```
import AVFoundation.AVAnimation
import AVFoundation.AVAsset
import AVFoundation.AVAssetCache
import
AVFoundation.AVAssetDownloadStorageManage
r
import AVFoundation.AVAssetDownloadTask
import AVFoundation.AVAssetExportSession
import AVFoundation.AVAssetImageGenerator
import
AVFoundation.AVAssetPlaybackAssistant
import AVFoundation.AVAssetReader
import AVFoundation.AVAssetReaderOutput
import AVFoundation.AVAssetResourceLoader
import AVFoundation.AVAssetSegmentReport
import AVFoundation.AVAssetTrack
import AVFoundation.AVAssetTrackGroup
import AVFoundation.AVAssetTrackSegment
import AVFoundation.AVAssetVariant
import AVFoundation.AVAssetWriter
import AVFoundation.AVAssetWriterInput
import
AVFoundation.AVAsynchronousKeyValueLoadin
g
import AVFoundation.AVAudioBuffer
import AVFoundation.AVAudioChannelLayout
import
AVFoundation.AVAudioConnectionPoint
import AVFoundation.AVAudioConverter
import AVFoundation.AVAudioEngine
import
AVFoundation.AVAudioEnvironmentNode
import AVFoundation.AVAudioFile
import AVFoundation.AVAudioFormat
import AVFoundation.AVAudioIONode
import AVFoundation.AVAudioMix
import AVFoundation.AVAudioMixerNode
import AVFoundation.AVAudioMixing
import AVFoundation.AVAudioNode
import AVFoundation.AVAudioPlayer
import AVFoundation.AVAudioPlayerNode
import
AVFoundation.AVAudioProcessingSettings
import AVFoundation.AVAudioRecorder
import AVFoundation.AVAudioRoutingArbiter
import AVFoundation.AVAudioSequencer
import AVFoundation.AVAudioSession
import
AVFoundation.AVAudioSessionDeprecated
import AVFoundation.AVAudioSessionRoute
import AVFoundation.AVAudioSessionTypes
import AVFoundation.AVAudioSettings
import AVFoundation.AVAudioTime
import AVFoundation.AVAudioTypes
import AVFoundation.AVAudioUnit
import AVFoundation.AVAudioUnitComponent
import AVFoundation.AVAudioUnitDelay
import AVFoundation.AVAudioUnitDistortion
import AVFoundation.AVAudioUnitEQ
import AVFoundation.AVAudioUnitEffect
import AVFoundation.AVAudioUnitGenerator
import
AVFoundation.AVAudioUnitMIDIInstrument
import AVFoundation.AVAudioUnitReverb
import AVFoundation.AVAudioUnitSampler
import AVFoundation.AVAudioUnitTimeEffect
import AVFoundation.AVAudioUnitTimePitch
import AVFoundation.AVAudioUnitVarispeed
import AVFoundation.AVBase
import
AVFoundation.AVCameraCalibrationData
import AVFoundation.AVCaption
import
AVFoundation.AVCaptionConversionValidator
import
AVFoundation.AVCaptionFormatConformer
import AVFoundation.AVCaptionGroup
import AVFoundation.AVCaptionGrouper
import AVFoundation.AVCaptionRenderer
import AVFoundation.AVCaptionSettings
import
AVFoundation.AVCaptureAudioDataOutput
import
AVFoundation.AVCaptureAudioPreviewOutput
import AVFoundation.AVCaptureControl
import
AVFoundation.AVCaptureDataOutputSynchroni
zer
import
AVFoundation.AVCaptureDepthDataOutput
import
AVFoundation.AVCaptureDeskViewApplication
import AVFoundation.AVCaptureDevice
import AVFoundation.AVCaptureFileOutput
import AVFoundation.AVCaptureIndexPicker
import AVFoundation.AVCaptureInput
import
AVFoundation.AVCaptureMetadataOutput
import AVFoundation.AVCaptureOutput
import AVFoundation.AVCaptureOutputBase
import AVFoundation.AVCapturePhotoOutput
import AVFoundation.AVCaptureReactions
import AVFoundation.AVCaptureSession
import
AVFoundation.AVCaptureSessionPreset
import AVFoundation.AVCaptureSlider
import
AVFoundation.AVCaptureStillImageOutput
import
AVFoundation.AVCaptureSystemExposureBiasS
lider
import
AVFoundation.AVCaptureSystemPressure
import
AVFoundation.AVCaptureSystemZoomSlider
import
AVFoundation.AVCaptureVideoDataOutput
import
AVFoundation.AVCaptureVideoPreviewLayer
import AVFoundation.AVComposition
import AVFoundation.AVCompositionTrack
import
AVFoundation.AVCompositionTrackSegment
import AVFoundation.AVContentKeySession
import AVFoundation.AVContinuityDevice
import AVFoundation.AVDepthData
import AVFoundation.AVError
import
AVFoundation.AVExternalStorageDevice
import AVFoundation.AVFAudio
import AVFoundation.AVFCapture
import AVFoundation.AVFCore
import AVFoundation.AVGeometry
import AVFoundation.AVMIDIPlayer
import AVFoundation.AVMediaFormat
import AVFoundation.AVMediaSelection
import AVFoundation.AVMediaSelectionGroup
import AVFoundation.AVMetadataFormat
import AVFoundation.AVMetadataIdentifiers
import AVFoundation.AVMetadataItem
import AVFoundation.AVMetadataObject
import AVFoundation.AVMetrics
import AVFoundation.AVMovie
import AVFoundation.AVMovieTrack
import
AVFoundation.AVOutputSettingsAssistant
import AVFoundation.AVPlaybackCoordinator
import AVFoundation.AVPlayer
import
AVFoundation.AVPlayerInterstitialEventCon
troller
import AVFoundation.AVPlayerItem
import
AVFoundation.AVPlayerItemIntegratedTimeli
ne
import
AVFoundation.AVPlayerItemMediaDataCollect
or
import AVFoundation.AVPlayerItemOutput
import
AVFoundation.AVPlayerItemProtectedContent
Additions
import AVFoundation.AVPlayerItemTrack
import AVFoundation.AVPlayerLayer
import AVFoundation.AVPlayerLooper
```

```
import
AVFoundation.AVPlayerMediaSelectionCriter
ia
import AVFoundation.AVPlayerOutput
import
AVFoundation.AVPortraitEffectsMatte
import
AVFoundation.AVQueuedSampleBufferRenderin
g
import
AVFoundation.AVRenderedCaptionImage
import AVFoundation.AVRouteDetector
import
AVFoundation.AVSampleBufferAudioRenderer
import
AVFoundation.AVSampleBufferDisplayLayer
import
AVFoundation.AVSampleBufferGenerator
import
AVFoundation.AVSampleBufferRenderSynchron
izer
import
AVFoundation.AVSampleBufferVideoRenderer
import AVFoundation.AVSampleCursor
import
AVFoundation.AVSemanticSegmentationMatte
import AVFoundation.AVSynchronizedLayer
import AVFoundation.AVTextStyleRule
import AVFoundation.AVTime
import AVFoundation.AVTimedMetadataGroup
import AVFoundation.AVUtilities
import AVFoundation.AVVideoCompositing
import AVFoundation.AVVideoComposition
```

```swift
import
AVFoundation.AVVideoPerformanceMetrics
import AVFoundation.AVVideoSettings
import CoreGraphics
import CoreMedia
import Dispatch
import Foundation
import _Concurrency
import _StringProcessing
import _SwiftConcurrencyShims

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
public class AVAnyAsyncProperty :
CustomStringConvertible, @unchecked
Sendable {

    /// A textual representation of this
instance.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(describing:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `description` property for types
that conform to
    /// `CustomStringConvertible`:
    ///
    ///     struct Point:
CustomStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var description: String {
    ///             return "(\(x), \(y))"
    ///         }
    ///     }
    ///
    ///     let p = Point(x: 21, y: 30)
    ///     let s = String(describing: p)
    ///     print(s)
    ///     // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `description`
property.
    public var description: String {
get }
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
public class AVAsyncProperty<Root, Value>
: AVPartialAsyncProperty<Root> {
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVAsyncProperty {

    @frozen public enum Status {

        /// property has not been loaded
        case notYetLoaded
        /// property is being loaded
        case loading

        /// property already loaded,
value is included
        case loaded(Value)

        /// property failed to load,
error is included
        case failed(NSError)
    }
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVAsyncProperty.Status :
Equatable where Value : Equatable {

    /// Returns a Boolean value
indicating whether two values are equal.
    ///
    /// Equality is the inverse of
inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is
`false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to
compare.
    public static func == (lhs:
AVAsyncProperty<Root, Value>.Status, rhs:
```

```
AVAsyncProperty<Root, Value>.Status) ->
Bool
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVAsyncProperty.Status :
Sendable where Value : Sendable {
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVAsyncProperty.Status :
CustomStringConvertible {

    /// A textual representation of this
instance.
    ///
    /// Calling this property directly is
discouraged. Instead, convert an
    /// instance of any type to a string
by using the `String(describing:)`
    /// initializer. This initializer
works with any type, and uses the custom
    /// `description` property for types
that conform to
    /// `CustomStringConvertible`:
    ///
    ///     struct Point:
CustomStringConvertible {
    ///         let x: Int, y: Int
    ///
    ///         var description: String {
```

```
    ///                 return "(\(x), \(y))"
    ///             }
    ///         }
    ///
    ///         let p = Point(x: 21, y: 30)
    ///         let s = String(describing: p)
    ///         print(s)
    ///         // Prints "(21, 30)"
    ///
    /// The conversion of `p` to a string
in the assignment to `s` uses the
    /// `Point` type's `description`
property.
    public var description: String {
get }
}

@available(macOS 15, iOS 18, tvOS 18,
watchOS 11, visionOS 2, *)
public struct
AVMergedMetrics<MetricEvent1,
MetricEvent2, each MetricEventPack> :
AsyncSequence where MetricEvent1 :
AVMetricEvent, MetricEvent2 :
AVMetricEvent, repeat each
MetricEventPack : AVMetricEvent {

    /// Creates the asynchronous iterator
that produces elements of this
    /// asynchronous sequence.
    ///
    /// - Returns: An instance of the
`AsyncIterator` type used to produce
```

```
    /// elements of the asynchronous
sequence.
    public func makeAsyncIterator() ->
AVMergedMetrics<MetricEvent1,
MetricEvent2, repeat each
MetricEventPack>.AsyncIterator

    /// The type of element produced by
this asynchronous sequence.
    public typealias Element =
(AVMetricEvent, any
AVMetricEventStreamPublisher)

    /// The type of asynchronous iterator
that produces elements of this
    /// asynchronous sequence.
    public struct AsyncIterator :
AsyncIteratorProtocol {

        /// Asynchronously advances to
the next element and returns it, or ends
the
        /// sequence if there is no next
element.
        ///
        /// - Returns: The next element,
if it exists, or `nil` to signal the end
of
        ///   the sequence.
        public mutating func next() async
throws -> (AVMetricEvent, any
AVMetricEventStreamPublisher)?
```

```
        @available(iOS 18, tvOS 18, 
watchOS 11, visionOS 2, macOS 15, *)
        public typealias Element = 
(AVMetricEvent, any 
AVMetricEventStreamPublisher)
    }
}

@available(macOS 15, iOS 18, tvOS 18, 
watchOS 11, visionOS 2, *)
public struct AVMetrics<MetricEvent> : 
AsyncSequence, @unchecked Sendable where 
MetricEvent : AVMetricEvent {

    /// Creates the asynchronous iterator 
that produces elements of this
    /// asynchronous sequence.
