About C# ATL ~ v. 0.9

Index

Introduction

This document is a short way to get started with the C# porting of the Audio Tools Library.

Overview

The main difference between the initial implementation and the C# porting is an additional layer that has been added on top of the format-specific classes, to ease and structure their use.

Each format-specific class belongs to the `ATL.AudioReaders.BinaryLogic` namespace, since it deals with the reading and writing of binary data. What's more, they are now divided into two kind of classes : **audio data readers** and **metadata readers**. These two roles are defined into the following interfaces :

«interface» AudioReaders:AudioDataReader	*«interface» AudioReaders:MetaDataReader*
+BitRate() : double +Duration() : double +IsVBR() : bool +CodecFamily() : int +ID3v1() : TID3v1 +ID3v2() : TID3v2 +APETag() : TAPEtag +ReadFromFile(input fileName : String) : bool	+Exists() : bool +Title() : String +Artist() : String +Comment() : String +Genre() : String +Track() : int +Year() : String +Album() : String

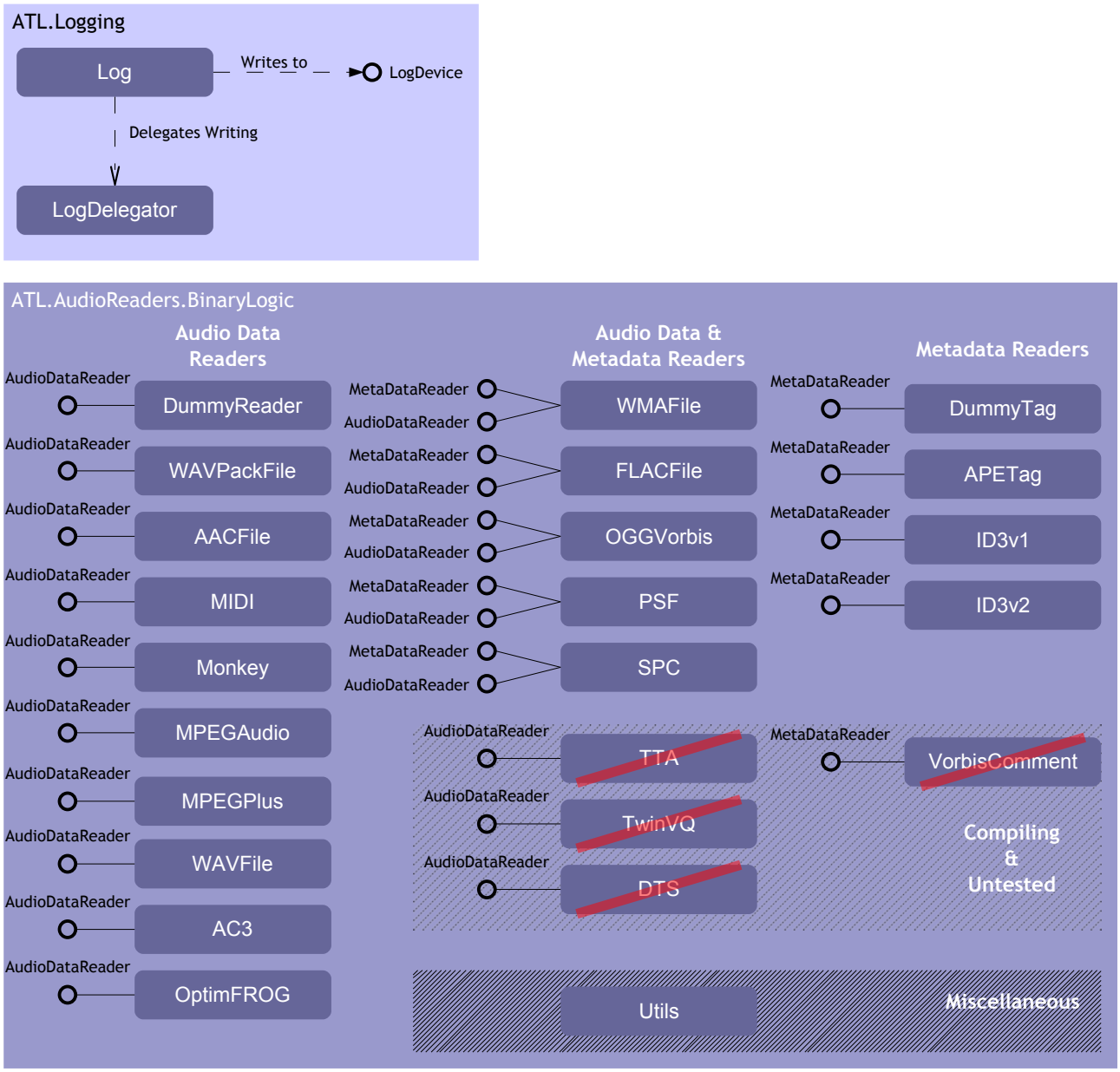
According to the extension of a source file, the `AudioReaderFactory` class will help instantiate the fittest objects to read from the file. If the format is unsupported, a dummy reader will be used.

