Class

AVAssetWriter

An object that writes media data to a container file.

iOS 4.1+ | iPadOS 4.1+ | Mac Catalyst 13.1+ | macOS 10.7+ | tvOS 9.0+ | visionOS 1.0+

```
class AVAssetWriter
```

Overview

You use an asset writer to write media to file formats such as the QuickTime movie file format and MPEG-4 file format. An asset writer automatically supports interleaving media data from concurrent tracks for efficient playback and storage. It can reencode media samples it writes to the output file, and may also write collections of metadata to the output file.

Important

An asset writer is a single-use object that writes one output file. Create multiple asset writer instances if your app requires writing multiple output files.

Topics

Creating an asset writer

```
convenience init(url: URL, fileType: AVFileType) throws
```

Returns a new object that writes media data to a container file at the output URL.

```
init(outputURL: URL, fileType: AVFileType) throws
```

Creates an object that writes media data to a container file at the output URL.

```
init(contentType: UTType)
```

Creates an object that outputs segment data in a specified container format.

Configuring inputs

```
var inputs: [AVAssetWriterInput]
```

The inputs an asset writer contains.

```
var availableMediaTypes: [AVMediaType]
```

The media types the asset writer supports adding as inputs.

```
func canApply(outputSettings: [String : Any]?, forMediaType: AVMedia
Type) -> Bool
```

Determines whether the output file format supports the output settings for a specific media type.

```
func canAdd(AVAssetWriterInput) -> Bool
```

Determines whether the asset writer supports adding the input.

```
func add(AVAssetWriterInput)
```

Adds an input to an asset writer.

Configuring input receivers

```
func inputReceiver(for: AVAssetWriterInput) -> sending AVAssetWriter
Input.SampleBufferReceiver
```

Attaches the input to the writer and returns an input receiver for writing sample buffers.

```
func inputCaptionReceiver(for: AVAssetWriterInput) -> sending AVAsset
WriterInput.CaptionReceiver
```

Attaches the input to the writer and returns an input receiver for writing caption data.

```
func inputCaptionReceiverRequestingMultiPass(for: AVAssetWriterInput) -
> sending (AVAssetWriterInput.CaptionReceiver, AVAssetWriterInput.Multi
PassController)
```

Attaches the input to the writer and returns a tuple with an input receiver for writing caption data, and an associated multi pass controller.

```
func inputMetadataReceiver(for: AVAssetWriterInput) -> sending AVAsset
WriterInput.MetadataReceiver
```

Attaches the input to the writer and returns an input receiver for writing timed metadata group.

```
func inputMetadataReceiverRequestingMultiPass(for: AVAssetWriterInput)
-> sending (AVAssetWriterInput.MetadataReceiver, AVAssetWriterInput.
MultiPassController)
```

Attaches the input to the writer and returns a tuple with an input receiver for writing timed metadata group, and an associated multi pass controller.

```
func inputPixelBufferReceiver(for: AVAssetWriterInput, pixelBuffer
Attributes: CVPixelBufferCreationAttributes?) -> sending AVAssetWriter
Input.PixelBufferReceiver
```

Attaches the input to the writer and returns an input receiver for writing pixel buffers.

```
func inputPixelBufferReceiverRequestingMultiPass(for: AVAssetWriter
Input, pixelBufferAttributes: CVPixelBufferCreationAttributes?) ->
sending (AVAssetWriterInput.PixelBufferReceiver, AVAssetWriterInput.
MultiPassController)
```

Attaches the input to the writer and returns a tuple with an input receiver for writing pixel buffers, and an associated multi pass controller.

```
func inputReceiverRequestingMultiPass(for: AVAssetWriterInput) ->
sending (AVAssetWriterInput.SampleBufferReceiver, AVAssetWriterInput.
MultiPassController)
```

Attaches the input to the writer and returns a tuple with an input receiver for writing sample buffers, and an associated multi pass controller.

```
func inputTaggedPixelBufferGroupReceiver(for: AVAssetWriterInput, pixel
BufferAttributes: CVPixelBufferCreationAttributes?) -> sending AVAsset
WriterInput.TaggedPixelBufferGroupReceiver
```

Attaches the input to the writer and returns an input receiver for writing tagged pixel buffers.

```
func inputTaggedPixelBufferGroupReceiverRequestingMultiPass(for:
AVAssetWriterInput, pixelBufferAttributes: CVPixelBufferCreation
Attributes?) -> sending (AVAssetWriterInput.TaggedPixelBufferGroup
Receiver, AVAssetWriterInput.MultiPassController)
```

Attaches the input to the writer and returns a tuple with an input receiver for writing tagged pixel buffers, and an associated multi pass controller.