    ///
    /// - Returns: An instance of the 
`AsyncIterator` type used to produce
    /// elements of the asynchronous 
sequence.
    public func makeAsyncIterator() -> 
AVMetrics<MetricEvent>.AsyncIterator

    /// The type of element produced by 
this asynchronous sequence.
    public typealias Element = 
MetricEvent

    /// The type of asynchronous iterator 
that produces elements of this
    /// asynchronous sequence.
    public struct AsyncIterator :
AsyncIteratorProtocol {

        /// Asynchronously advances to
the next element and returns it, or ends
the
        /// sequence if there is no next
element.
        ///
        /// - Returns: The next element,
if it exists, or `nil` to signal the end
of
        ///   the sequence.
        public mutating func next() async
throws -> MetricEvent?

        @available(iOS 18, tvOS 18,
watchOS 11, visionOS 2, macOS 15, *)
        public typealias Element =
MetricEvent
    }

    public func
chronologicalMerge<OtherSecondMetric,
each MetricEventPack>(with secondMetric:
AVMetrics<OtherSecondMetric>, _ metrics:
repeat AVMetrics<each MetricEventPack>)
-> AVMergedMetrics<MetricEvent,
OtherSecondMetric, repeat each
MetricEventPack> where
OtherSecondMetric : AVMetricEvent, repeat
each MetricEventPack : AVMetricEvent
}
```

```
@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
public class AVPartialAsyncProperty<Root>
: AVAnyAsyncProperty {

    override public var description:
String { get }
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVPartialAsyncProperty where
Root : AVMetadataItem {

    /**
     Provides a dictionary of the
additional attributes.
     */
    public static var extraAttributes:
AVAsyncProperty<Root,
[AVMetadataExtraAttributeKey : Any]?> {
get }

    /**
     Provides the value of the metadata
item.
     */
    public static var value:
AVAsyncProperty<Root, (any NSCopying &
NSObjectProtocol)?> { get }
}
```

```
@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVPartialAsyncProperty where
Root : AVMetadataItem {

    /**
            Provides the value of the
metadata item as a string.

            Will be nil if the value cannot
be represented as a string.
            */
    public static var stringValue:
AVAsyncProperty<Root, String?> { get }

    /**
            Provides the value of the
metadata item as an NSNumber.

            Will be nil if the value cannot
be represented as a number.
            */
    public static var numberValue:
AVAsyncProperty<Root, NSNumber?> { get }

    /**
            Provides the value of the
metadata item as an Date.

            Will be nil if the value cannot
be represented as a date.
            */
    public static var dateValue:
```

```
AVAsyncProperty<Root, Date?> { get }

    /**
     Provides the raw bytes of the value
of the metadata item.
     */
    public static var dataValue:
AVAsyncProperty<Root, Data?> { get }
}

@available(macOS 12, iOS 15, watchOS 8,
visionOS 1, *)
@available(tvOS, unavailable)
extension AVPartialAsyncProperty where
Root : AVMutableMovie {

    /**
     Provides the array of
AVMutableMovieTracks contained by the
mutable movie.
     */
    public static var tracks:
AVAsyncProperty<Root,
[AVMutableMovieTrack]> { get }
}

@available(macOS 12, iOS 15, watchOS 8,
visionOS 1, *)
@available(tvOS, unavailable)
extension AVPartialAsyncProperty where
Root : AVFragmentedMovie {

    /**
```

```
     Provides the array of
AVFragmentedMovieTracks contained by the
fragmented movie.
     */
    public static var tracks:
AVAsyncProperty<Root,
[AVFragmentedMovieTrack]> { get }
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVPartialAsyncProperty where
Root : AVComposition {

    /**
     Provides the array of
AVCompositionTracks contained by the
composition.
     */
    public static var tracks:
AVAsyncProperty<Root,
[AVCompositionTrack]> { get }
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVPartialAsyncProperty where
Root : AVAsset {

    /**
     Indicates the duration of the asset.

     If
```

```
`providesPreciseDurationAndTiming` is
false, a best-available estimate of the
duration is returned.
     The degree of precision preferred
for timing-related properties can be set
at initialization time for assets
initialized with URLs.
     See
`AVURLAssetPreferPreciseDurationAndTiming
Key` for AVURLAsset.
    */
    public static var duration:
AVAsyncProperty<Root, CMTime> { get }

    /**
     Indicates the natural rate at which
the asset is to be played; often but not
always 1.0
    */
    public static var preferredRate:
AVAsyncProperty<Root, Float> { get }

    /**
     Indicates the preferred volume at
which the audible media of an asset is to
be played; often but not always 1.0
    */
    public static var preferredVolume:
AVAsyncProperty<Root, Float> { get }

    /**
     Indicates the preferred transform to
apply to the visual content of the asset
```

```
for presentation or processing; the value
is often but not always the identity
transform
    */
    public static var preferredTransform:
AVAsyncProperty<Root, CGAffineTransform>
{ get }

    /**
     Indicates how close to the latest
content in a live stream playback can be
sustained.

     For non-live assets this value is
kCMTimeInvalid.
     */
    public static var
minimumTimeOffsetFromLive:
AVAsyncProperty<Root, CMTime> { get }

    /**
     Indicates that the asset provides
precise timing. See `duration` and
AVURLAssetPreferPreciseDurationAndTimingK
ey.
    */
    public static var
providesPreciseDurationAndTiming:
AVAsyncProperty<Root, Bool> { get }
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
```

```
extension AVPartialAsyncProperty where
Root : AVAsset {

    /**
     Provides the array of AVAssetTracks
contained by the asset.
     */
    public static var tracks:
AVAsyncProperty<Root, [AVAssetTrack]> {
get }

    /**
     All track groups in the asset.
    */
    public static var trackGroups:
AVAsyncProperty<Root,
[AVAssetTrackGroup]> { get }
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVPartialAsyncProperty where
Root : AVAsset {

    /**
     Indicates the creation date of the
asset as an AVMetadataItem. May be nil.

     If a creation date has been stored
by the asset in a form that can be
converted to a Date, the dateValue
property of the AVMetadataItem will
provide an instance of NSDate. Otherwise
```

```
the creation date is available only as a
string value, via stringValue property of
AVMetadataItem.
     */
    public static var creationDate:
AVAsyncProperty<Root, AVMetadataItem?> {
get }

    /**
     Provides access to the lyrics of the
asset suitable for the current locale.
     */
    public static var lyrics:
AVAsyncProperty<Root, String?> { get }

    /**
     Provides access to an array of
AVMetadataItems for each common metadata
key for which a value is available

     Items can be filtered according to
language via
`AVMetadataItem.metadataItems(from:filter
edAndSortedAccordingToPreferredLanguages:
)` and according to identifier via
`AVMetadataItem.metadataItems(from:filter
edByIdentifier:)`.
     */
    public static var commonMetadata:
AVAsyncProperty<Root, [AVMetadataItem]> {
get }

    /**
```

```
     Provides access to an array of
AVMetadataItems for all metadata
identifiers for which a value is
available

     Items can be filtered according to
language via
`AVMetadataItem.metadataItems(from:filter
edAndSortedAccordingToPreferredLanguages:
)` and according to identifier via
`AVMetadataItem.metadataItems(from:filter
edByIdentifier:)`.
     */
    public static var metadata:
AVAsyncProperty<Root, [AVMetadataItem]> {
get }

    /**
     Provides an array containing
metadata format that's available to the
asset (e.g. ID3, iTunes metadata, etc.)
    */
    public static var
availableMetadataFormats:
AVAsyncProperty<Root, [AVMetadataFormat]>
{ get }
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVPartialAsyncProperty where
Root : AVAsset {
    /**
     Indicates the locales for which
chapter metadata items are available
     */
    public static var
availableChapterLocales:
AVAsyncProperty<Root, [Locale]> { get }
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVPartialAsyncProperty where
Root : AVURLAsset {

    /**
         Provides an array of
AVAssetVariants contained in the asset.

         Some variants may not be
playable according to the current device
configuration.
     */
    public static var variants:
AVAsyncProperty<Root, [AVAssetVariant]> {
get }
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVPartialAsyncProperty where
Root : AVAsset {

    /**
```

```
     Provides an array with elements
indicating media characteristic for which
a media selection option is available.
    */
    public static var
availableMediaCharacteristicsWithMediaSel
ectionOptions: AVAsyncProperty<Root,
[AVMediaCharacteristic]> { get }

    /**
     Provides an instance of
AVMediaSelection with default selections
for each of the receiver's media
selection groups.
     */
    public static var
preferredMediaSelection:
AVAsyncProperty<Root, AVMediaSelection> {
get }

    /**
     Provides an array of all
permutations of AVMediaSelection for this
asset.
     */
    public static var allMediaSelections:
AVAsyncProperty<Root, [AVMediaSelection]>
{ get }
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVPartialAsyncProperty where
```

```
Root : AVAsset {

    /**
     Indicates whether or not the asset
has protected content.

     Assets containing protected content
may not be playable without successful
authorization, even if the value of the
`playable` property is true. See the
properties in the AVAssetUsability
category for details on how such an asset
may be used. On macOS, clients can use
the interfaces in
AVPlayerItemProtectedContentAdditions.h
to request authorization to play the
asset.
     */
    @available(macOS 12, iOS 15, tvOS 15,
visionOS 1, *)
    @available(watchOS, unavailable)
    public static var
hasProtectedContent:
AVAsyncProperty<Root, Bool> { get }
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVPartialAsyncProperty where
Root : AVAsset {

    /**
     Indicates whether the asset is
```

```
capable of being extended by fragments.

     For QuickTime movie files and MPEG-4
files, the value of canContainFragments
is true if an 'mvex' box is present in
the 'moov' box. For those types, the
'mvex' box signals the possible presence
of later 'moof' boxes.
     */
    @available(macOS 12, iOS 15, tvOS 15,
visionOS 1, *)
    @available(watchOS, unavailable)
    public static var
canContainFragments:
AVAsyncProperty<Root, Bool> { get }

    /**
     Indicates whether the asset is
extended by at least one fragment.

     For QuickTime movie files and MPEG-4
files, the value of this property is true
if canContainFragments is true and at
least one 'moof' box is present after the
'moov' box.
     */
    @available(macOS 12, iOS 15, tvOS 15,
visionOS 1, *)
    @available(watchOS, unavailable)
    public static var containsFragments:
AVAsyncProperty<Root, Bool> { get }

    /**
```

```
     Indicates the total duration of
fragments that either exist now or may be
appended in the future in order to extend
the duration of the asset.

     For QuickTime movie files and MPEG-4
files, the value of this property is
obtained from the 'mehd' box of the
'mvex' box, if present. If no total
fragment duration hint is available, the
value of this property is kCMTimeInvalid.
     */
    public static var
overallDurationHint:
AVAsyncProperty<Root, CMTime> { get }
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVPartialAsyncProperty where
Root : AVAsset {

    /**
     Indicates whether an AVPlayer can
play the contents of the asset in a
manner that meets user expectations.

     A client can attempt playback when
playable is false, this however may lead
to a substandard playback experience.
     */
    public static var isPlayable:
AVAsyncProperty<Root, Bool> { get }
```

```
    /**
     Indicates whether an
AVAssetExportSession can be used with the
receiver for export.
     */
    @available(macOS 12, iOS 15, tvOS 15,
visionOS 1, *)
    @available(watchOS, unavailable)
    public static var isExportable:
AVAsyncProperty<Root, Bool> { get }

    /**
     Indicates whether an AVAssetReader
can be used with the receiver for
extracting media data.
