WaveLab Application Programming Interface

Introduction

Application scripting in WaveLab enables you to automate tasks that might otherwise take a long time to perform manually, or to control WaveLab remotely as part of an automated system. For example, suppose you often have to normalize and convert audio files to MP3 with specific settings… with the appropriate script, all you have to do is to drag and drop the audio file onto the script file: WaveLab is executed and the appropriate sequence of functions is called. Done! Or suppose you need to automate the recording or playback of files: a couple of script lines and you're there. You can run scripts that others have written or write your own scripts. You can also control WaveLab from a C++ application.You can write scripts in two languages: VBScript and JScript. Learn about these simple languages at http://msdn.microsoft.com/library (once there, search for VBScript or Jscript). You simply use a text editor to write scripts. There are many references on the internet about VBScript and JScript. If you want to create scripts, you will need to learn the basis of one of these script languages.

To run a script, simply double-click on the script file in the Windows environment (you run script files the same way you run regular executable files). For certain scripts, drag and drop a file on them. Alternate method: in the Windows environment, click the **Start** button, and then click **Run.** In the **Open** field of the **Run** dialog box, type the full path of the script together with some possible arguments (eg. file name), and click **OK**.

You can find some small script samples in the WaveLab\Scripting\ folder. But open them in a text editor to see the comments first, to know what they do. There is also the WaveLab "type library" (WaveLab.tlb") if you need to implement a client application eg. in C++. Finally, there is a very simple C++ test application in the CppTest folder (this was generated from Visual C++; this application only opens WaveLab, then closes it 2 seconds later).

API details

WaveLab provides a COM interface. This API can be called by scripts or standard applications (eg. C++). The following documentation provides a summary of classes, methods and properties exposed by the WaveLab scripting API. This API is simple by design: few classes, few methods, few arguments, but combining them can create powerful solutions.

Note: WaveLab makes much use of presets throughout the application (eg. File formats, Master Section presets, etc.) . The API often refers to these presets. That is, instead of defining too many details in a script (eg. to set a file format), you first make proper presets in WaveLab, then you refers to these presets in the script. See further.

Class **Application**

This class represents WaveLab and its main window.

Methods

- **Quit**()

  Quit WaveLab. If any file is not saved, the standard save dialog pops up.

- **ShowWindow**(int mode)

  mode = 0 -> then the WaveLab window gets minimized.

  mode = 1 -> then the WaveLab window gets maximized.

  mode = 2 -> then the WaveLab window gets restored to the position and size it had before being minimized.

Script Example:

```
Dim app
Set app = CreateObject("WaveLab.Application")
app.ShowWindow(0)
WScript.Sleep(2000)
app.ShowWindow(2)
WScript.Sleep(2000)
app.ShowWindow(1)
WScript.Sleep(2000)
app.Quit()
```

Class **AudioFileFormat**

This class represents a file format definition. It is used as an argument for file saving and rendering (see further). In WaveLab 4.01, you can save file format presets (Save As dialog, Render dialog, etc.), and an **AudioFileFormat** object represents exactly such a preset.

Methods

- **Load**(String presetName)

  Initialize the object with the preset called "presetName"

Script Example:

```
Dim ff
Set ff = CreateObject("WaveLab.AudiofileFormat")
ff.Load("mp2")
```

Class **MasterSectionPreset**

This class represents a Master Section preset. It is used as an argument for file rendering (see further).

Methods

- **Load**(String groupName, String presetName)

  Initialize the object with the preset called “presetName” from the group called “groupName”. See the WaveLab Master Section dialog for a better understanding of these two names.

  The preset names are case-sensitive!

Script Example:

```
Dim msp
Set msp = CreateObject("WaveLab.MasterSectionPreset")
Call msp.Load("MyGroup", "MyPreset")
```

Class **RecordSetup**

This class represents a Recording setup (see the WaveLab Record dialog). It is used as an argument for the recording API (see further).

Methods

- **Load**(String presetName)

  Initialize the object with the preset called “presetName”.

Script Example:

```
Dim rs
Set rs = CreateObject("WaveLab.RecordSetup")
rs.Load(“MyPreset”)
```

Class **Wave**

This class represents a wave file.

Methods

- **Open**(String filename)

  The wave file called "filename" will be open in a window. If the file is already open, nothing happens.

- **Close**()

  The wave file window gets closed.

- **Play**()

  The wave file is played from start to end.

- **SaveAs**(String fileName, AudioFileFormat* fmt)

  The wave file is saved under a specific name and with the specified format.

- **Render**(String fileName, AudioFileFormat* fmt, MasterSectionPreset* msp)

  The wave file is processed through a chain of plugins, and saved with the specified format and under the specified name.

