Audio Toolbox / AUAudioUnit

Class

AUAudioUnit

A class that defines a host’s interface to an audio unit.

iOS 9.0+ | iPadOS 9.0+ | Mac Catalyst 13.1+ | macOS 10.11+ | tvOS 9.0+ | visionOS 1.0+

```
class AUAudioUnit
```

Mentioned in

Adding Parallel Real-Time Threads to Audio Workgroups

Overview

Hosts can instantiate either version 3 or version 2 audio units with this class, and to some extent control whether an audio unit is instantiated in-process or in a separate extension process.

Version 3 audio units should subclass the `AUAudioUnit` class. Version 3 audio unit components can be registered in the following ways:

- Package the component into an app extension containing an `AudioComponents` `Info.plist` entry. The principal class must conform to the `AUAudioUnitFactory` protocol, which will typically instantiate an `AUAudioUnit` subclass.
- Call the `registerSubclass(_:as:name:version:)` method to associate a component description with an `AUAudioUnit` subclass. Use the convention `<manufacturer name>: <audio unit name>` when naming your audio unit component.

Version 2 audio units should subclass the `AUAudioUnitV2Bridge` class instead. Version 2 audio unit components can be registered in the following ways:

- Package the component into a component bundle containing an `AudioComponents` `Info.plist` entry, referring to an `AudioComponentFactoryFunction` function.

- Call the `AudioComponentRegister` function to associate a component description with an `AudioComponentFactoryFunction` function.

A host does not need to be aware of the concrete `AUAudioUnit` subclass that is being instantiated. The `init(componentDescription:options:)` method ensures that the proper subclass is used.

Important

When using the `AUAudioUnit` class with a version 2 audio unit, or the C `AudioComponent` and `AudioUnit` APIs with a version 3 audio unit, all major pieces of functionality are bridged between the two APIs. When applicable, this document references the version 2 API equivalent of each version 3 method or property.

Topics

Creating an Audio Unit

`convenience init(componentDescription: AudioComponentDescription) throws`

Synchronously initializes a new audio unit object.

`init(componentDescription: AudioComponentDescription, options: AudioComponentInstantiationOptions) throws`

Synchronously initializes a new audio unit object.

`class func instantiate(with: AudioComponentDescription, options: AudioComponentInstantiationOptions, completionHandler: (AUAudioUnit?, (any Error)?) -> Void)`

Asynchronously creates an audio unit instance.

Returning the Audio Busses

`var inputBusses: AUAudioUnitBusArray`

An array containing the audio unit’s input connection points.

`var outputBusses: AUAudioUnitBusArray`

An array containing the audio unit’s output connection points.

Customizing the Audio Unit Behavior

These methods and properties are only of interest to audio unit subclasses.

`class func registerSubclass(AnyClass, as: AudioComponentDescription, name: String, version: UInt32)`

Registers an audio unit subclass.

`func shouldChange(to: AVAudioFormat, for: AUAudioUnitBus) -> Bool`

This is called when you set the format on a bus.

`func setRenderResourcesAllocated(Bool)`

Sets the Boolean value of the `renderResourcesAllocated` property.

`var internalRenderBlock: AUInternalRenderBlock`

The block which you must provide, via a getter, in order to implement rendering.

`var midiOutputBufferSizeHint: Int`

`typealias AUInternalRenderBlock`

A block to render the audio unit.

Querying Parameters

`var parameterTree: AUParameterTree?`

An audio unit's parameters, organized in a tree hierarchy.

`var allParameterValues: Bool`

Special read-only property for KVO.

`func parametersForOverview(withCount: Int) -> [NSNumber]`

Returns the audio unit's most important parameters.

Providing Data to the Host

`var musicalContextBlock: AUHostMusicalContextBlock?`

A callback to the host for musical context information.

`var transportStateBlock: AUHostTransportStateBlock?`

A callback to the host for transport state information.

`var contextName: String?`

Information about the host context in which the audio unit is connected, for display in the audio unit’s view.

`var supportsMPE: Bool`

A Boolean value that indicates whether the audio unit supports multi-dimensional polyphonic expression.

`typealias AUHostMusicalContextBlock`

A block through which hosts provide musical tempo, time signature, and beat position.

`typealias AUHostTransportStateBlock`

A block through which hosts provide information about their transport state.

Managing MIDI Events

`var isMusicDeviceOrEffect: Bool`

Specifies whether an audio unit responds to MIDI events.

`var virtualMIDICableCount: Int`

The number of virtual MIDI cables implemented by a music device or effect.