     */
    @available(macOS 12, iOS 15, tvOS 15,
visionOS 1, *)
    @available(watchOS, unavailable)
    public static var isReadable:
AVAsyncProperty<Root, Bool> { get }

    /**
     Indicates whether the receiver can
be used to build an AVMutableComposition.
     */
    public static var isComposable:
AVAsyncProperty<Root, Bool> { get }

    /**
         Indicates whether the asset is
compatible with AirPlay Video.
```

```
         true if an AVPlayerItem
initialized with the receiver can be
played by an external device via AirPlay
Video.
         */
    @available(macOS 12, iOS 15, tvOS 15,
visionOS 1, *)
    @available(watchOS, unavailable)
    public static var
isCompatibleWithAirPlayVideo:
AVAsyncProperty<Root, Bool> { get }
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVPartialAsyncProperty where
Root : AVMutableComposition {

    /**
     Provides the array of
AVMutableCompositionTracks contained by
the mutable composition.
     */
    public static var tracks:
AVAsyncProperty<Root,
[AVMutableCompositionTrack]> { get }
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVPartialAsyncProperty where
Root : AVAssetTrack {
    /**
     Provides an array of
CMFormatDescriptions each of which
indicates the format of media samples
referenced by the track.

     A track that presents uniform media,
e.g. encoded according to the same
encoding settings, will provide an array
with a count of 1
     */
    public static var formatDescriptions:
AVAsyncProperty<Root,
[CMFormatDescription]> { get }

    /**
     Indicates whether the receiver is
playable in the current environment.

     If `true`, an AVPlayerItemTrack of
an AVPlayerItem initialized with the
receiver's asset can be enabled for
playback.
     */
    public static var isPlayable:
AVAsyncProperty<Root, Bool> { get }

    /**
     Indicates whether the receiver is
decodable in the current environment.

     If `true`, the track can be decoded
```

```
even though decoding may be too slow for
real time playback.
     */
    public static var isDecodable:
AVAsyncProperty<Root, Bool> { get }

    /**
     Indicates whether the track is
enabled according to state stored in its
container or construct.

     Note that its presentation state can
be changed from this default via
AVPlayerItemTrack
     */
    public static var isEnabled:
AVAsyncProperty<Root, Bool> { get }

    /**
     Indicates whether the track
references sample data only within its
storage container.
     */
    public static var isSelfContained:
AVAsyncProperty<Root, Bool> { get }

    /**
     Indicates the total number of bytes
of sample data required by the track.
     */
    public static var
totalSampleDataLength:
AVAsyncProperty<Root, Int64> { get }
```

```
    /**
     Indicates all available media
characteristics for the track.

     Media characteristics values are
`.visual`, `.audible`, `.legible` etc.
     */
    public static var
mediaCharacteristics:
AVAsyncProperty<Root,
[AVMediaCharacteristic]> { get }
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVPartialAsyncProperty where
Root : AVAssetTrack {

    /**
     Indicates the timeRange of the track
within the overall timeline of the asset.

     A track with `timeRange.start
> .zero` will initially present an empty
interval.
     */
    public static var timeRange:
AVAsyncProperty<Root, CMTimeRange> {
get }

    /**
     Indicates a timescale in which time
```

```
values for the track can be operated upon
without extraneous numerical conversion.
     */
    public static var naturalTimeScale:
AVAsyncProperty<Root, CMTimeScale> {
get }

    /**
     Indicates the estimated data rate of
the media data referenced by the track,
in units of bits per second
     */
    public static var estimatedDataRate:
AVAsyncProperty<Root, Float> { get }
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVPartialAsyncProperty where
Root : AVAssetTrack {

    /**
     Indicates the language associated
with the track, as an ISO 639-2/T
language code

     May be nil if no language is
indicated
     */
    public static var languageCode:
AVAsyncProperty<Root, String?> { get }

    /**
```

```
     Indicates the language tag
associated with the track, as an IETF BCP
47 (RFC 4646) language identifier

     May be nil if no language tag is
indicated
     */
    public static var
extendedLanguageTag:
AVAsyncProperty<Root, String?> { get }
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVPartialAsyncProperty where
Root : AVAssetTrack {

    /**
     Indicates the natural dimensions of
the media data referenced by the track as
a CGSize.
     */
    public static var naturalSize:
AVAsyncProperty<Root, CGSize> { get }

    /**
     Indicates the transform specified in
the track's storage container as the
preferred transformation of the visual
media data for display purposes

     Value returned is often but not
always `.identity`
```

```
     */
    public static var preferredTransform:
AVAsyncProperty<Root, CGAffineTransform>
{ get }
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVPartialAsyncProperty where
Root : AVAssetTrack {

    /**
     Indicates the volume specified in
the track's storage container as the
preferred volume of the audible media
data.
     */
    public static var preferredVolume:
AVAsyncProperty<Root, Float> { get }

    /**
     Indicates whether this audio track
has dependencies (e.g.
kAudioFormatMPEGD_USAC) .
     */
    public static var
hasAudioSampleDependencies:
AVAsyncProperty<Root, Bool> { get }
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVPartialAsyncProperty where
```

```
Root : AVAssetTrack {

    /**
     Indicates the frame rate associated
with this track.

     For tracks that carry a full frame
per media sample, indicates the frame
rate of the track in units of frames per
second. For field-based video tracks that
carry one field per media sample, the
value of this property is the field rate,
not the frame rate.
     */
    public static var nominalFrameRate:
AVAsyncProperty<Root, Float> { get }

    /**
     Indicates the minimum duration of
the track's frames

     The value will be kCMTimeInvalid if
the minimum frame duration is not known
or cannot be calculated
     */
    public static var minFrameDuration:
AVAsyncProperty<Root, CMTime> { get }

    /**
     Indicates whether samples in the
track may have different values for their
presentation and decode timestamps.
     */
    public static var
requiresFrameReordering:
AVAsyncProperty<Root, Bool> { get }
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVPartialAsyncProperty where
Root : AVAssetTrack {

    /**
     Provides an array of
AVAssetTrackSegments with time mappings
from the timeline of the track's media
samples to the timeline of the track.

     Empty edits, i.e. timeRanges for
which no media data is available to be
presented, have a value of
AVAssetTrackSegment.empty equal to true.
     */
    public static var segments:
AVAsyncProperty<Root,
[AVAssetTrackSegment]> { get }
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVPartialAsyncProperty where
Root : AVAssetTrack {

    /**
     Provides access to an array of
```

```
AVMetadataItems for each common metadata
key for which a value is available
     */
    public static var commonMetadata:
AVAsyncProperty<Root, [AVMetadataItem]> {
get }

    /**
     Provides access to an array of
AVMetadataItems for all metadata
identifiers for which a value is
available

     Items can be filtered according to
language via
     `AVMetadataItem.
metadataItems(from:filteredAndSortedAccor
dingToPreferredLanguages:)` and according
to identifier via
`AVMetadataItem.metadataItems(from:filter
edByIdentifier:)`.
    */
    public static var metadata:
AVAsyncProperty<Root, [AVMetadataItem]> {
get }

    /**
     Provides an array in which each
element represents a format of metadata
that's available for the track (e.g.
QuickTime userdata, etc.)

     Metadata formats are defined in
```

```
AVMetadataFormat.
     */
    public static var
availableMetadataFormats:
AVAsyncProperty<Root, [AVMetadataFormat]>
{ get }
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVPartialAsyncProperty where
Root : AVAssetTrack {

    /**
     Provides an array in which each
element represents a type of track
association that the receiver has with
one or more of the other tracks of the
asset (e.g. `.chapterList`, `.timecode`,
etc).
     */
    public static var
availableTrackAssociationTypes:
AVAsyncProperty<Root,
[AVAssetTrack.AssociationType]> { get }
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVPartialAsyncProperty where
Root : AVAssetTrack {

    /**
```

```
     Indicates whether the receiver can
provide instances of AVSampleCursor for
traversing its media samples and
discovering information about them.
     */
    @available(macOS 12, iOS 16, tvOS 16,
watchOS 9, visionOS 1, *)
    public static var
canProvideSampleCursors:
AVAsyncProperty<Root, Bool> { get }
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVPartialAsyncProperty where
Root : AVFragmentedAsset {

    /**
     Provides the array of
AVFragmentedAssetTracks contained by the
fragmented asset.
     */
    public static var tracks:
AVAsyncProperty<Root,
[AVFragmentedAssetTrack]> { get }
}

@available(macOS 12, iOS 15, watchOS 8,
visionOS 1, *)
@available(tvOS, unavailable)
extension AVPartialAsyncProperty where
Root : AVMovie {
    /**
     Provides the array of AVMovieTracks
contained by the movie.
     */
    public static var tracks:
AVAsyncProperty<Root, [AVMovieTrack]> {
get }
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVPartialAsyncProperty where
Root : AVURLAsset {

    /**
     Provides the array of AVAssetTracks
contained by the url asset.
     */
    public static var tracks:
AVAsyncProperty<Root, [AVAssetTrack]> {
get }
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVPlayerInterstitialEvent {

    /**
         AVPlayerInterstitialEvent
initializer by time

         - Parameters:
            - primaryItem: The
```

AVPlayerItem playing the primary content, against which the interstitial event will be scheduled.

- identifier: A persistent identifier for the event.
- time: The time within the duration of the primary item at which playback of the primary content should be temporarily suspended and the interstitial items played.
- templateItems: An array of AVPlayerItems with configurations that will be reproduced for the playback of interstitial content.
- restrictions: Indicates restrictions on the use of end user playback controls that are imposed by the event.
- resumptionOffset: Specifies the offset in time at which playback of the primary item should resume after interstitial playback has finished. Definite numeric values are supported. The value .indefinite can also be used, in order to specify that the effective resumption time offset should accord with the wallclock time elapsed during interstitial playback.
- playoutLimit: Specifies the offset from the beginning of the interstitial at which interstitial playback should end, if the interstitial asset(s) are longer. Pass a positive

```
numeric value, or .invalid to indicate no
playout limit.
            - userDefinedAttributes:
Storage for attributes defined by the
client or the content vendor. Attribute
names should begin with X- for uniformity
with server insertion.
         */
    @available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
    public convenience init(primaryItem:
AVPlayerItem, identifier: String?, time:
CMTime, templateItems: [AVPlayerItem],
restrictions:
AVPlayerInterstitialEvent.Restrictions =
[], resumptionOffset: CMTime
= .indefinite, playoutLimit: CMTime
= .invalid, userDefinedAttributes:
[String : Any] = [:])

    /**
         AVPlayerInterstitialEvent
initializer by date

         - Parameters:
            - primaryItem: The
AVPlayerItem playing the primary content,
against which the interstitial event will
be scheduled. The primaryItem must have
an AVAsset that provides an intrinsic
mapping from its timeline to real-time
dates.
            - identifier: A persistent
```

identifier for the event.
            - date: The date within the date range of the primary item at which playback of the primary content should be temporarily suspended and the interstitial items played.
            - templateItems: An array of AVPlayerItems with configurations that will be reproduced for the playback of interstitial content.
            - restrictions: Indicates restrictions on the use of end user playback controls that are imposed by the event.
            - resumptionOffset: Specifies the offset in time at which playback of the primary item should resume after interstitial playback has finished. Definite numeric values are supported. The value .indefinite can also be used, in order to specify that the effective resumption time offset should accord with the wallclock time elapsed during interstitial playback.
            - playoutLimit: Specifies the offset from the beginning of the interstitial at which interstitial playback should end, if the interstitial asset(s) are longer. Pass a positive numeric value, or .invalid to indicate no playout limit.
            - userDefinedAttributes: Storage for attributes defined by the

```
client or the content vendor. Attribute
names should begin with X- for uniformity
with server insertion.
