


// IExtEnum
int CWavTgtFactory::GetCount()
{
  return 2;
}

// The returned value can't be stored for later use.
// The application must copy it.
const char* CWavTgtFactory::GetExt(int iNum)
{
  if(0 == iNum)
  {
    return "wav";
  }
  else if(1 == iNum)
  {
    return "wave";
  }

  ASSERT(FALSE);

  return NULL;
}

long CWavTgtFactory::GetRefCount()
{
  return m_lRef;
}
```