Script Example:

```
Dim ff, msp, wave

Set ff = CreateObject("WaveLab.AudiofileFormat")
ff.Load("mp2")

Set msp = CreateObject("WaveLab.MasterSectionPreset")
Call msp.Load("MyGroup", "MyPreset")

Set wave = CreateObject("WaveLab.Wave")
wave.Play()
Call wave.Render(c:\a.wav, ff, msp)
Call wave.SaveAs(c:\b.wav, ff)
wave.Close()
```

Class **AudioMontage**

This class represents an Audio Montage

Methods

- **Open**(String filename)

  The Audio Montage called “filename” will be open in a window. If the file is already open, nothing happens.

- **Close**()

  The Audio Montage window gets closed.

- **Play**(String clipName)

  If “clipName” is an empty string, then the Audio Montage is played from start to end. Else, the specified clip, and only it, is played from start to end.

- **Render**(String fileName, AudioFileFormat* fmt, MasterSectionPreset* msp)

  The Audio Montage is processed through a chain of plugins, and a wave file is produced with the specified format and under the specified name.

Script Example:

```
Dim ff, msp, mon

Set ff = CreateObject("WaveLab.AudiofileFormat")
ff.Load("mp2")

Set msp = CreateObject("WaveLab.MasterSectionPreset")
Call msp.Load("MyGroup", "MyPreset")

Set mon = CreateObject("WaveLab.AudioMontage")
mon.Play("")
Call mon.Render(c:\a.wav, ff, msp)
mon.Close()
```

Class **Player**

This class represents the Playback engine.

Methods

- **Stop**()

  Stop any playback activity.

- **WaitEnd**()

  The function only returns when the current playback activity reaches its natural end (eg. end of file).

Properties

- **IsNowPlaying**

  This value is TRUE if playback is currently happening.

Script Example:

```
Dim mon, player

Set player = CreateObject("WaveLab.Player")
Set mon = CreateObject("WaveLab.AudioMontage")

mon.Play("")
WScript.Sleep(2000)
WScript.Echo player.IsNowPlaying
player.Stop()

mon.Play("")
player.WaitEnd()
```

Class **Recorder**

This class represents the Recorder engine.

Methods

- **Init**(String fileName, IRecordSetup* rs)

  Open the recorder window, sets the specified setup and destination file name.

- **Record**()

  Start recording (just like pressing the RECORD button in the window). By design, and unlike when using WaveLab directly, there is no message to warn you about a possible file overwriting.

- **Stop**()

  End recording (just like pressing the STOP button in the window)

- **Pause**()

  Pause recording (just like pressing the PAUSE button in the window)

- **Discard**()

  Discard recording (just like pressing the DISCARD button in the window, but without any warning).

- **End**()

  End recording and close the window.

Script Example:

```
Dim setup, recorder
Set setup = CreateObject("WaveLab.RecordSetup")
setup.Load("aaa")
Set recorder = CreateObject("WaveLab.Recorder")
Call recorder.Init("D:\a.wav", setup)

recorder.Record()
WScript.Sleep(4000)
recorder.Pause()
WScript.Sleep(4000)
recorder.Record()
WScript.Sleep(4000)
recorder.Stop()
```

Class **Batch**

This class represents a batch processor.

Methods

- **Set**(String outPath, String options, String pluginSet)

  Initialize the batch process with a destination path, the batch options (corresponding to a name from the batch preset page, ie. File formats, etc.), the plugins to use (corresponding to a name from the batch plugin preset dialog).

  The preset names are case-sensitive!

- **Clear**()

  Remove all file from the batch list, and remove all plugins too.

- **AddFile**(String fileName)

  Add a single audio file to the batch.

- **AddFileList**(String fileName)

  Read all the file names from a text file (one full name per line) and add these files to the batch.

- **AddFolder**(String path, String ext, BOOL subFolders)

  Add to the batch all the files contained in the specified folder, with the specified file extension, and optionally also search in sub-folders.

- **Run**()

  Start the batch processing and wait till the end of it.

- **End**()

  Close the batch window.

Properties

- **Error**

  This value is TRUE if an error has occurred during processing.

Script Example:

```
Dim batch
Set batch = CreateObject("WaveLab.Batch")
batch.Clear()
batch.AddFile("C:\MyPiano.wav")
batch.AddFolder("c:\MyProject", "wav", TRUE)
batch.AddFileList("c:\MyFiles.txt")
Call batch.Set("d:\out", "MyOptions", "MyPlugins")
batch.Run()
WScript.Echo batch.Error
batch.End()
```