AudioFile

The AudioFile class describes the properties of audio files.

On this page:

Class Hierarchy

- AudioFile
- Element
  - Bus
  - Effect
  - Instance
  - Layer
    - Program
  - MidiModule
  - ModulationMatrixRow
  - Slot
  - Zone
- Event
- LoadProgress
- ParameterDefinition

Classes

AudioFile Class

Description

The `AudioFile.open` function creates an AudioFile object of the specified audio file. The AudioFile object can be used to retrieve information from the audio file, for example, the sample rate, bit depth, length in samples, etc. The AudioFile object has the following fields.

All fields of the AudioFile object are read-only.

Available in: Controller, Processor.

Fields

.valid	Indicates if the file is a supported audio file and whether it could be opened or not.	boolean
.fileName	The file name that was used for opening the audio file.	string
.rate	The sample rate of the audio file. Returns `nil` if the audio file could not be opened or is invalid.	number
.bits	The bit depth of the audio file. Returns `nil` if the audio file could not be opened or is invalid.	number
.channels	The number of channels of the audio file. Returns `nil` if the audio file could not be opened or is invalid.	number
.length	The number of samples in the audio file. Returns `nil` if the audio file could not be opened or is invalid.	number
.rootKey	The root key stored in the sampler chunk of the audio file. Returns `nil` if the audio file does not contain an appropriate sampler chunk or could not be opened or is invalid.	number
.keyLow	The lowest key of the key range stored in the sampler chunk of the audio file. Returns `nil` if the audio file does not contain an appropriate sampler chunk or could not be opened or is invalid.	number
.keyHigh	The highest key of the key range stored in the sampler chunk of the audio file. Returns `nil` if the audio file does not contain an appropriate sampler chunk or could not be opened or is invalid.	number
.velLow	The lowest velocity of the velocity range stored in the sampler chunk of the audio file. Returns `nil` if the audio file does not contain an appropriate sampler chunk or could not be opened or is invalid.	number
.velHigh	The highest velocity of the velocity range stored in the sampler chunk of the audio file. Returns `nil` if the audio file does not contain an appropriate sampler chunk or could not be opened or is invalid.	number
.detune	The tune offset in cents stored in the sampler chunk of the audio file. Returns `nil` if the audio file does not contain an appropriate sampler chunk or could not be opened or is invalid.	number
.level	The level offset in dB stored in the sampler chunk of the audio file. Returns `nil` if the audio file does not contain an appropriate sampler chunk or could not be opened or is invalid.	number
.tempo	The tempo in bpm stored in a data chunk of the audio file. Returns `nil` if the audio file does not contain an appropriate data chunk or could not be opened or is invalid.	number
.beats	The number of beats stored in a data chunk of the audio file. Returns `nil` if the audio file does not contain an appropriate data chunk or could not be opened or is invalid.	number
.signature	A pair of values for the numerator and denominator of the signature stored in a data chunk of the audio file. The values are `nil` if the audio file does not contain an appropriate data chunk or could not be opened or is invalid.	number, number
.sampleStart	The position of the sample start in samples stored in a data chunk of the audio file. Returns `nil` if the audio file does not contain an appropriate data chunk or could not be opened or is invalid.	number
.sampleEnd	The position of the sample end in samples stored in a data chunk of the audio file. Returns `nil` if the audio file does not contain an appropriate data chunk or could not be opened or is invalid.	number
.loops	The loop start and end positions in samples stored in a data chunk of the audio file. The returned table is an array that contains tables with the fields loopStart and loopEnd for each loop. Returns `nil` if the audio file does not contain an appropriate data chunk or could not be opened or is invalid.	table

Example

```
-- open an audio file from HALion Sonic 1.0
fname = "vstsound://502B301A6C914CEDA5C7500DC890C4DC/.Samples/g:/projects/yamahacontentserver/download
/release/smtg/winds/Samples/DP060_FluteC3.wav"
af = AudioFile.open(fname)
loops = af.loops
-- print information from the audio file
if af.valid then
  print(fname, "opened.")
  print("Sample Rate: ", af.rate)
  print("Bit Depth: ", af.bits)
  print("Channels: ", af.channels)
  print("Sample Length: ", af.length)
  if loops then
    print("Loop Start: ", loops[1].loopStart)
    print("Loop End: ", loops[1].loopEnd)
  end
else
  print(fname, "does not exist.")
end
```