Configuring input groups

`var inputGroups: [AVAssetWriterInputGroup]`

The input groups an asset writer contains.

`func canAdd(AVAssetWriterInputGroup) -> Bool`

Determines whether the asset writer supports adding the input group.

`func add(AVAssetWriterInputGroup)`

Adds an input group to an asset writer.

Configuring output

`var metadata: [AVMetadataItem]`

An array of metadata items to write to the output file.

`var shouldOptimizeForNetworkUse: Bool`

A Boolean value that indicates whether to write the output file to make it more suitable for playback over a network.

`var directoryForTemporaryFiles: URL?`

A directory to contain temporary files that the export process generates.

Configuring fragment output

`var movieFragmentInterval: CMTime`

The interval at which to write movie fragments.

`var initialMovieFragmentInterval: CMTime`

The interval at which to write the initial movie fragment.

`var initialMovieFragmentSequenceNumber: Int`

The sequence number of the initial movie fragment.

`var producesCombinableFragments: Bool`

A Boolean value that indicates whether the asset writer outputs movie fragments suitable for combining with others.

`var overallDurationHint: CMTime`

A hint of the final duration of the output file.

`var movieTimeScale: CMTimeScale`

The time scale of the movie.

Managing writing sessions

`func start() throws`

Prepares the writer to write media data to its output file.

`func startWriting() -> Bool`

Tells the writer to start writing its output.

`func startSession(atSourceTime: CMTime)`

Starts an asset-writing session.

`func endSession(atSourceTime: CMTime)`

Finishes an asset-writing session.

`func finishWriting(completionHandler: () -> Void)`

Marks all unfinished inputs as finished and completes the writing of the output file.

`func cancelWriting()`

Cancels the creation of the output file.

~~`func finishWriting() -> Bool`~~

Completes the writing of the output file.

Deprecated

Inspecting writing status

`var status: AVAssetWriter.Status`

The status of writing samples to the output file.

`enum Status`

Values that indicate the state of an asset writer.

`var error: (any Error)?`

An error object that describes an asset-writing failure.

Configuring segment writing

`var delegate: (any AVAssetWriterDelegate)?`

A delegate object that responds to asset-writing events.

`protocol AVAssetWriterDelegate`

A delegate protocol that defines the methods to implement to respond to asset-writing events.

`var preferredOutputSegmentInterval: CMTime`

The interval of output segments that you prefer.

`var initialSegmentStartTime: CMTime`

The start time of the initial segment.

`var outputFileTypeProfile: AVFileTypeProfile?`

A profile for the output file type.

`func flushSegment()`

Closes the current segment and outputs it to a delegate method.

Accessing output settings

`var outputURL: URL`

The location of the container file that the writer outputs.

`var outputFileType: AVFileType`

The type of container file that the writer outputs.

Relationships

Inherits From

`NSObject`

Conforms To

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

`NSObjectProtocol`

See Also

Media writing

{} Converting projected video to Apple Projected Media Profile

Convert content with equirectangular or half-equirectangular projection to APMP.

{} Converting side-by-side 3D video to multiview HEVC and spatial video

Create video content for visionOS by converting an existing 3D HEVC file to a multiview HEVC format, optionally adding spatial metadata to create a spatial video.

{} Writing fragmented MPEG-4 files for HTTP Live Streaming

Create an HTTP Live Streaming presentation by turning a movie file into a sequence of fragmented MPEG-4 files.

Creating spatial photos and videos with spatial metadata

Add spatial metadata to stereo photos and videos to create spatial media for viewing on Apple Vision Pro.

Tagging media with video color information

Inspect and set video color space information when writing and transcoding media.

Evaluating an app's video color

Check color reproduction for a video in your app by using test patterns, video test equipment, and light-measurement instruments.

`class AVOutputSettingsAssistant`

An object that builds audio and video output settings dictionaries.

`class AVAssetWriterInput`

An object that appends media samples to a track in an asset writer's output file.

`class AVAssetWriterInputPixelBufferAdaptor`

An object that appends video samples to an asset writer input.

`class AVAssetWriterInputTaggedPixelBufferGroupAdaptor`

An object that appends tagged buffer groups to an asset writer input.

`class AVAssetWriterInputMetadataAdaptor`

An object that appends timed metadata groups to an asset writer input.

`class AVAssetWriterInputGroup`

A group of inputs with tracks that are mutually exclusive to each other for playback or processing.