`var scheduleMIDIEventBlock: AUScheduleMIDIEventBlock?`

A block used to schedule MIDI events.

`var midiOutputEventBlock: AUMIDIOutputEventBlock?`

`var midiOutputNames: [String]`

The names of the MIDI outputs.

`typealias AUScheduleMIDIEventBlock`

A block to schedule MIDI events.

`typealias AUMIDIOutputEventBlock`

Managing Presets

`var fullState: [String : Any]?`

A persistable snapshot of the audio unit’s properties and parameters, suitable for saving as a user preset.

`var fullStateForDocument: [String : Any]?`

A persistable snapshot of the audio unit's properties and parameters, suitable for saving in a user's document.

```
var factoryPresets: [AUAudioUnitPreset]?
```

A collection of presets provided by the audio unit's developer.

```
var currentPreset: AUAudioUnitPreset?
```

The audio unit's last-selected preset.

```
var supportsUserPresets: Bool
```

```
var userPresets: [AUAudioUnitPreset]
```

```
func saveUserPreset(AUAudioUnitPreset) throws
```

```
func deleteUserPreset(AUAudioUnitPreset) throws
```

```
func presetState(for: AUAudioUnitPreset) throws -> [String : Any]
```

Managing the Render Cycle

```
func allocateRenderResources() throws
```

Allocates resources required to render audio.

```
func deallocateRenderResources()
```

Deallocates resources required to render audio.

```
func reset()
```

Resets transitory rendering state to its initial state.

```
var renderResourcesAllocated: Bool
```

Determines whether the audio unit has allocated render resources.

```
var renderBlock: AURenderBlock
```

The block that hosts use to ask the audio unit to render audio.

```
var scheduleParameterBlock: AUScheduleParameterBlock
```

The block that hosts use to schedule parameters.

```
var maximumFramesToRender: AUAudioFrameCount
```

The maximum number of frames that the audio unit can render at once.

```
func token(byAddingRenderObserver: AURenderObserver) -> Int
```

Adds a block to be called on each render cycle.

`func removeRenderObserver(Int)`

Removes an observer block previously added to the render cycle.

`typealias AURenderObserver`

A block called when an audio unit renders audio.

Messaging Channels

`func messageChannel(for: String) -> any AUMessageChannel`

Returns an object for bidirectional communication between an audio unit and its host.

`protocol AUMessageChannel`

A specification for a bidirectional communication message channel.

Optimizing Performance

`var latency: TimeInterval`

The audio unit’s processing latency, in seconds.

`var tailTime: TimeInterval`

The audio unit’s tail time, in seconds.

`var renderQuality: Int`

Provides a trade-off between rendering quality and CPU load.

`var shouldBypassEffect: Bool`

Determines whether an effect should route input directly to output, without any processing.

`var canProcessInPlace: Bool`

Determines whether an audio unit can process in place.

`var isRenderingOffline: Bool`

Communicates to an audio unit that it is rendering offline.

Describing the Audio Unit

`var componentDescription: AudioComponentDescription`

The component description with which the audio unit was created.

`var component: AudioComponent`

The component found in the component description with which the audio unit was created.

`var componentName: String?`

The audio unit's component's name.

`var componentVersion: UInt32`

The audio unit's component's version.

`var audioUnitName: String?`

The audio unit's name, derived from the component's name.

`var audioUnitShortName: String?`

`var manufacturerName: String?`

The manufacturer's name, derived from the component's name.

Configuring the Channel Capabilities

`var channelCapabilities: [NSNumber]?`

Expresses valid combinations of input and output channels.

`var channelMap: [NSNumber]?`

`func profileState(forCable: UInt8, channel: MIDIChannelNumber) -> MIDICIProfileState`

`func enable(MIDICIProfile, cable: UInt8, onChannel: MIDIChannelNumber) throws`

`func disableProfile(MIDICIProfile, cable: UInt8, onChannel: MIDIChannelNumber) throws`

`var profileChangedBlock: AUMIDICIProfileChangedBlock?`

Configuring the Device

`var deviceID: AUAudioObjectID`

Gets the I/O hardware device.

`func setDeviceID(AUAudioObjectID) throws`

Sets the I/O hardware device.

`var canPerformInput: Bool`

Determines whether the I/O device can perform input.

`var canPerformOutput: Bool`

Determines whether the I/O device can perform output.

`var isInputEnabled: Bool`

A flag enabling audio input from the unit.