         */
    @available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
    public convenience init(primaryItem:
AVPlayerItem, identifier: String?, date:
Date, templateItems: [AVPlayerItem],
restrictions:
AVPlayerInterstitialEvent.Restrictions =
[], resumptionOffset: CMTime
= .indefinite, playoutLimit: CMTime
= .invalid, userDefinedAttributes:
[String : Any] = [:])
}

extension NSNotification.Name {

    @available(macOS, introduced: 13.3,
deprecated: 14.0, renamed:
"AVPlayerInterstitialEventMonitor.assetLi
stResponseStatusDidChangeNotification")
    @available(iOS, introduced: 16.4,
deprecated: 17.0, renamed:
"AVPlayerInterstitialEventMonitor.assetLi
stResponseStatusDidChangeNotification")
    @available(tvOS, introduced: 16.4,
deprecated: 17.0, renamed:
"AVPlayerInterstitialEventMonitor.assetLi
stResponseStatusDidChangeNotification")
    @available(watchOS, introduced: 9.4,
deprecated: 10.0, renamed:
```

```
"AVPlayerInterstitialEventMonitor.assetLi
stResponseStatusDidChangeNotification")
    @available(visionOS, introduced: 1,
deprecated: 1, renamed:
"AVPlayerInterstitialEventMonitor.assetLi
stResponseStatusDidChangeNotification")
    public static var
AVPlayerInterstitialEventMonitorAssetList
ResponseStatusDidChange:
NSNotification.Name { get }
}

@available(watchOS 6.0, *)
extension AVError {

    @available(swift 4.2)
    @available(macCatalyst 14.0, tvOS
17.0, *)
    @available(visionOS, unavailable)
    @available(watchOS, unavailable)
    public var device: AVCaptureDevice? {
get }

    /// The time.
    @available(watchOS 6.0, *)
    public var time: CMTime? { get }

    /// The file size.
    @available(watchOS 6.0, *)
    public var fileSize: Int64? { get }

    /// The process ID number.
    @available(watchOS 6.0, *)
    public var processID: Int? { get }

    /// Whether the recording
successfully finished.
    @available(watchOS 6.0, *)
    public var
recordingSuccessfullyFinished: Bool? {
get }

    /// The media type.
    @available(swift 4.2)
    @available(watchOS 6.0, *)
    public var mediaType: AVMediaType? {
get }

    /// The media subtypes.
    @available(watchOS 6.0, *)
    public var mediaSubtypes: [Int]? {
get }

    /// The presentation time stamp.
    @available(swift 4.2)
    @available(macOS 10.10, iOS 8.0, tvOS
9.0, watchOS 6.0, visionOS 1.0, *)
    public var presentationTimeStamp:
CMTime? { get }

    /// The persistent track ID.
    @available(swift 4.2)
    @available(macOS 10.10, iOS 8.0, tvOS
9.0, watchOS 6.0, visionOS 1.0, *)
    public var persistentTrackID:
CMPersistentTrackID? { get }
```

```
    /// The file type.
    @available(swift 4.2)
    @available(macOS 10.10, iOS 8.0, tvOS
9.0, watchOS 6.0, visionOS 1.0, *)
    public var fileType: AVFileType? {
get }
}

@available(macOS 15.0, iOS 18.0, tvOS
18.0, *)
@available(visionOS, unavailable)
@available(watchOS, unavailable)
extension AVCaptureIndexPicker {

    @nonobjc public func setActionQueue(_
actionQueue: DispatchQueue, action:
@escaping (Int) -> ())
}

@available(macOS 10.13, iOS 11.0,
macCatalyst 14.0, tvOS 11.0, visionOS
1.0, *)
@available(watchOS, unavailable)
extension AVDepthData {

    @available(macOS 10.13, iOS 11.0,
macCatalyst 14.0, tvOS 11.0, visionOS
1.0, *)
    @available(watchOS, unavailable)
    @nonobjc public var
availableDepthDataTypes: [OSType] { get }
}
```

```
extension NSNotification.Name {

    @available(macOS, introduced: 10.7,
deprecated: 15.0, renamed:
"AVCaptureSession.runtimeErrorNotificatio
n")
    @available(iOS, introduced: 4.0,
deprecated: 18.0, renamed:
"AVCaptureSession.runtimeErrorNotificatio
n")
    @available(macCatalyst, introduced:
14.0, deprecated: 18.0, renamed:
"AVCaptureSession.runtimeErrorNotificatio
n")
    @available(tvOS, introduced: 17.0,
deprecated: 18.0, renamed:
"AVCaptureSession.runtimeErrorNotificatio
n")
    @available(visionOS, introduced: 1.0,
deprecated: 2.0, renamed:
"AVCaptureSession.runtimeErrorNotificatio
n")
    @available(watchOS, unavailable)
    public static var
AVCaptureSessionRuntimeError:
NSNotification.Name { get }

    @available(macOS, introduced: 10.7,
deprecated: 15.0, renamed:
"AVCaptureSession.didStartRunningNotifica
tion")
    @available(iOS, introduced: 4.0,
deprecated: 18.0, renamed:
"AVCaptureSession.didStartRunningNotifica
tion")
    @available(macCatalyst, introduced:
14.0, deprecated: 18.0, renamed:
"AVCaptureSession.didStartRunningNotifica
tion")
    @available(tvOS, introduced: 17.0,
deprecated: 18.0, renamed:
"AVCaptureSession.didStartRunningNotifica
tion")
    @available(visionOS, introduced: 1.0,
deprecated: 2.0, renamed:
"AVCaptureSession.didStartRunningNotifica
tion")
    @available(watchOS, unavailable)
    public static var
AVCaptureSessionDidStartRunning:
NSNotification.Name { get }

    @available(macOS, introduced: 10.7,
deprecated: 15.0, renamed:
"AVCaptureSession.didStopRunningNotificat
ion")
    @available(iOS, introduced: 4.0,
deprecated: 18.0, renamed:
"AVCaptureSession.didStopRunningNotificat
ion")
    @available(macCatalyst, introduced:
14.0, deprecated: 18.0, renamed:
"AVCaptureSession.didStopRunningNotificat
ion")
    @available(tvOS, introduced: 17.0,
deprecated: 18.0, renamed:
"AVCaptureSession.didStopRunningNotificat
ion")
    @available(visionOS, introduced: 1.0,
deprecated: 2.0, renamed:
"AVCaptureSession.didStopRunningNotificat
ion")
    @available(watchOS, unavailable)
    public static var
AVCaptureSessionDidStopRunning:
NSNotification.Name { get }

    @available(macOS, introduced: 10.14,
deprecated: 15.0, renamed:
"AVCaptureSession.wasInterruptedNotificat
ion")
    @available(iOS, introduced: 4.0,
deprecated: 18.0, renamed:
"AVCaptureSession.wasInterruptedNotificat
ion")
    @available(macCatalyst, introduced:
14.0, deprecated: 18.0, renamed:
"AVCaptureSession.wasInterruptedNotificat
ion")
    @available(tvOS, introduced: 17.0,
deprecated: 18.0, renamed:
"AVCaptureSession.wasInterruptedNotificat
ion")
    @available(visionOS, introduced: 1.0,
deprecated: 2.0, renamed:
"AVCaptureSession.wasInterruptedNotificat
ion")
    @available(watchOS, unavailable)
    public static var
AVCaptureSessionWasInterrupted:
NSNotification.Name { get }

    @available(macOS, introduced: 10.14,
deprecated: 15.0, renamed:
"AVCaptureSession.interruptionEndedNotifi
cation")
    @available(iOS, introduced: 4.0,
deprecated: 18.0, renamed:
"AVCaptureSession.interruptionEndedNotifi
cation")
    @available(macCatalyst, introduced:
14.0, deprecated: 18.0, renamed:
"AVCaptureSession.interruptionEndedNotifi
cation")
    @available(tvOS, introduced: 17.0,
deprecated: 18.0, renamed:
"AVCaptureSession.interruptionEndedNotifi
cation")
    @available(visionOS, introduced: 1.0,
deprecated: 2.0, renamed:
"AVCaptureSession.interruptionEndedNotifi
cation")
    @available(watchOS, unavailable)
    public static var
AVCaptureSessionInterruptionEnded:
NSNotification.Name { get }
}

extension AVAsynchronousKeyValueLoading {

    /**
```

```
         Get current status of a property

         - Parameters:
            - property: Property to be
checked.
         - Returns: Current status of the
property

         If the property is already
loaded, Status contains the value of the
property.
         If the property failed to load,
Status contains the error identifier.
     If property loading was cancelled,
the status will be .failed and the error
will be an NSError with domain
AVFoundationErrorDomain and code
AVError.operationCancelled.rawValue.
         */
    @available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
    public func status<T>(of property:
AVAsyncProperty<Self, T>) ->
AVAsyncProperty<Self, T>.Status

    /**
         Loads a property and returns the
current value.

         - Parameters:
            - property: Property to be
loaded
         - Returns: Value of the property
```

```
or throws an error if it could not be
loaded

         Note that this method
asynchronously loads the property before
returning the value.
     If property loading was cancelled,
this method throws an NSError with domain
AVFoundationErrorDomain and code
AVError.operationCancelled.rawValue.
         */
    @available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
    public func load<T>(_ property:
AVAsyncProperty<Self, T>) async throws ->
T
}

extension AVAsynchronousKeyValueLoading {

    /**
         Loads properties and returns the
values.

         - Parameters:
            - propertyA: First property
to load.
            - propertyB: Second property
to load.
         - Returns: Values of the
properties or throws an error if any of
them failed to load.
```

```
        Note that this method
asynchronously loads the properties
before returning the values.
     If property loading was cancelled,
this method throws an NSError with domain
AVFoundationErrorDomain and code
AVError.operationCancelled.rawValue.
        */
    @available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
    public func load<A, B>(_ propertyA:
AVAsyncProperty<Self, A>, _ propertyB:
AVAsyncProperty<Self, B>) async throws ->
(A, B)

    /**
        Loads properties and returns the
values.

        - Parameters:
           - propertyA: First property
to load.
           - propertyB: Second property
to load.
           - propertyC: Third property
to load.
        - Returns: Values of the
properties or throws an error if any of
them failed to load.

        Note that this method
asynchronously loads the properties
before returning the values.
```

```
     If property loading was cancelled,
this method throws an NSError with domain
AVFoundationErrorDomain and code
AVError.operationCancelled.rawValue.
        */
    @available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
    public func load<A, B, C>(_
propertyA: AVAsyncProperty<Self, A>, _
propertyB: AVAsyncProperty<Self, B>, _
propertyC: AVAsyncProperty<Self, C>)
async throws -> (A, B, C)

    /**
        Loads properties and returns the
values.

        - Parameters:
           - propertyA: First property
to load.
           - propertyB: Second property
to load.
           - propertyC: Third property
to load.
           - propertyD: Fourth property
to load.
        - Returns: Values of the
properties or throws an error if any of
them failed to load.

        Note that this method
asynchronously loads the properties
before returning the values.
```

```
     If property loading was cancelled,
this method throws an NSError with domain
AVFoundationErrorDomain and code
AVError.operationCancelled.rawValue.
         */
    @available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
    public func load<A, B, C, D>(_
propertyA: AVAsyncProperty<Self, A>, _
propertyB: AVAsyncProperty<Self, B>, _
propertyC: AVAsyncProperty<Self, C>, _
propertyD: AVAsyncProperty<Self, D>)
async throws -> (A, B, C, D)

    /**
         Loads properties and returns the
values.

         - Parameters:
            - propertyA: First property
to load.
            - propertyB: Second property
to load.