Jump to Top

Methods

analyzePitch

analyzePitch(callback, channel)

Description

Function to analyze the pitch of an audio file. You specify the audio file with the AudioFile object that is returned by the AudioFile.open function. The arguments `callback` and `channel` are optional. If called without a callback function, analyzePitch will be executed in the same thread. If called with a callback function as argument, analyzePitch will be executed in a separate, parallel thread. You can specify the channel to be analyzed with the `channel` argument. Without the `channel` argument multiple channels of an audio file will be summed before the pitch is analyzed. The callback function is called with the AudioFile object as the first and the `channel` as the second argument after the pitch has been analyzed. The results of analyzePitch are cashed as long as the corresponding AudioFile object exists. The function itself does not return any pitch information. You must use getPitch to obtain the analyzed pitch.

Available in: Controller.

Arguments

callback	Callback function that is called with the AudioFile object as argument after the pitch has been analyzed.	function, optional
channel	Use this to specify the channel of the audio file to be analyzed. Leave this empty or set this to 0 for summing all audio channels before they are analyzed.	number, optional

Example

```
channelNames = { [0] = "All", "Left", "Right" }

defineParameter( "Channel", nil, 0, channelNames)
defineParameter( "Start", nil, false, function() if Start then onStart() end end)
defineParameter( "Cancel", nil, false)

-- requires the Skylab content
path = "vstsound://724ACB205EFF46F885735D1B216C37AD/.AppData/Steinberg/Skylab/Sub Presets/Layer Presets
/Ambient Pads/Ambient Pad 01.vstpreset"
layer = loadPreset(path)

function onPitchAnalysisFinished(audioFile, channelNum)
    print("Progress: 100%")
    print(channelNames[channelNum].." channel(s) of "..audioFile.fileName.." analyzed.")
end

function onStart()
    zones = layer:findZones(true)
    for i, zone in ipairs(zones) do
        local samplePath = zone:getParameter("SampleOsc.Filename")
        print("File: "..samplePath)
        local afile = AudioFile.open(samplePath)
        afile:analyzePitch(onPitchAnalysisFinished, Channel)
        while afile:getPitchAnalysisProgress(Channel) < 1 do
            if Cancel then
                afile:cancelPitchAnalysis(Channel)
                break
            end
            local progressPercent = 100 * afile:getPitchAnalysisProgress(Channel)
            print(string.format("Progress: %2d%%", progressPercent))
            wait(2000)
        end
        if Cancel then
            Cancel = false
            print("Canceled!")
            break
        end
        local pitch = afile:getPitch(0, -1, Channel)
        print("Analyzed Pitch: "..pitch)
    end
    print("Done!")
    Start = false
end
```

Jump to Top

cancelPitchAnalysis

cancelPitchAnalysis(channel)

Description

Function to cancel a pitch analysis you started with analyzePitch. You specify the audio file with the AudioFile object that is returned by the AudioFile.open function. The `channel` argument specifies the channel of the audio file. The AudioFile object and the `channel` argument must match the call to analyzePitch.

Available in: Controller.

Arguments

channel	Use this to specify the channel of the audio file that is being analyzed.	number, optional

Example

```
channelNames = { [0] = "All", "Left", "Right" }

defineParameter( "Channel", nil, 0, channelNames)
defineParameter( "Start", nil, false, function() if Start then onStart() end end)
defineParameter( "Cancel", nil, false)

-- requires the Skylab content
path = "vstsound://724ACB205EFF46F885735D1B216C37AD/.AppData/Steinberg/Skylab/Sub Presets/Layer Presets
/Ambient Pads/Ambient Pad 01.vstpreset"
layer = loadPreset(path)

function onPitchAnalysisFinished(audioFile, channelNum)
    print("Progress: 100%")
    print(channelNames[channelNum].." channel(s) of "..audioFile.fileName.." analyzed.")
end

function onStart()
    zones = layer:findZones(true)
    for i, zone in ipairs(zones) do
        local samplePath = zone:getParameter("SampleOsc.Filename")
        print("File: "..samplePath)
        local afile = AudioFile.open(samplePath)
        afile:analyzePitch(onPitchAnalysisFinished, Channel)
        while afile:getPitchAnalysisProgress(Channel) < 1 do
            if Cancel then
                afile:cancelPitchAnalysis(Channel)
                break
            end
            local progressPercent = 100 * afile:getPitchAnalysisProgress(Channel)
            print(string.format("Progress: %2d%%", progressPercent))
            wait(2000)
        end
        if Cancel then
            Cancel = false
            print("Canceled!")
            break
        end
        local pitch = afile:getPitch(0, -1, Channel)
        print("Analyzed Pitch: "..pitch)
    end
    print("Done!")
    Start = false
end
```