On an upper level, `AudioFileReader` makes the calls to the factory and provides its caller all the required properties – both audio and meta – without having it worrying about anything.

Last but not least, events occurring during the acquisition are logged into the Ethos logging system, which is for the moment packaged with the C# ATL. Future releases will most probably have this constraint loosened.

Classes from the current version (0.9) have only had their reading methods tested ("get info from a file"). The writing methods (rebuild file, change tags, ...) have not been tested yet !

Class diagram



NB : Strikeouted classes have been translated so that they compile, but have never been tested. They will most probably not provide the expected results.

Sample Code : Getting info from an audio file

All mentioned fields (m_size, m_title, ...) are private attributes of the class where this sample method is located.

```
public void GetInfoFromAudioFile(String filePath)
{
	ATL.AudioReaders.AudioFileReader theATLReader = new ATL.AudioReaders.AudioFileReader(filePath);
	FileInfo theFileInfo = new FileInfo(filePath);

	if (theFileInfo.Exists)
	{
		m_size = theFileInfo.Length;
		m_lastModified = theFileInfo.LastWriteTime.Ticks;
	}

	m_title = theATLReader.Title;
	if ("" == m_title || null == m_title)
	{
		m_title = System.IO.Path.GetFileNameWithoutExtension(filePath);
	}
	m_artist = theATLReader.Artist;
	if (null == m_artist) { m_artist = ""; }
	m_comment = theATLReader.Comment;
	if (null == m_comment) { m_comment = ""; }
	m_genre = theATLReader.Genre;
	if (null == m_genre) { m_genre = ""; }
	m_year = theATLReader.Year;
	m_album = theATLReader.Album;
	m_trackNumber = theATLReader.Track;
	m_bitrate = theATLReader.BitRate;
	m_codecFamily = theATLReader.CodecFamily;
	m_duration = theATLReader.Duration;
}
```

Note that using the logging mechanism is optional : you may completely ignore it while using ATL. However, if you want to get an output of the errors (that's why the logging interface exists), you should configure it this way :

```
	ATL.Logging.Log log = new ATL.Logging.Log();

	// Log now delegates its logging to LogDelegator
	ATL.Logging.LogDelegator.SetLog(ref log);

	// someObject implements LogDevice
	// => all the logged messages will result to a call to someObject's DoLog method
	log.Register(someObject);
```

Known issues of porting Pascal to C#

Several routines from the Delphi API don't have any equivalent in C#, thus the porting has proven to be a little tricky in several parts :

- Basically, every method that can be found in the `ATL.AudioReaders.BinaryLogic.Utils` class is the solution to a porting problem that has been encountered.
- The native handling of Unicode in the .NET framework is both a good thing (no need for special libraries such as TntWare Delphi Unicode Controls) and a bad thing (no native routine ensures to read 1-byte chars from a stream, since any char can be an unicode char from the .NET point of view – a workaround is to read bytes and convert them).
- Any algorithm involving pointers is in the best case labelled as unsafe (if not unreproductible) when using C#. The checksum routines used in `OggVorbis` are temporarily commented until a cleaner alternative is though and implemented.

History

0.9 – 07/08/2006

- First public release
- ID3v2 : fixed a bug in genre recognition, UTF-8-encoded tags are readable
- Support for SPC700 audio format & ID666 tags
- Support for PSF audio format and its derivates (PSF2, SSF, DSF, USF, GSF, QSF)

0.8 – 02/06/2006

- A bug concerning null-terminated strings reading has been fixed on ID3v1
- MIDI parsing is faster but still needs to be improved
- OptimFROG and AC3 have been tested for reading

0.7 – 11/29/2005

First release of the C# library source

Credits

The Audio Tools Library has been created and maintained by the MAC Team, in Pascal language :

http://mac.sourceforge.net/atl/

The C# porting has been initiated by Zeugma 440 for the Ethos Project :

http://ethos.no-ip.com