            - propertyC: Third property
to load.
            - propertyD: Fourth property
to load.
            - propertyE: Fifth property
to load.
         - Returns: Values of the
properties or throws an error if any of
them failed to load.
```

```
         Note that this method
asynchronously loads the properties
before returning the values.
     If property loading was cancelled,
this method throws an NSError with domain
AVFoundationErrorDomain and code
AVError.operationCancelled.rawValue.
        */
    @available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
    public func load<A, B, C, D, E>(_
propertyA: AVAsyncProperty<Self, A>, _
propertyB: AVAsyncProperty<Self, B>, _
propertyC: AVAsyncProperty<Self, C>, _
propertyD: AVAsyncProperty<Self, D>, _
propertyE: AVAsyncProperty<Self, E>)
async throws -> (A, B, C, D, E)

    /**
         Loads properties and returns the
values.

         - Parameters:
            - propertyA: First property
to load.
            - propertyB: Second property
to load.
            - propertyC: Third property
to load.
            - propertyD: Fourth property
to load.
            - propertyE: Fifth property
to load.
```

```
            - propertyF: Sixth property
to load.
         - Returns: Values of the
properties or throws an error if any of
them failed to load.

         Note that this method
asynchronously loads the properties
before returning the values.
     If property loading was cancelled,
this method throws an NSError with domain
AVFoundationErrorDomain and code
AVError.operationCancelled.rawValue.
         */
    @available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
    public func load<A, B, C, D, E, F>(_
propertyA: AVAsyncProperty<Self, A>, _
propertyB: AVAsyncProperty<Self, B>, _
propertyC: AVAsyncProperty<Self, C>, _
propertyD: AVAsyncProperty<Self, D>, _
propertyE: AVAsyncProperty<Self, E>, _
propertyF: AVAsyncProperty<Self, F>)
async throws -> (A, B, C, D, E, F)

    /**
         Loads properties and returns the
values.

         - Parameters:
            - propertyA: First property
to load.
            - propertyB: Second property
```

```
to load.
            - propertyC: Third property
to load.
            - propertyD: Fourth property
to load.
            - propertyE: Fifth property
to load.
            - propertyF: Sixth property
to load.
            - propertyG: Seventh property
to load.
         - Returns: Values of the
properties or throws an error if any of
them failed to load.

         Note that this method
asynchronously loads the properties
before returning the values.
     If property loading was cancelled,
this method throws an NSError with domain
AVFoundationErrorDomain and code
AVError.operationCancelled.rawValue.
        */
    @available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
    public func load<A, B, C, D, E, F,
G>(_ propertyA: AVAsyncProperty<Self, A>,
_ propertyB: AVAsyncProperty<Self, B>, _
propertyC: AVAsyncProperty<Self, C>, _
propertyD: AVAsyncProperty<Self, D>, _
propertyE: AVAsyncProperty<Self, E>, _
propertyF: AVAsyncProperty<Self, F>, _
propertyG: AVAsyncProperty<Self, G>)
```

```
async throws -> (A, B, C, D, E, F, G)

    /**
         Loads properties and returns the
values.

         - Parameters:
            - propertyA: First property
to load.
            - propertyB: Second property
to load.
            - propertyC: Third property
to load.
            - propertyD: Fourth property
to load.
            - propertyE: Fifth property
to load.
            - propertyF: Sixth property
to load.
            - propertyG: Seventh property
to load.
            - propertyH: Eighth property
to load.
         - Returns: Values of the
properties or throws an error if any of
them failed to load.

         Note that this method
asynchronously loads the properties
before returning the values.
     If property loading was cancelled,
this method throws an NSError with domain
AVFoundationErrorDomain and code
```

```
AVError.operationCancelled.rawValue.
         */
    @available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
    public func load<A, B, C, D, E, F, G,
H>(_ propertyA: AVAsyncProperty<Self, A>,
_ propertyB: AVAsyncProperty<Self, B>, _
propertyC: AVAsyncProperty<Self, C>, _
propertyD: AVAsyncProperty<Self, D>, _
propertyE: AVAsyncProperty<Self, E>, _
propertyF: AVAsyncProperty<Self, F>, _
propertyG: AVAsyncProperty<Self, G>, _
propertyH: AVAsyncProperty<Self, H>)
async throws -> (A, B, C, D, E, F, G, H)
}

@available(macOS 11.0, iOS 7.0,
macCatalyst 14.0, tvOS 17.0, *)
@available(watchOS, unavailable)
@available(visionOS, unavailable)
extension
AVMetadataMachineReadableCodeObject {

    @nonobjc public var corners:
[CGPoint] { get }
}

extension AVAsset {

    /**
         Tests, in order of preference,
for a match between language identifiers
in the specified array of preferred
```

```
languages and the available chapter
locales,
         and loads the array of chapters
corresponding to the first match that's
found.

         - Parameters:
            - locale: Locale of the
metadata items carrying chapter titles to
be returned (supports the IETF BCP 47
specification).
            - commonKeys: Array of common
keys of AVMetadataItem to be included; if
no common keys are required, send an
empty list. AVMetadataCommonKeyArtwork is
the only supported key for now.
         - Returns: An array of
AVTimedMetadataGroup objects.

         Each object in the array always
contains an AVMetadataItem representing
the chapter title; the timeRange property
of the AVTimedMetadataGroup object is
equal to the time range of the chapter
title item. An AVMetadataItem with the
specified common key will be added to an
existing AVTimedMetadataGroup object if
the time range (timestamp and duration)
of the metadata item and the metadata
group overlaps. The locale of items not
carrying chapter titles need not match
the specified locale parameter. Further
filtering of the metadata items in
```

```
AVTimedMetadataGroups according to
language can be accomplished using
`AVMetadataItem.metadataItems(from:filter
edAndSortedAccordingToPreferredLanguages:
)`. Filtering of the metadata items
according to locale can be accomplished
using
`AVMetadataItem.metadataItems(from:withLo
cale:)`.
         */
    @available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
    public func
loadChapterMetadataGroups(withTitleLocale
locale: Locale,
containingItemsWithCommonKeys commonKeys:
[AVMetadataKey] = []) async throws ->
[AVTimedMetadataGroup]
}

@available(macOS 15, iOS 18, tvOS 18,
visionOS 2, *)
@available(watchOS, unavailable)
extension AVAssetResourceLoader :
@unchecked Sendable {
}

@available(macOS 15, iOS 18, tvOS 18,
visionOS 2, *)
@available(watchOS, unavailable)
extension AVAssetResourceLoadingRequest :
@unchecked Sendable {
}
```

```
@available(macOS 15, iOS 18, tvOS 18,
visionOS 2, *)
@available(watchOS, unavailable)
extension AVAssetResourceLoadingRequestor
: @unchecked Sendable {
}

@available(macOS 15, iOS 18, tvOS 18,
visionOS 2, *)
@available(watchOS, unavailable)
extension AVAssetResourceRenewalRequest :
@unchecked Sendable {
}

@available(macOS 15, iOS 18, tvOS 18,
visionOS 2, *)
@available(watchOS, unavailable)
extension
AVAssetResourceLoadingContentInformationR
equest : @unchecked Sendable {
}

@available(macOS 15, iOS 18, tvOS 18,
visionOS 2, *)
@available(watchOS, unavailable)
extension
AVAssetResourceLoadingDataRequest :
@unchecked Sendable {
}

@available(macOS 12.0, iOS 18.0,
macCatalyst 15.0, *)
```

```
@available(tvOS, unavailable)
@available(watchOS, unavailable)
@available(visionOS, unavailable)
extension AVCaption {

    @nonobjc public func textColor(at
index: String.Index) -> (CGColor?,
Range<String.Index>)

    @nonobjc public func
backgroundColor(at index: String.Index)
-> (CGColor?, Range<String.Index>)

    @nonobjc public func fontWeight(at
index: String.Index) ->
(AVCaption.FontWeight,
Range<String.Index>)

    @nonobjc public func fontStyle(at
index: String.Index) ->
(AVCaption.FontStyle,
Range<String.Index>)

    @nonobjc public func decoration(at
index: String.Index) ->
(AVCaption.Decoration,
Range<String.Index>)

    @nonobjc public func textCombine(at
index: String.Index) ->
(AVCaption.TextCombine,
Range<String.Index>)
```

```
    @nonobjc public func ruby(at index:
String.Index) -> (AVCaption.Ruby?,
Range<String.Index>)
}

@available(macOS 12.0, iOS 18.0,
macCatalyst 15.0, *)
@available(tvOS, unavailable)
@available(watchOS, unavailable)
@available(visionOS, unavailable)
extension AVMutableCaption {

    @nonobjc public func setTextColor(_
textColor: CGColor, in range: NSRange)

    @nonobjc public func
setBackgroundColor(_ backgroundColor:
CGColor, in range: NSRange)

    @nonobjc public func setFontWeight(_
fontWeight: AVCaption.FontWeight, in
range: NSRange)

    @nonobjc public func setFontStyle(_
fontStyle: AVCaption.FontStyle, in range:
NSRange)

    @nonobjc public func setDecoration(_
decoration: AVCaption.Decoration, in
range: NSRange)

    @nonobjc public func setTextCombine(_
textCombine: AVCaption.TextCombine, in
range: NSRange)

    @nonobjc public func setRuby(_
rubyText: AVCaption.Ruby, in range:
NSRange)

    @nonobjc public func
removeTextColor(in range: NSRange)

    @nonobjc public func
removeBackgroundColor(in range: NSRange)

    @nonobjc public func
removeFontWeight(in range: NSRange)

    @nonobjc public func
removeFontStyle(in range: NSRange)

    @nonobjc public func
removeDecoration(in range: NSRange)

    @nonobjc public func
removeTextCombine(in range: NSRange)

    @nonobjc public func removeRuby(in
range: NSRange)
}

@available(macOS 15, iOS 18, tvOS 18,
watchOS 11, visionOS 2, *)
extension AVMetricEventStreamPublisher {

    public func
```

```
metrics<MetricEvent>(forType metricType:
MetricEvent.Type) ->
AVMetrics<MetricEvent> where
MetricEvent : AVMetricEvent

    public func allMetrics() ->
AVMetrics<AVMetricEvent>
}

@available(macOS 15, iOS 18, tvOS 18,
watchOS 11, visionOS 2, *)
extension
AVMetricPlayerItemLikelyToKeepUpEvent {

    /**
    - Parameter loadedTimeRanges:
Provides a collection of time ranges for
which the player has the media data
readily available. The ranges provided
might be discontinuous.
    - Returns: An array containing
CMTimeRanges.
     */
    @nonobjc public var loadedTimeRanges:
[CMTimeRange] { get }
}

@available(macOS 15, iOS 18, tvOS 18,
watchOS 11, visionOS 2, *)
extension
AVMetricPlayerItemVariantSwitchEvent {

    /**
    - Parameter loadedTimeRanges:
Provides a collection of time ranges for
which the player has the media data
readily available. The ranges provided
might be discontinuous.
    - Returns: An array containing
CMTimeRanges.
     */
    @nonobjc public var loadedTimeRanges:
[CMTimeRange] { get }
}

@available(macOS 15, iOS 18, tvOS 18,
watchOS 11, visionOS 2, *)
extension
AVMetricPlayerItemVariantSwitchStartEvent
{

    /**
    - Parameter loadedTimeRanges:
Provides a collection of time ranges for
which the player has the media data
readily available. The ranges provided
might be discontinuous.
    - Returns: An array containing
CMTimeRanges.
