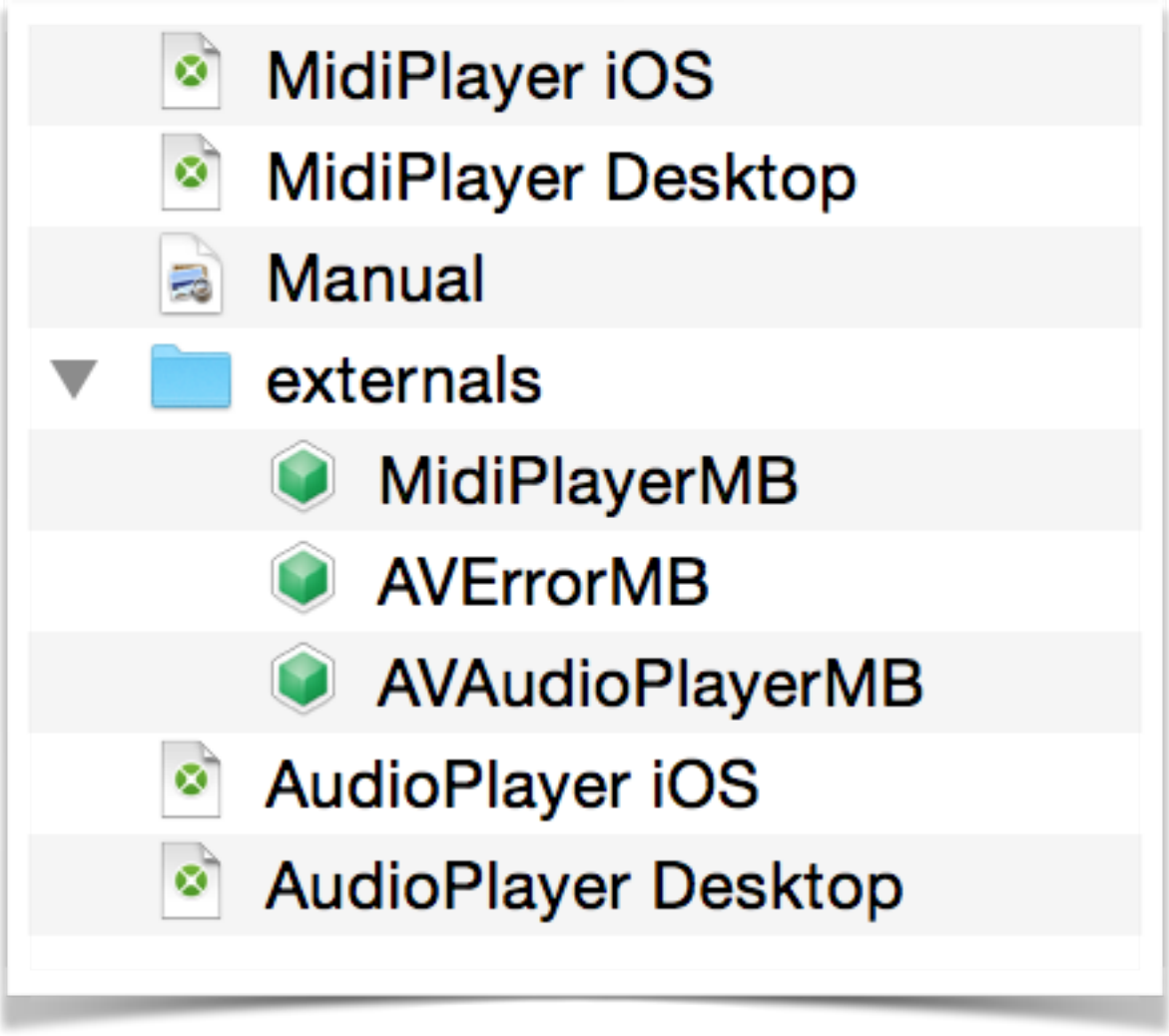
MBS Xojo AudioPlayer Kit

Version 1.1, © 2019 by Christian Schmitz


MidiPlayer iOS
MidiPlayer Desktop
Manual
externals
MidiPlayerMB
AVErrorMB
AVAudioPlayerMB
AudioPlayer iOS
AudioPlayer Desktop

About the MBS Xojo AudioPlayer Kit

The MBS Xojo AudioPlayer Kit provides you with a few useful classes for audio playback on OS X and iOS and midi playback on OS X.