Jump to Top

getOnsets

getOnsets(start, length, peakThreshold, sensThreshold, minLength)

Description

Function to analyze the onsets in an audio file. You specify the audio file with the AudioFile object that is returned by the AudioFile.open function. The onset analysis is performed on the sum of all channels in the audio file. The arguments `start` and `length` define the time range in the audio file to be analyzed. The `peakThreshold` and `sensThreshold` arguments determine the minimum level and the weight of an onset, the `minLength` argument determines the minimum duration between consecutive onsets.

Available in: Controller.

Arguments

start	The start position in samples.	number
length	The duration in samples. Set this to equal to or less than 0 to use all samples from the specified `start` to the end of the sample.	number
peakThreshold	The minimum level in decibels. Onsets with lower level are skipped. The value range is from -96 to 0.	number
sensThreshold	The minimum weight in percent. Onsets with lower weight are skipped. The value range is from 0 to 100.	number
minLength	The minimum duration between consecutive onsets in milliseconds. The value range is from 0 to 10000.	number

Return Values

Returns an array with the positions of the detected onsets.

Example

```
fname = "vstsound://271CB2CFA75E4295B1C273FA651FE11D/.Samples/g:/projects/yamahacontentserver/Download
/Release/ycj/ME_Waveform/Loop145/samples/Loop145_072(2).wav"

af = AudioFile.open(fname)

onsets = af:getOnsets(0, -1, -24, 0, 10)

for i, pos in ipairs(onsets) do
   print(i.." ".."Onset: "..pos)
end
```

Jump to Top

getPeak

getPeak(start, length, rms)

Description

Function to analyze the levels in an audio file. You specify the audio file with the AudioFile object that is returned by the `AudioFile.open` function. The arguments `start` and `length` define the range in the audio file to be analyzed. The `rms` argument determines whether the peak level or the RMS level of the specified range is returned.

Available in: Controller.

Arguments

start	The start position in samples.	number
length	The duration in samples. Set this to equal to or less than 0 to use all samples from the specified `start` to the end of the file.	number
rms	If this is set to 0, the peak level of the specified range will be returned. If this is set to a value above 0, the RMS level over the specified range will be calculated.	number

Return Values

Returns the level of the specifed range as a linear value. The example shows how to convert the value from linear to dB.

Example

```
function lin2db(lin)
  return 20 * math.log(lin) / math.log(10)
end
fname = "vstsound://F29C895D6D8E4D6C9BCBBA5198412192/.samples/Ambient Pad 01/Ambient Pad 01 - C3.tg3c"
af = AudioFile.open(fname)
-- analyze the peak level in the first 1000 samples
attpeak = af:getPeak(0, 1000, 0)
-- analyze the RMS level in the range from 1000 samples till the end of the file
susrms = af:getPeak(1000, -1, 1)
print("Attack Peak:", attpeak, "(", lin2db(attpeak), "dB )")
print("Sustain RMS:", susrms, "(", lin2db(susrms), "dB )")
```

Jump to Top

getPitch

getPitch(start, length, channel)

Description

Function to obtain the pitch of an audio file that has been analyzed with analyzePitch. The audio file you want to obtain the pitch from is specified with the AudioFile object that is returned by the AudioFile.open function. The arguments `start` and `length` define the range in the audio file that is used for obtaining the pitch. The `channel` argument specifies the channel of the audio file that was analyzed. The AudioFile object and the `channel` argument must match the call to analyzePitch. The function returns two values: The pitch as MIDI note number with decimals for cents and a boolean for voiced/unvoiced detection. If `length` is greater than 20 ms, the average of the pitches in the specified range is returned. If the audio file has not been analyzed in advance, getPitch returns `nil`.

Available in: Controller.