     */
    @nonobjc public var loadedTimeRanges:
[CMTimeRange] { get }
}

@available(macOS 14, iOS 17, visionOS 1,
*)
```

```
@available(tvOS, unavailable)
@available(watchOS, unavailable)
extension
AVAssetWriterInputTaggedPixelBufferGroupA
daptor {

    public func appendTaggedBuffers(_
taggedBuffers: [CMTaggedBuffer],
withPresentationTime: CMTime) -> Bool
}

@available(macOS 15.0, iOS 18.0, tvOS
18.0, *)
@available(visionOS, unavailable)
@available(watchOS, unavailable)
extension AVCaptureSlider {

    @nonobjc public var prominentValues:
[Float]

    @nonobjc public convenience init(_
localizedTitle: String, symbolName:
String, in range: ClosedRange<Float>)

    @nonobjc public convenience init(_
localizedTitle: String, symbolName:
String, in range: ClosedRange<Float>,
step: Float)

    @nonobjc public convenience init(_
localizedTitle: String, symbolName:
String, values: [Float])
```

```
    @nonobjc public func setActionQueue(_
actionQueue: DispatchQueue, action:
@escaping (Float) -> ())
}

@available(macOS 12.0, iOS 15.0, tvOS
15.0, visionOS 1, *)
@available(watchOS, unavailable)
extension
AVAsynchronousVideoCompositionRequest {

    @nonobjc public var
sourceSampleDataTrackIDs:
[CMPersistentTrackID] { get }
}

@available(macOS 12.0, iOS 15.0, tvOS
15.0, visionOS 1, *)
@available(watchOS, unavailable)
extension AVVideoComposition {

    @objc(_sourceSampleDataTrackIDs)
dynamic public var
sourceSampleDataTrackIDs:
[CMPersistentTrackID] { get }
}

@available(macOS 12.0, iOS 15.0, tvOS
15.0, visionOS 1, *)
@available(watchOS, unavailable)
extension AVMutableVideoComposition {

    @objc(_sourceSampleDataTrackIDs)
```

```
override dynamic public var
sourceSampleDataTrackIDs:
[CMPersistentTrackID]
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVAssetVariant {

    /**
    - Parameter peakBitRate: If it is not
declared, the value will be nil.
     */
    @nonobjc public var peakBitRate:
Double? { get }

    /**
    - Parameter averageBitRate: If it is
not declared, the value will be nil.
     */
    @nonobjc public var averageBitRate:
Double? { get }
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVAssetVariant.VideoAttributes
{

    /**
    - Parameter nominalFrameRate: If it
is not declared, the value will be nil.
     */
    @nonobjc public var nominalFrameRate:
Double? { get }

    @nonobjc public var codecTypes:
[CMVideoCodecType] { get }
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension AVAssetVariant.AudioAttributes
{

    @nonobjc public var formatIDs:
[AudioFormatID] { get }
}

@available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
extension
AVAssetVariant.AudioAttributes.RenditionS
pecificAttributes {

    /**
     - Parameter channelCount: If it is
not declared, the value will be nil.
     */
    @nonobjc public var channelCount:
Int? { get }
}

@available(macOS 13.0, iOS 16.0, tvOS
16.0, watchOS 9.0, visionOS 1.0, *)
extension
```

```
AVAssetVariant.VideoAttributes :
@unchecked Sendable {
}

@available(macOS 15, iOS 18, tvOS 18,
watchOS 11, visionOS 2.0, *)
extension AVPlayerItemSegment {

    @available(macOS 15, iOS 18, tvOS 18,
watchOS 11, visionOS 2.0, *)
    @nonobjc public var loadedTimeRanges:
[CMTimeRange] { get }
}

@available(macOS 15, iOS 18, tvOS 18,
watchOS 11, visionOS 2.0, *)
extension
AVPlayerItemIntegratedTimelineSnapshot {

    @available(macOS 15, iOS 18, tvOS 18,
watchOS 11, visionOS 2.0, *)
    public func
segmentAndOffsetIntoSegment(forTimelineTi
me: CMTime) -> (AVPlayerItemSegment,
CMTime)
}

@available(macOS 15, iOS 18, tvOS 18,
watchOS 11, visionOS 2.0, *)
extension AVPlayerItemIntegratedTimeline
{

    public struct PeriodicTimes :
```

```
AsyncSequence, Sendable {

        /// The type of element produced
by this asynchronous sequence.
        public typealias Element = CMTime

        /// Creates the asynchronous
iterator that produces elements of this
        /// asynchronous sequence.
        ///
        /// - Returns: An instance of the
`AsyncIterator` type used to produce
        /// elements of the asynchronous
sequence.
        public func makeAsyncIterator()
->
AVPlayerItemIntegratedTimeline.PeriodicTi
mes.Iterator

        public struct Iterator :
AsyncIteratorProtocol {

            /// Asynchronously advances
to the next element and returns it, or
ends the
            /// sequence if there is no
next element.
            ///
            /// - Returns: The next
element, if it exists, or `nil` to signal
the end of
            ///   the sequence.
            public mutating func next()
async ->
AVPlayerItemIntegratedTimeline.PeriodicTi
mes.Element?

            @available(iOS 18, tvOS 18,
watchOS 11, visionOS 2.0, macOS 15, *)
            public typealias Element =
AVPlayerItemIntegratedTimeline.PeriodicTi
mes.Element
        }

        /// The type of asynchronous
iterator that produces elements of this
        /// asynchronous sequence.
        @available(iOS 18, tvOS 18,
watchOS 11, visionOS 2.0, macOS 15, *)
        public typealias AsyncIterator =
AVPlayerItemIntegratedTimeline.PeriodicTi
mes.Iterator
    }

    public struct BoundaryTimes :
AsyncSequence, Sendable {

        /// The type of element produced
by this asynchronous sequence.
        public typealias Element = CMTime

        /// Creates the asynchronous
iterator that produces elements of this
        /// asynchronous sequence.
        ///
        /// - Returns: An instance of the
```

```
`AsyncIterator` type used to produce
        /// elements of the asynchronous
sequence.
        public func makeAsyncIterator()
->
AVPlayerItemIntegratedTimeline.BoundaryTi
mes.Iterator

        public struct Iterator :
AsyncIteratorProtocol {

            /// Asynchronously advances
to the next element and returns it, or
ends the
            /// sequence if there is no
next element.
            ///
            /// - Returns: The next
element, if it exists, or `nil` to signal
the end of
            ///   the sequence.
            public mutating func next()
async ->
AVPlayerItemIntegratedTimeline.BoundaryTi
mes.Element?

            @available(iOS 18, tvOS 18,
watchOS 11, visionOS 2.0, macOS 15, *)
            public typealias Element =
AVPlayerItemIntegratedTimeline.BoundaryTi
mes.Element
        }
        /// The type of asynchronous
iterator that produces elements of this
        /// asynchronous sequence.
        @available(iOS 18, tvOS 18,
watchOS 11, visionOS 2.0, macOS 15, *)
        public typealias AsyncIterator =
AVPlayerItemIntegratedTimeline.BoundaryTi
mes.Iterator
    }

    /**
     Returns an asynchronous sequence of
Times periodically as playback
progresses.
     */
    public func
periodicTimes(forInterval: CMTime) ->
AVPlayerItemIntegratedTimeline.PeriodicTi
mes

    /**
     Returns an asynchronous sequence of
Times every time playback reaches
segmentTime in the segment.
     One can configure boundaryTimes for
traversal of a single point segment. If
the segment is no longer
     mappable to the current timeline,
the sequence will end.
     */
    public func boundaryTimes(for
segment: AVPlayerItemSegment,
offsetsIntoSegment: [CMTime]) ->
```

```
AVPlayerItemIntegratedTimeline.BoundaryTi
mes
}

extension AVPlayerItem {

    /**
         AVPlayerItem initializer that
loads supplied properties of an asset
automatically

         - Parameters:
            - asset: Asset to load.
            -
automaticallyLoadedAssetKeys: Asset
properties to load automatically.

         The value of each key in
automaticallyLoadedAssetKeys will be
automatically be loaded by the underlying
AVAsset before the receiver achieves the
status `.readyToPlay`; i.e. when the item
is ready to play, the value of
`AVPlayerItem.asset.status(of:)` will be
`.loaded` or `.failed`.

         This initializer, along with the
companion `asset` property, is MainActor-
isolated because AVAsset is not Sendable.
If you are using a Sendable subclass of
AVAsset, such as AVURLAsset, an overload
of this initializer will be chosen
automatically to allow you to initialize
```

```
an AVPlayerItem while not running on the
main actor.
         */
    @available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
    @MainActor @preconcurrency public
convenience init(asset: AVAsset,
automaticallyLoadedAssetKeys:
[AVPartialAsyncProperty<AVAsset>] = [])

    /**
         AVPlayerItem initializer that
can be called from any concurrency domain
when provided with a Sendable asset.

         - Parameters:
            - asset: Asset to load.
         */
    @available(macOS 10.7, iOS 4.0, tvOS
9.0, watchOS 1.0, visionOS 1.0, *)
    nonisolated public convenience
init(asset: any AVAsset & Sendable)

    /**
         AVPlayerItem initializer that
loads supplied properties of a Sendable
asset automatically and can be called
from any concurrency domain

         - Parameters:
            - asset: Asset to load.
            -
automaticallyLoadedAssetKeys: Asset
```

```
properties to load automatically.

         The value of each key in
automaticallyLoadedAssetKeys will be
automatically be loaded by the underlying
AVAsset before the receiver achieves the
status `.readyToPlay`; i.e. when the item
is ready to play, the value of
`AVPlayerItem.asset.status(of:)` will be
`.loaded` or `.failed`.
         */
    @available(macOS 12, iOS 15, tvOS 15,
watchOS 8, visionOS 1, *)
    nonisolated public convenience
init(asset: any AVAsset & Sendable,
automaticallyLoadedAssetKeys:
[AVPartialAsyncProperty<AVAsset>])

    /**
     Sets the current playback time to
the time specified by the date.

     - Parameter date: The date to which
to seek.
     - Returns: Returns true if the seek
operation completed, false if it did not.

     Use this method to seek to a
specified date in the player item and
await the operation's completion. If the
seek request completes without being
interrupted (either by another seek
request or by any other operation), this
```

```
method will return true.

     If another seek request is already
in progress when you call this method,
the  the in-progress seek request
immediately returns false.
     */
    @available(macOS 13, iOS 16, tvOS 16,
watchOS 9, visionOS 1, *)
    nonisolated public func seek(to date:
Date) async -> Bool
}

extension NSNotification.Name {

    @available(macOS, introduced: 10.7,
deprecated: 100000, message: "Use
AVPlayerItem.timeJumpedNotification
instead.")
    @available(iOS, introduced: 5.0,
deprecated: 100000, message: "Use
AVPlayerItem.timeJumpedNotification
instead.")
    @available(tvOS, introduced: 9.0,
deprecated: 100000, message: "Use
AVPlayerItem.timeJumpedNotification
instead.")
    @available(watchOS, introduced: 1.0,
deprecated: 100000, message: "Use
AVPlayerItem.timeJumpedNotification
instead.")
    @available(visionOS, introduced: 1.0,
deprecated: 100000, message: "Use
```

```
AVPlayerItem.timeJumpedNotification
instead.")
    public static var
AVPlayerItemTimeJumped:
NSNotification.Name { get }

    @available(macOS, introduced: 10.9,
deprecated: 100000, message: "Use
AVPlayerItem.playbackStalledNotification
instead.")
    @available(iOS, introduced: 6.0,
deprecated: 100000, message: "Use
AVPlayerItem.playbackStalledNotification
instead.")