The Kit contains:

- Wrapper for iOS AudioPlayer class in AVFoundation
  - AVAudioPlayerMB class
  - AVErrorMB class
- Midi Player class to use Apple's midi synthesizer
  - MidiPlayerMB class
- Sample application

AudioPlayer
The AudioPlayer Kit contains a wrapper for AVAudioPlayer class:

Features

- Play sounds loaded from memory and file
- Events like DidFinishPlaying
- Audio Channel assignment
- average and peak power per channel
- volume, pan and rate controls
- play, pause and stop methods
- query/set current position

Wrapper Features

- For new Xojo framework
- Using exception handling to track error
- Using enums for saver constant passing
- Test code included
- All classes with MB postfix to avoid name conflicts.
- All module definitions are protected to avoid conflicts
- Inline documentation
- Full Source code, no encryption
- Works for 32bit and 64bit targets.

MidiPlayer
The AudioPlayer Kit contains a midi player class:

Features

- Open digital synthesizer for midi note playback
- query instruments and set instrument per channel
- send midi notes
- Start and Stop method
- Volume, Tuning, Reverbvolume properties
- query CPU load

Wrapper Features

- For new Xojo framework
- Using exception handling to track error
- Using enums for saver constant passing
- Test code included
- All classes with MB postfix to avoid name conflicts.
- All module definitions are protected to avoid conflicts
- Inline documentation
- Full Source code, no encryption
- Works for 32bit and 64bit targets.

Limitations

- While we can compile for iOS and include a sample project, Apple does not include the synthesizer component with current iOS versions.
- Accuracy of timing is limited to Xojo's timer precision.

Interfaces

AVErrorMB

```
Class AVErrorMB
ComputedProperty code As integer
        Sub Get()
                get error code
ComputedProperty domain As text
        Sub Get()
                get error domain
ComputedProperty localizedDescription As text
        Sub Get()
                get localized error description
Sub Constructor(Handle as Integer, Retain as Boolean)
        Create from Handle
Sub Destructor()
        Cleanup
Note "Copyright"
        © 2015 by Christian Schmitz, Monkeybread Software

        This is part of the MBS Xojo Event Kit for iOS

        http://www.monkeybreadsoftware.de/xojo/
Property Handle As Integer
End Class
```

AVAudioPlayerMB

```
Class AVAudioPlayerMB
ComputedProperty URL As Text
        Sub Get()
                Returns nil if object was not created with a URL
ComputedProperty currentTime As Double
        Sub Set()
                If the sound is playing, currentTime is the offset into the sound of the current playback position.
                If the sound is not playing, currentTime is the offset into the sound where playing would start.
        Sub Get()
                The playback point, in seconds, within the timeline of the sound associated with the audio player.
                If the sound is playing, currentTime is the offset of the current playback position,
                measured in seconds from the start of the sound. If the sound is not playing,
                currentTime is the offset of where playing starts upon calling the play method,
                measured in seconds from the start of the sound.

                By setting this property you can seek to a specific point in a sound file or implement audio fast-forward and
rewind functions.
ComputedProperty deviceCurrentTime As Double
        Sub Get()
                returns the current time associated with the output device
ComputedProperty duration As Double
        Sub Get()
                Returns the total duration, in seconds, of the sound associated with the audio player.
ComputedProperty enableRate As Boolean
        Sub Set()
                A Boolean value that specifies whether playback rate adjustment is enabled for an audio player.

                To enable adjustable playback rate for an audio player, set this property to true after you initialize the player
                and before you call the prepareToPlay instance method for the player.
        Sub Get()
ComputedProperty meteringEnabled As Boolean
        Sub Set()
                A Boolean value that specifies the audio-level metering on/off state for the audio player.

                The default value for the meteringEnabled property is off (Boolean false).
                Before using metering for an audio player, you need to enable it by setting this property to true.
        Sub Get()
ComputedProperty numberOfChannels As integer
        Sub Get()
                The number of audio channels in the sound associated with the audio player.
ComputedProperty numberOfLoops As integer
        Sub Set()
                The number of times a sound will return to the beginning, upon reaching the end, to repeat playback.

                A value of 0, which is the default, means to play the sound once.
                Set a positive integer value to specify the number of times to return to the start and play again.
                For example, specifying a value of 1 results in a total of two plays of the sound.
                Set any negative integer value to loop the sound indefinitely until you call the stop method.
        Sub Get()
ComputedProperty pan As Single
        Sub Set()
                The audio player's stereo pan position.
                By setting this property you can position a sound in the stereo field.
                A value of –1.0 is full left, 0.0 is center, and 1.0 is full right.
        Sub Get()
ComputedProperty playing As Boolean
        Sub Get()
                A Boolean value that indicates whether the audio player is playing (true) or not (false).
                To find out when playback has stopped implement the DidFinishPlaying event.
ComputedProperty rate As Single
        Sub Set()
                The audio player's playback rate.
```

This property's default value of 1.0 provides normal playback rate.
The available range is from 0.5 for half-speed playback through 2.0 for double-speed playback.