Arguments

start	The start position in samples.	number
length	The duration in samples. Set this to less than or equal to 0 to use all samples from the specified `start` to the end of the file.	number
channel	Use this to specify the audio channel that was analyzed.	number, optional

Return Values

Returns a tuple with two values:

- A float value representing the pitch as MIDI note number with decimals for cents,
- a boolean for voiced/unvoiced detection. The return value `true` means that a pitch was detected in the specified range.

If `length` is greater than 20 ms, the average of the pitches in the specified range is returned. If the audio file has not been analyzed in advance, getPitch returns `nil`.

Example

```
channelNames = { [0] = "All", "Left", "Right" }

defineParameter( "Channel", nil, 0, channelNames)
defineParameter( "Start", nil, false, function() if Start then onStart() end end)
defineParameter( "Cancel", nil, false)

-- requires the Skylab content
path = "vstsound://724ACB205EFF46F885735D1B216C37AD/.AppData/Steinberg/Skylab/Sub Presets/Layer Presets
/Ambient Pads/Ambient Pad 01.vstpreset"
layer = loadPreset(path)

function onPitchAnalysisFinished(audioFile, channelNum)
    print("Progress: 100%")
    print(channelNames[channelNum].." channel(s) of "..audioFile.fileName.." analyzed.")
end

function onStart()
    zones = layer:findZones(true)
    for i, zone in ipairs(zones) do
        local samplePath = zone:getParameter("SampleOsc.Filename")
        print("File: "..samplePath)
        local afile = AudioFile.open(samplePath)
        afile:analyzePitch(onPitchAnalysisFinished, Channel)
        while afile:getPitchAnalysisProgress(Channel) < 1 do
            if Cancel then
                afile:cancelPitchAnalysis(Channel)
                break
            end
            local progressPercent = 100 * afile:getPitchAnalysisProgress(Channel)
            print(string.format("Progress: %2d%%", progressPercent))
            wait(2000)
        end
        if Cancel then
            Cancel = false
            print("Canceled!")
            break
        end
        local pitch = afile:getPitch(0, -1, Channel)
        print("Analyzed Pitch: "..pitch)
    end
    print("Done!")
    Start = false
end
```

Jump to Top

getPitchAnalysisProgress

getPitchAnalysisProgress(channel)

Description

Function to monitor the progress of analyzePitch. You specify the audio file with the AudioFile object that is returned by the AudioFile.open function. The `channel` argument specifies the channel of the audio file. The AudioFile object and the `channel` argument must match the call to analyzePitch. The function returns the progress as a float value between 0 and 1.

Available in: Controller.

Arguments

channel	Use this to specify the channel of the audio file that is being analyzed.	number, optional

Return Values

Returns the progress as a float value between 0 and 1.

Example

```
channelNames = { [0] = "All", "Left", "Right" }

defineParameter( "Channel", nil, 0, channelNames)
defineParameter( "Start", nil, false, function() if Start then onStart() end end)
defineParameter( "Cancel", nil, false)

-- requires the Skylab content
path = "vstsound://724ACB205EFF46F885735D1B216C37AD/.AppData/Steinberg/Skylab/Sub Presets/Layer Presets
/Ambient Pads/Ambient Pad 01.vstpreset"
layer = loadPreset(path)

function onPitchAnalysisFinished(audioFile, channelNum)
    print("Progress: 100%")
    print(channelNames[channelNum].." channel(s) of "..audioFile.fileName.." analyzed.")
end

function onStart()
    zones = layer:findZones(true)
    for i, zone in ipairs(zones) do
        local samplePath = zone:getParameter("SampleOsc.Filename")
        print("File: "..samplePath)
        local afile = AudioFile.open(samplePath)
        afile:analyzePitch(onPitchAnalysisFinished, Channel)
        while afile:getPitchAnalysisProgress(Channel) < 1 do
            if Cancel then
                afile:cancelPitchAnalysis(Channel)
                break
            end
            local progressPercent = 100 * afile:getPitchAnalysisProgress(Channel)
            print(string.format("Progress: %2d%%", progressPercent))
            wait(2000)
        end
        if Cancel then
            Cancel = false
            print("Canceled!")
            break
        end
        local pitch = afile:getPitch(0, -1, Channel)
        print("Analyzed Pitch: "..pitch)
    end
    print("Done!")
    Start = false
end
```

Jump to Top