    @available(tvOS, introduced: 9.0,
deprecated: 100000, message: "Use
AVPlayerItem.playbackStalledNotification
instead.")
    @available(watchOS, introduced: 1.0,
deprecated: 100000, message: "Use
AVPlayerItem.playbackStalledNotification
instead.")
    @available(visionOS, introduced: 1.0,
deprecated: 100000, message: "Use
AVPlayerItem.playbackStalledNotification
instead.")
    public static var
AVPlayerItemPlaybackStalled:
NSNotification.Name { get }

    @available(macOS, introduced: 10.9,
deprecated: 100000, message: "Use
AVPlayerItem.newErrorLogEntryNotification
```

```
instead.")
    @available(iOS, introduced: 6.0,
deprecated: 100000, message: "Use
AVPlayerItem.newErrorLogEntryNotification
instead.")
    @available(tvOS, introduced: 9.0,
deprecated: 100000, message: "Use
AVPlayerItem.newErrorLogEntryNotification
instead.")
    @available(watchOS, introduced: 1.0,
deprecated: 100000, message: "Use
AVPlayerItem.newErrorLogEntryNotification
instead.")
    @available(visionOS, introduced: 1.0,
deprecated: 100000, message: "Use
AVPlayerItem.newErrorLogEntryNotification
instead.")
    public static var
AVPlayerItemNewErrorLogEntry:
NSNotification.Name { get }

    @available(macOS, introduced: 10.9,
deprecated: 100000, message: "Use
AVPlayerItem.newAccessLogEntryNotificatio
n instead.")
    @available(iOS, introduced: 6.0,
deprecated: 100000, message: "Use
AVPlayerItem.newAccessLogEntryNotificatio
n instead.")
    @available(tvOS, introduced: 9.0,
deprecated: 100000, message: "Use
AVPlayerItem.newAccessLogEntryNotificatio
n instead.")
```

```
    @available(watchOS, introduced: 1.0,
deprecated: 100000, message: "Use
AVPlayerItem.newAccessLogEntryNotificatio
n instead.")
    @available(visionOS, introduced: 1.0,
deprecated: 100000, message: "Use
AVPlayerItem.newAccessLogEntryNotificatio
n instead.")
    public static var
AVPlayerItemNewAccessLogEntry:
NSNotification.Name { get }

    @available(macOS, introduced: 10.7,
deprecated: 100000, message: "Use
AVPlayerItem.didPlayToEndTimeNotification
instead.")
    @available(iOS, introduced: 4.0,
deprecated: 100000, message: "Use
AVPlayerItem.didPlayToEndTimeNotification
instead.")
    @available(tvOS, introduced: 9.0,
deprecated: 100000, message: "Use
AVPlayerItem.didPlayToEndTimeNotification
instead.")
    @available(watchOS, introduced: 1.0,
deprecated: 100000, message: "Use
AVPlayerItem.didPlayToEndTimeNotification
instead.")
    @available(visionOS, introduced: 1.0,
deprecated: 100000, message: "Use
AVPlayerItem.didPlayToEndTimeNotification
instead.")
    public static var
```

```
AVPlayerItemDidPlayToEndTime:
NSNotification.Name { get }

    @available(macOS, introduced: 10.7,
deprecated: 100000, message: "Use
AVPlayerItem.failedToPlayToEndTimeNotific
ation instead.")
    @available(iOS, introduced: 4.3,
deprecated: 100000, message: "Use
AVPlayerItem.failedToPlayToEndTimeNotific
ation instead.")
    @available(tvOS, introduced: 9.0,
deprecated: 100000, message: "Use
AVPlayerItem.failedToPlayToEndTimeNotific
ation instead.")
    @available(watchOS, introduced: 1.0,
deprecated: 100000, message: "Use
AVPlayerItem.failedToPlayToEndTimeNotific
ation instead.")
    @available(visionOS, introduced: 1.0,
deprecated: 100000, message: "Use
AVPlayerItem.failedToPlayToEndTimeNotific
ation instead.")
    public static var
AVPlayerItemFailedToPlayToEndTime:
NSNotification.Name { get }
}

@available(watchOS, unavailable)
extension AVAssetImageGenerator {

    /// Creates an image object for an
asset at or near specified the time.
```

```
    /// - Parameter time: The time at
which the image of the asset is to be
created.
    /// - Returns: A tuple containing the
image object as a CGImage, and the time
at which the image was actually generated
as a CMTime.
    @available(macOS 13, iOS 16, tvOS 16,
visionOS 1, *)
    @available(watchOS, unavailable)
    public func image(at time: CMTime)
async throws -> (image: CGImage,
actualTime: CMTime)

    /// Creates a series of image objects
for an asset at or near specified times.
    /// - Parameter times: An array of
times at which the images of the asset
are to be created.
    /// - Returns: The generated images
or errors for each time, as an
asynchronous sequence of Results.
    @available(macOS 13, iOS 16, tvOS 16,
visionOS 1, *)
    @available(watchOS, unavailable)
    public func images(for times:
[CMTime]) -> AVAssetImageGenerator.Images

    /// An asynchronous sequence where
each element is a Result<(requestedTime:
CMTime, image: CGImage, actualTime:
CMTime), Error>. When image generation is
successful, the result is a tuple
```

```
containing the requested time as a
CMTime, the image object as a CGImage,
and the time at which the image was
actually generated as a CMTime.
Otherwise, when image generation fails,
the result contains an Error.
    @available(macOS 13, iOS 16, tvOS 16,
visionOS 1, *)
    @available(watchOS, unavailable)
    public struct Images : AsyncSequence,
AsyncIteratorProtocol {

        /// The type of element produced
by this asynchronous sequence.
        @frozen public enum Element :
Sendable {

            case success(requestedTime:
CMTime, image: CGImage, actualTime:
CMTime)

            case failure(requestedTime:
CMTime, error: any Error)
        }

        /// Creates the asynchronous
iterator that produces elements of this
        /// asynchronous sequence.
        ///
        /// - Returns: An instance of the
`AsyncIterator` type used to produce
        /// elements of the asynchronous
sequence.
```

```
        public func makeAsyncIterator()
-> AVAssetImageGenerator.Images

        /// Asynchronously advances to
the next element and returns it, or ends
the
        /// sequence if there is no next
element.
        ///
        /// - Returns: The next element,
if it exists, or `nil` to signal the end
of
        ///   the sequence.
        public mutating func next() async
-> AVAssetImageGenerator.Images.Element?

        /// The type of asynchronous
iterator that produces elements of this
        /// asynchronous sequence.
        @available(iOS 16, tvOS 16,
visionOS 1, macOS 13, *)
        @available(watchOS, unavailable)
        public typealias AsyncIterator =
AVAssetImageGenerator.Images
    }
}

@available(macOS 15, iOS 18, tvOS 18,
visionOS 2.0, *)
@available(watchOS, unavailable)
extension AVAssetExportSession {

    /// Initiates an asset export
```

```
operation.  Progress can be monitored
using states(updateInterval:).  Thrown
errors may include:
    ///   - AVError.operationCancelled:
export operation is cancelled
    /// - Parameters:
    ///   - url: Indicates the URL of the
export session's output. You may use
UTType.preferredFilenameExtension to
obtain an appropriate path extension for
the fileType you have specified. For more
information, see
<UniformTypeIdentifiers/UTType.h>
    ///   - fileType: Indicates the type
of file to be written by the session.
    ///   - isolated: The actor on which
this async function should be isolated.
    public func export(to url: URL, as
fileType: AVFileType, isolation: isolated
(any Actor)? = #isolation) async throws

    /// Describes the state of an export
session.
    public enum State : Sendable {

        case pending

        case waiting

        case exporting(progress:
Progress)
    }
    /// Monitor the progress of an asset
export session
    /// - Parameter updateInterval: time
interval between updates while in
exporting state
    /// - Returns: sequence of asset
export session progress states
    public func states(updateInterval:
TimeInterval = .infinity) -> some
Sendable &
AsyncSequence<AVAssetExportSession.State,
Never>

}

@available(macOS 13.0, iOS 16.0, tvOS
16.0, visionOS 1.0, *)
@available(watchOS, unavailable)
extension AVRouteDetector : @unchecked
Sendable {
}

@available(macOS 13, iOS 16, tvOS 16,
watchOS 9, visionOS 1, *)
extension AVMutableComposition {

    /// Inserts all the tracks of a
timeRange of an asset into a composition.
    /// - Parameters:
    ///   - timeRange: Specifies the
timeRange of the asset to be inserted.
    ///   - asset: Specifies the asset
that contains the tracks that are to be
```

```
inserted. Only instances of AVURLAsset
and AVComposition are supported.
    ///   - time: Specifies the time at
which the inserted tracks are to be
presented by the composition.
    ///   - isolation: The actor
isolation for accessing non-Sendable
values. Inherits the calling isolation by
default.
    @backDeployed(before: macOS 15, iOS
18, tvOS 18, watchOS 11, visionOS 2)
    final public func insertTimeRange(_
timeRange: CMTimeRange, of asset:
AVAsset, at time: CMTime, isolation:
isolated (any Actor)? = #isolation) async
throws
}

@available(macOS 11.0, iOS 10.0,
macCatalyst 14.0, tvOS 17.0, *)
@available(visionOS, unavailable)
@available(watchOS, unavailable)
extension AVCapturePhotoOutput {

    @nonobjc public var
supportedFlashModes:
[AVCaptureDevice.FlashMode] { get }

    @nonobjc public var
availablePhotoPixelFormatTypes: [OSType]
{ get }

    @nonobjc public var
```

```
availableRawPhotoPixelFormatTypes:
[OSType] { get }
}

@available(macOS 10.15, iOS 11.0,
macCatalyst 14.0, tvOS 17.0, *)
@available(visionOS, unavailable)
@available(watchOS, unavailable)
extension AVCapturePhotoOutput {

    @available(macOS 10.15, iOS 11.0,
macCatalyst 14.0, tvOS 17.0, *)
    @available(visionOS, unavailable)
    @available(watchOS, unavailable)
    @nonobjc public func
supportedPhotoPixelFormatTypes(for
fileType: AVFileType) -> [OSType]
}

@available(macOS 11.0, iOS 10.0,
macCatalyst 14.0, tvOS 17.0, *)
@available(visionOS, unavailable)
@available(watchOS, unavailable)
extension AVCapturePhotoSettings {

    @nonobjc public var
availablePreviewPhotoPixelFormatTypes:
[OSType] { get }
}

@available(macOS 14.4, iOS 17.4, tvOS
17.4, visionOS 1.1, *)
@available(watchOS, unavailable)
```

```
extension AVSampleBufferVideoRenderer {

    /**
     Options for specifying the expected
upcoming PTS values for the samples that
will be enqueued.
     */
    public enum
PresentationTimeExpectation : Sendable {

        /**
             No promises about the
upcoming PTS values.
             */
        case none

        /**
             Promises that future sample
buffers will have monotonically
increasing PTS values. Only applicable
for forward playback. Calling flush
resets such expectations. Only applicable
for forward playback. Enqueueing a buffer
with a lower PTS than any previously
enqueued PTS has the potential to lead to
dropped buffers.
             */
        case monotonicallyIncreasing

        /**
             Promises that future sample
buffers will have PTS values no less than
a specified lower-bound PTS. For best
results, set minimumUpcoming regularly,
in between calls to enqueueSampleBuffer,
to advance the lower-bound PTS. Calling
flush resets such expectations. Only
applicable for forward playback.
Enqueueing a buffer with a lower PTS than
the specified PTS has the potential to
lead to dropped buffers.