`var isOutputEnabled: Bool`

A flag enabling audio output from the unit.

`var inputHandler: AUInputHandler?`

The block that the output unit will call to notify when input is available.

`var outputProvider: AURenderPullInputBlock?`

The block that the output unit will call to get audio to send to the output.

`var deviceInputLatency: TimeInterval`

The audio device's input latency, in seconds.

`var deviceOutputLatency: TimeInterval`

The audio devic's output latency, in seconds.

`func startHardware() throws`

Starts the audio hardware.

`func stopHardware()`

Stops the audio hardware.

`typealias AURenderPullInputBlock`

A block to supply audio input to a render block.

Configuring the User Interface

`var providesUserInterface: Bool`

A Boolean that indicates whether the audio unit provides a user interface, normally in the form of a view controller.

`func supportedViewConfigurations([AUAudioUnitViewConfiguration]) -> IndexSet`

`func select(AUAudioUnitViewConfiguration)`

Getting the Runtime Behavior

`var isRunning: Bool`

`var isLoadedInProcess: Bool`

Constants

AUEventSampleTime

Expresses time as a sample count.

`enum AUAudioUnitBusType`

`struct AUHostTransportStateFlags`

`enum AURenderEventType`

`typealias AURenderBlock`

A block to render the audio unit.

`typealias AUInputHandler`

A block to notify the host of an I/O unit that an input is available.

Getting the Audio Unit Presets

`var kAUPresetNumberKey: String`

`var kAUPresetCPULoadKey: String`

`var kAUPresetDataKey: String`

`var kAUPresetElementNameKey: String`

`var kAUPresetExternalFileRefs: String`

`var kAUPresetMASDataKey: String`

`var kAUPresetManufacturerKey: String`

`var kAUPresetNameKey: String`

`var kAUPresetPartKey: String`

If present, distinguishes a global preset that is set on the global scope from a part-based preset that is set on the part scope. The value of this key is defined by the audio unit it applies to.

`var kAUPresetRenderQualityKey: String`

`var kAUPresetSubtypeKey: String`

`var kAUPresetTypeKey: String`

`var kAUPresetVSTDataKey: String`

VST state from a VST “bank.”

`var kAUPresetVSTPresetKey: String`

VST state from a VST “preset.”

`var kAUPresetVersionKey: String`

Instance properties

`var audioUnitMIDIProtocol: MIDIProtocolID`

`var hostMIDIProtocol: MIDIProtocolID`

`var midiOutputEventListBlock: AUMIDIEventListBlock?`

`var migrateFromPlugin: [Any]`

`var scheduleMIDIEventListBlock: AUMIDIEventListBlock?`

Instance Methods

`func requestViewController(completionHandler: (UIViewController?) -> Void)`

Requests an audio unit’s custom view controller.

Instance Properties

`var intendedSpatialExperience: any SpatialAudioExperience`

The AUAudioUnit’s intended spatial audio experience.

Relationships

Inherits From

`NSObject`

Inherited By

`AUAudioUnitV2Bridge`

Conforms To

`CVarArg`
`CustomDebugStringConvertible`
`CustomStringConvertible`
`Equatable`
`Hashable`
`NSObjectProtocol`

See Also

Audio Units

Creating an audio unit extension

Build an extension by using an Xcode template.

Creating custom audio effects

Add custom audio-effect processing to apps like Logic Pro X and GarageBand by creating Audio Unit (AU) plug-ins.

Incorporating Audio Effects and Instruments

Add custom audio processing and MIDI instruments to your app by hosting Audio Unit (AU) plug-ins.

Debugging Out-of-Process Audio Units on Apple Silicon

Connect to out-of-process audio units using the Xcode debugger.

`class AUAudioUnitBus`

A class that defines an input or output connection point on an audio unit.

`class AUAudioUnitBusArray`

A class that defines a container for an audio unit’s input or output busses.

`class AUAudioUnitPreset`

A class that describes an interface for custom parameter settings provided by the audio unit developer.

`class AUAudioUnitV2Bridge`

A class that wraps a version 2 audio unit as version 3 audio unit.

```
func AudioUnitExtensionCopyComponentList(CFString) -> Unmanaged<CFArray
>?
```

Returns the component registrations for a given audio unit extension.

`func AudioUnitExtensionSetComponentList(CFString, CFArray?) -> OSStatus`

Allows the implementor of an audio unit extension to dynamically modify the list of component registrations for the extension.

`protocol AUAudioUnitFactory`

An object that creates a version 3 audio unit.