To set an audio player's playback rate, you must first enable rate adjustment as described
in the enableRate property description.

Sub Get()

ComputedProperty volume **As Single**

Sub Set()

The playback volume for the audio player, ranging from 0.0 through 1.0 on a linear scale.

Sub Get()

Sub Constructor(Handle **as Integer**, Retain **as Boolean**)

Create from Handle

Sub Constructor(Data **as** MemoryBlock, **byref** error **as** AVErrorMB)

Initializes and returns an audio player for playing a designated memory buffer.

Sub Constructor(Data **as** MemoryBlock, utiString **as** Text, **byref** error **as** AVErrorMB)

Initializes audio with MemoryBlock

Sub Constructor(URL **as** Text, **byref** error **as** AVErrorMB)

Initializes and returns an audio player for playing a designated sound file.

Sub Constructor(URL **as** Text, utiString **as** Text, **byref** error **as** AVErrorMB)

Initializes audio with URL

Sub Constructor(file **as** folderitem, **byref** error **as** AVErrorMB)

Initializes audio with FolderItem

Sub Constructor(file **as** folderitem, utiString **as** Text, **byref** error **as** AVErrorMB)

Initializes audio with FolderItem

Sub Destructor()

Cleanup

Function averagePowerForChannel(channel **as integer**) **As single**

Returns the average power for a given channel, in decibels, for the sound being played.

channel: The audio channel whose average power value you want to obtain. Channel numbers are zero-indexed.
A monaural signal, or the left channel of a stereo signal, has channel number 0.

Returns a floating-point representation, in decibels, of a given audio channel's current average power.
A return value of 0 dB indicates full scale, or maximum power;
a return value of –160 dB indicates minimum power (that is, near silence).

If the signal provided to the audio player exceeds ±full scale, then the return value may exceed 0
(that is, it may enter the positive range).

To obtain a current average power value, you must call the updateMeters method before calling this method.

Function channelAssignments() **As Integer**()

The channels property lets you assign the output to play to specific channels as described by AVAudioSession's channels property

This property is nil valued until set.
The array must have the same number of channels as returned by the numberOfChannels property.

Sub pause()

Pauses playback; sound remains ready to resume playback from where it left off.
Calling pause leaves the audio player prepared to play;
it does not release the audio hardware that was acquired upon calling play or prepareToPlay.

Function peakPowerForChannel(channel **as integer**) **As single**

Returns the peak power for a given channel, in decibels, for the sound being played.

channel: The audio channel whose peak power value you want to obtain. Channel numbers are zero-indexed.
A monaural signal, or the left channel of a stereo signal, has channel number 0.

Returns a floating-point representation, in decibels, of a given audio channel's current peak power.
A return value of 0 dB indicates full scale, or maximum power;
a return value of –160 dB indicates minimum power (that is, near silence).

If the signal provided to the audio player exceeds ±full scale, then the return value may exceed 0
(that is, it may enter the positive range).

To obtain a current peak power value, you must call the updateMeters method before calling this method.

Function play() **As Boolean**

plays sound. sound is played asynchronously.
Calling this method implicitly calls the prepareToPlay method if the audio player is not already prepared to play.

Function playAtTime(time **as Double**) **As Boolean**
	Plays a sound asynchronously, starting at a specified point in the audio output device's timeline.
	time is an absolute time based on and greater than deviceCurrentTime.
Function prepareToPlay() **As Boolean**
	Prepares the audio player for playback by preloading its buffers.
	Calling this method preloads buffers and acquires the audio hardware needed for playback,
	which minimizes the lag between calling the play method and the start of sound output.

	Calling the stop method, or allowing a sound to finish playing, undoes this setup.
Sub setChannelAssignments(values() **as Integer**)
	The channels property lets you assign the output to play to specific channels as described by AVAudioSession's channels property
	This property is nil valued until set.
	The array must have the same number of channels as returned by the numberOfChannels property.
Sub stop()
	Stops playback and undoes the setup needed for playback.

	Calling this method, or allowing a sound to finish playing,
	undoes the setup performed upon calling the play or prepareToPlay methods.

	The stop method does not reset the value of the currentTime property to 0.
	In other words, if you call stop during playback and then call play,
	playback resumes at the point where it left off.
Sub updateMeters()
	Refreshes the average and peak power values for all channels of an audio player.