             */
        case minimumUpcoming(CMTime)
    }

    /**
     Specifies the expected upcoming PTS
values for the samples that will be
enqueued. The purpose is to enable power
optimizations.
     */
    public var
presentationTimeExpectation:
AVSampleBufferVideoRenderer.PresentationT
imeExpectation
}

@available(macOS 14.2, iOS 17.2, tvOS
17.2, watchOS 10.2, visionOS 1.1, *)
extension AVPlayerVideoOutput {

    /**
        Retrieves a video frame along
with auxiliary information for display at
the specified host time.
        - Parameter hostTime: A CMTime
```

```swift
that expresses a desired host time.
        - Returns: A tuple containing the
frame, presentation timestamp, and active
configuration for the specified host
time, or nil if no frame was available
for that host time.
         - taggedBufferGroup: An array of
CMTaggedBuffers containing the frame for
the specified time.
         - presentationTime: A CMTime
whose value is the presentation time in
terms of the corresponding AVPlayerItem's
timebase for the associated
taggedBufferGroup.
         - activeConfiguration:  The
active configuration corresponding to the
associated taggedBufferGroup.
     */
    @available(macOS 14.2, iOS 17.2, tvOS
17.2, watchOS 10.2, visionOS 1.1, *)
    public func taggedBuffers(forHostTime
hostTime: CMTime) -> (taggedBufferGroup:
[CMTaggedBuffer], presentationTime:
CMTime, activeConfiguration:
AVPlayerVideoOutput.Configuration)?
}

@available(macOS 14.2, iOS 17.2, tvOS
17.2, watchOS 10.2, visionOS 1.1, *)
extension AVVideoOutputSpecification {

    /**
     Creates an instance of
```

```
AVVideoOutputSpecification initialized
with the specified tag collections.
        - Parameter tagCollections:
Expects a non-empty array of
CMTagCollections.  Tag collections are
given priority based on their position in
the array, where position i take priority
over position i+1.
        - Note: This method will produce
a fatal error if the input tagCollection
has a count of 0.
     */
    @available(macOS 14.2, iOS 17.2, tvOS
17.2, watchOS 10.2, visionOS 1.1, *)
    public convenience
init(tagCollections: [[CMTag]])

    /**
     Specifies a mapping between a tag
collection and a set of pixel buffer
attributes.
     - Parameters:
       - pixelBufferAttributes: The
client requirements for CVPixelBuffers
related to the tags in tagCollection,
expressed using the constants in
<CoreVideo/CVPixelBuffer.h>.
       - tagCollection: A single tag
collection for which these pixel buffer
attributes should map to.
     - Note: Pixel buffer attributes are
translated into output settings,
therefore, the rules of
```

```
`setOutputSettings` apply to this method
as well.
     Namely, if you set pixel buffer
attributes for a tag collection and then
output settings for that same tag
collection, your pixel buffer attributes
will be overridden and vice-versa.
     */
    @available(macOS, introduced: 14.2,
deprecated: 100000, message: "Use
setOutputSettings instead")
    @available(iOS, introduced: 17.2,
deprecated: 100000, message: "Use
setOutputSettings instead")
    @available(tvOS, introduced: 17.2,
deprecated: 100000, message: "Use
setOutputSettings instead")
    @available(watchOS, introduced: 10.2,
deprecated: 100000, message: "Use
setOutputSettings instead")
    @available(visionOS, introduced: 1.1,
deprecated: 100000, message: "Use
setOutputSettings instead")
    public func
setOutputPixelBufferAttributes(_
pixelBufferAttributes: [String : Any]?,
for tagCollection: [CMTag])

    /**
        Specifies a mapping between a tag
collection and a set of output settings.
        - Parameters:
        - outputSettings: The client
```

```
requirements for output CVPixelBuffers
related to the tags in tagCollection,
expressed using the constants in
AVVideoSettings.h. For uncompressed video
output, start with kCVPixelBuffer* keys
in <CoreVideo/CVPixelBuffer.h>. In
addition to the keys in CVPixelBuffer.h,
uncompressed video settings dictionaries
may also contain the key
`AVVideoAllowWideColorKey`.
        - tagCollection: A single tag
collection for which these pixel buffer
attributes should map to.
        - Note: If this method is called
twice on the same tag collection, the
first requested output settings will be
overridden.
     */
    @available(macOS 15.0, iOS 18.0, tvOS
18.0, watchOS 11.0, visionOS 2.0, *)
    public func setOutputSettings(_
outputSettings: [String : any Sendable]?,
for tagCollection: [CMTag])

    /**
     Tag collections held by
AVTaggedVideoOutputSpecification.
     */
    @available(macOS 14.2, iOS 17.2, tvOS
17.2, watchOS 10.2, visionOS 1.1, *)
    public var preferredTagCollections:
[[CMTag]] { get }
}
```

```
@available(macOS 14.2, iOS 17.2, tvOS
17.2, watchOS 10.2, visionOS 1.1, *)
extension
AVPlayerVideoOutput.Configuration {

    /**
     List of data channels, represented
as an array of CMTags, selected for this
configuration.
     */
    public var dataChannelDescription:
[[CMTag]] { get }
}

@available(macOS 14.2, iOS 17.2, tvOS
17.2, watchOS 10.2, visionOS 1.1, *)
extension Array where Element == CMTag {

    /**
     Creates a collection of CMTags with
the required tags to describe monoscopic
video, where there is no stereo view,
e.g. kCMTagStereoNone.
     */
    public static func
monoscopicForVideoOutput() -> [CMTag]

    /**
     Creates a collection of CMTags with
the required tags to describe basic
stereoscopic video, where both left and
right stereo eyes are present, e.g.
kCMTagStereoLeftAndRight.
     */
    public static func
stereoscopicForVideoOutput() -> [CMTag]
}

extension NSNotification.Name {

    @available(macOS, introduced: 10.7,
deprecated: 15.0, renamed:
"AVCaptureInput.Port.formatDescriptionDid
ChangeNotification")
    @available(iOS, introduced: 4.0,
deprecated: 18.0, renamed:
"AVCaptureInput.Port.formatDescriptionDid
ChangeNotification")
    @available(macCatalyst, introduced:
14.0, deprecated: 18.0, renamed:
"AVCaptureInput.Port.formatDescriptionDid
ChangeNotification")
    @available(tvOS, introduced: 17.0,
deprecated: 18.0, renamed:
"AVCaptureInput.Port.formatDescriptionDid
ChangeNotification")
    @available(visionOS, unavailable)
    @available(watchOS, unavailable)
    public static var
AVCaptureInputPortFormatDescriptionDidCha
nge: NSNotification.Name { get }
}

@available(macOS 15, iOS 18, visionOS 2,
*)
```

```
@available(watchOS, unavailable)
@available(tvOS, unavailable)
extension AVAssetDownloadStorageManager :
@unchecked Sendable {
}

@available(macOS 12.3, iOS 15.4, tvOS
15.4, visionOS 1, *)
@available(watchOS, unavailable)
extension AVCoordinatedPlaybackSuspension
: @unchecked Sendable {
}

@available(macOS 15, iOS 18, tvOS 18,
visionOS 2, *)
@available(watchOS, unavailable)
extension AVPlaybackCoordinator :
@unchecked Sendable {
}

@available(macOS 10.7, iOS 5.0,
macCatalyst 14.0, tvOS 17.0, *)
@available(visionOS, unavailable)
@available(watchOS, unavailable)
extension AVCaptureVideoDataOutput {

    @nonobjc public var
availableVideoPixelFormatTypes: [OSType]
{ get }
}

@available(macOS 11.0, iOS 10.0,
macCatalyst 14.0, tvOS 17.0, *)
```

```
@available(visionOS, unavailable)
@available(watchOS, unavailable)
extension AVCaptureDevice.Format {

    @nonobjc public var
supportedColorSpaces:
[AVCaptureColorSpace] { get }
}

@available(macOS 13.0, iOS 16.0, tvOS
17.0, *)
@available(visionOS, unavailable)
@available(watchOS, unavailable)
extension AVCaptureDevice.Format {

    @nonobjc public var
supportedMaxPhotoDimensions:
[CMVideoDimensions] { get }
}

@available(macOS 13.0, iOS 16.0, tvOS
17.0, *)
@available(visionOS, unavailable)
@available(watchOS, unavailable)
extension AVCaptureDevice.Format {

    @nonobjc public var
secondaryNativeResolutionZoomFactors:
[CGFloat] { get }
}

@available(macOS 14.2, iOS 17.2, tvOS
17.2, *)
```

```
@available(visionOS, unavailable)
@available(watchOS, unavailable)
extension AVCaptureDevice.Format {

    @nonobjc public var
supportedVideoZoomRangesForDepthDataDeliv
ery: [ClosedRange<CGFloat>] { get }
}

@available(macOS 15.0, iOS 18.0, tvOS
18.0, *)
@available(visionOS, unavailable)
@available(watchOS, unavailable)
extension AVCaptureDevice.Format {

    @available(macOS 15.0, iOS 18.0, tvOS
18.0, *)
    @available(visionOS, unavailable)
    @available(watchOS, unavailable)
    @nonobjc public var
systemRecommendedVideoZoomRange:
ClosedRange<CGFloat>? { get }

    @available(macOS 15.0, iOS 18.0, tvOS
18.0, *)
    @available(visionOS, unavailable)
    @available(watchOS, unavailable)
    @nonobjc public var
systemRecommendedExposureBiasRange:
ClosedRange<Float>? { get }
}

extension NSNotification.Name {
    @available(macOS, introduced: 10.7,
deprecated: 15.0, renamed:
"AVCaptureDevice.wasConnectedNotification
")
    @available(iOS, introduced: 4.0,
deprecated: 18.0, renamed:
"AVCaptureDevice.wasConnectedNotification
")
    @available(macCatalyst, introduced:
14.0, deprecated: 18.0, renamed:
"AVCaptureDevice.wasConnectedNotification
")
    @available(tvOS, introduced: 17.0,
deprecated: 18.0, renamed:
"AVCaptureDevice.wasConnectedNotification
")
    @available(visionOS, unavailable)
    @available(watchOS, unavailable)
    public static var
AVCaptureDeviceWasConnected:
NSNotification.Name { get }

    @available(macOS, introduced: 10.7,
deprecated: 15.0, renamed:
"AVCaptureDevice.wasDisconnectedNotificat
ion")
    @available(iOS, introduced: 4.0,
deprecated: 18.0, renamed:
"AVCaptureDevice.wasDisconnectedNotificat
ion")
    @available(macCatalyst, introduced:
14.0, deprecated: 18.0, renamed:
```

```
"AVCaptureDevice.wasDisconnectedNotificat
ion")
    @available(tvOS, introduced: 17.0,
deprecated: 18.0, renamed:
"AVCaptureDevice.wasDisconnectedNotificat
ion")
    @available(visionOS, unavailable)
    @available(watchOS, unavailable)
    public static var
AVCaptureDeviceWasDisconnected:
NSNotification.Name { get }
}

extension AVMovie {

    /**
     Initializes an instance of AVMovie
for inspection of a media resource.

     - Parameters:
        - URL: A URL that references a
media resource.
    */
    @available(macOS 10.10, iOS 13.0,
watchOS 6.0, visionOS 1.0, *)
    @available(tvOS, unavailable)
    public convenience init(url: URL)
}

extension AVURLAsset {

    /**
     Initializes an instance of
```

```
AVURLAsset for inspection of a media
resource.

     - Parameters:
        - URL: A URL that references a
media resource.
    */
    @available(macOS 10.7, iOS 4.0, tvOS
9.0, watchOS 1.0, visionOS 1.0, *)
    public convenience init(url: URL)
}
```