	To obtain current audio power values, you must call this method before calling averagePowerForChannel or peakPowerForChannel.
Note "Copyright"
	

	This is part of the MBS Xojo Network Kit for iOS

	http://www.monkeybreadsoftware.de/xojo/
Property Data **As** MemoryBlock
	nil if object was not created with a MemoryBlock
Property Handle **As Integer**
End Class

MidiPlayerMB

```
Class MidiPlayerMB
ComputedProperty CPULoad As Single
        Sub Get()
                Returns a short-term running average of the current CPU load
ComputedProperty IsRunning As Boolean
        Sub Get()
ComputedProperty MaxCPULoad As Single
        Sub Get()
                Returns the max CPU load of the graph since this call was last made or the graph was last started.
ComputedProperty ReverbVolume As Single
        Sub Set()
        Sub Get()
ComputedProperty StreamFromDisk As Boolean
        Sub Set()
                set StreamFromDisk
        Sub Get()
                StreamFromDisk?
ComputedProperty Tuning As Single
        Sub Set()
        Sub Get()
ComputedProperty UsesInternalReverb As Boolean
        Sub Set()
                set to use internal reverb
        Sub Get()
                uses internal reverb?
ComputedProperty Volume As Single
        Sub Set()
        Sub Get()
ComputedProperty instrumentCount As Integer
        Sub Get()
                query number of instruments
Sub Constructor(WithMatrixReverb as Boolean, Start as Boolean = false)
        initialize the midi player
Sub Destructor()
        Cleanup
Sub SendMidiEvent(Status as Integer, Data1 as Integer, Data2 as Integer, OffsetSampleFrame as Integer)
        sends midi event
Sub Start()
        start playback
Sub Stop()
        stop playback
Function channelInstrument(channel as integer) As integer
        query which instrument is on which channel
Function instrumentIDAtIndex(InstrumentIndex as integer) As Integer
        query id of instrument
Function nameOfInstrument(InstrumentID as integer) As text
        query name of instrument
Sub setInstrument(Channel as Integer, instrumentID as Integer)
        sets instrument
Note "DataTypes"
        AUGraph    = pointer to struct OpaqueAUGraph -> Integer or Ptr
        AudioUnit  = pointer to struct ComponentInstanceRecord -> Integer or Ptr
        AUNode     = Int32
Property LastError As Integer
Property filterNode As Integer
Property graphHandle As Integer
Property mChannelInstrument(15) As Integer
Property outputNode As Integer
Property synthNode As Integer
End Class
```

Version History

Tip: If you want to update your existing code with new release, you'd best compare projects with Arbed (http://www.tempel.org/Arbed) and copy modifications to your project.

1.1, 31st Juli 2019

- Fixed bug in nameOfInstrument function.
- Uses different codes now to initialize sampler on iOS, so Midi Player works there.
- Added LoadPreset and LoadFromDLSOrSoundFont
- Updated for Xojo 2019r1

1.0, first release

Installation

To get your projects working with this AudioPlayer Kit, you need to follow a few steps.

Drop the folder „externals“ into your project and access all the common crypto or zip modules and classes. Or copy from existing example projects what you need.

Requirements

You need Xojo 2015r1 or newer.
We did not test with older versions, but you can if you need.

License

Summary:
- You may use AudioPlayer Kit only with one licensed Xojo installation.
- You agree not to share the AudioPlayer Kit or use someone else's AudioPlayer Kit copy.

Christian Schmitz Software GmbH, of Nickenich Germany is the owner, developer and sole copyright holder of this product, which is licensed -not sold- to you on a non-exclusive basis.
You agree not to share your MBS Xojo AudioPlayer Kit with anyone.

You may transfer your license to another person only after receiving written authorization from Christian Schmitz Software GmbH and only if the recipient agrees to be bound by the terms of this agreement.
Christian Schmitz Software GmbH reserves the right to cancel the license key(s) of any user who Christian Schmitz Software GmbH determines is in violation of this agreement.
THE WARRANTIES IN THIS AGREEMENT REPLACE ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING ANY WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE IS PROVIDED "AS IS" AND Christian Schmitz Software GmbH DISCLAIMS AND EXCLUDES ALL OTHER WARRANTIES. IN NO EVENT WILL Christian Schmitz Software GmbH BE LIABLE FOR ANY SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, EVEN IF WE HAVE KNOWLEDGE OF THE POTIENTIAL LOSS OR DAMAGE.
If you are located in Germany this agreement is subject to the laws of Germany. If you are located outside Germany local law may apply. Some states do not allow the exclusion of warranties, so the above exclusion may not apply to you.
Christian Schmitz Software GmbH does not charge royalties or deployment fees for Xojo applications.

Access to updates is included for one year. After that time you can order an update or keep using the old version you have.

Contact

Christian Schmitz Software GmbH
Eckertshohl 22
56645 Nickenich
Germany

Email: support@monkeybreadsoftware.de

Phone: +49 26 32 95 89 55 (Office) or +49 17 58 36 37 10 (Mobile)