

Harnessing Metadata in Audiovisual Media

Session 505

Adam Sonnanstine

AVFoundation Engineer

What Is Metadata?

Data that describes audio, video, etc. in a movie file or stream

iTunes Metadata

Bridge Song

Summary Info Video Sorting Options Lyrics Artwork

Name
Bridge Song

Artist
The Bridge Singers

Year
2014

Album Artist

Track Number
7 of 21

Album
The Bridge Album

Disc Number
 of

Grouping

BPM

Composer

Comments

Genre
New Age

Part of a compilation

Cancel OK

Bridge Song

Summary Info Video Sorting Options Lyrics Artwork

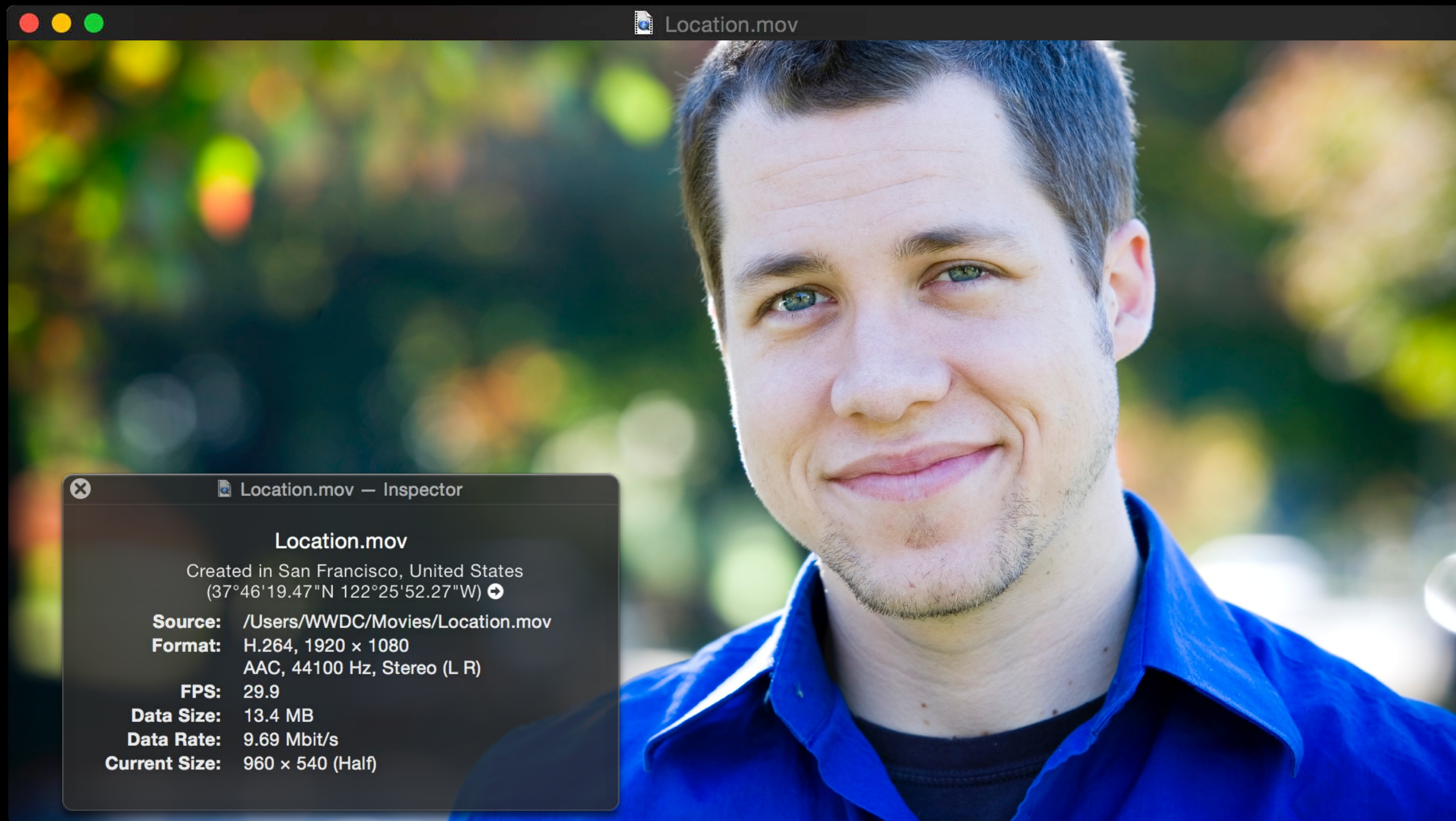
The Bridge Album

The Bridge Singers

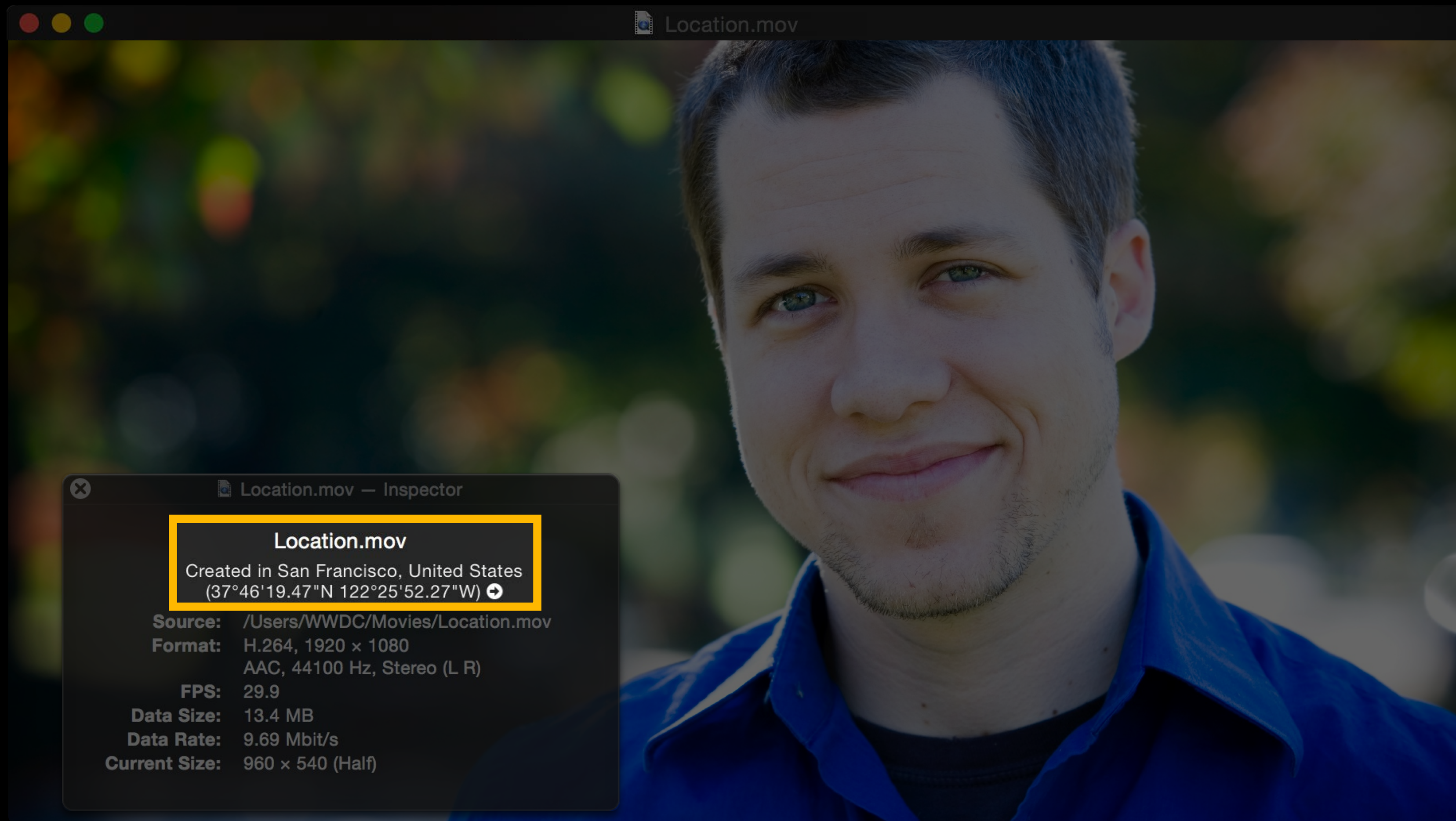
Add... Delete

Cancel OK

Location

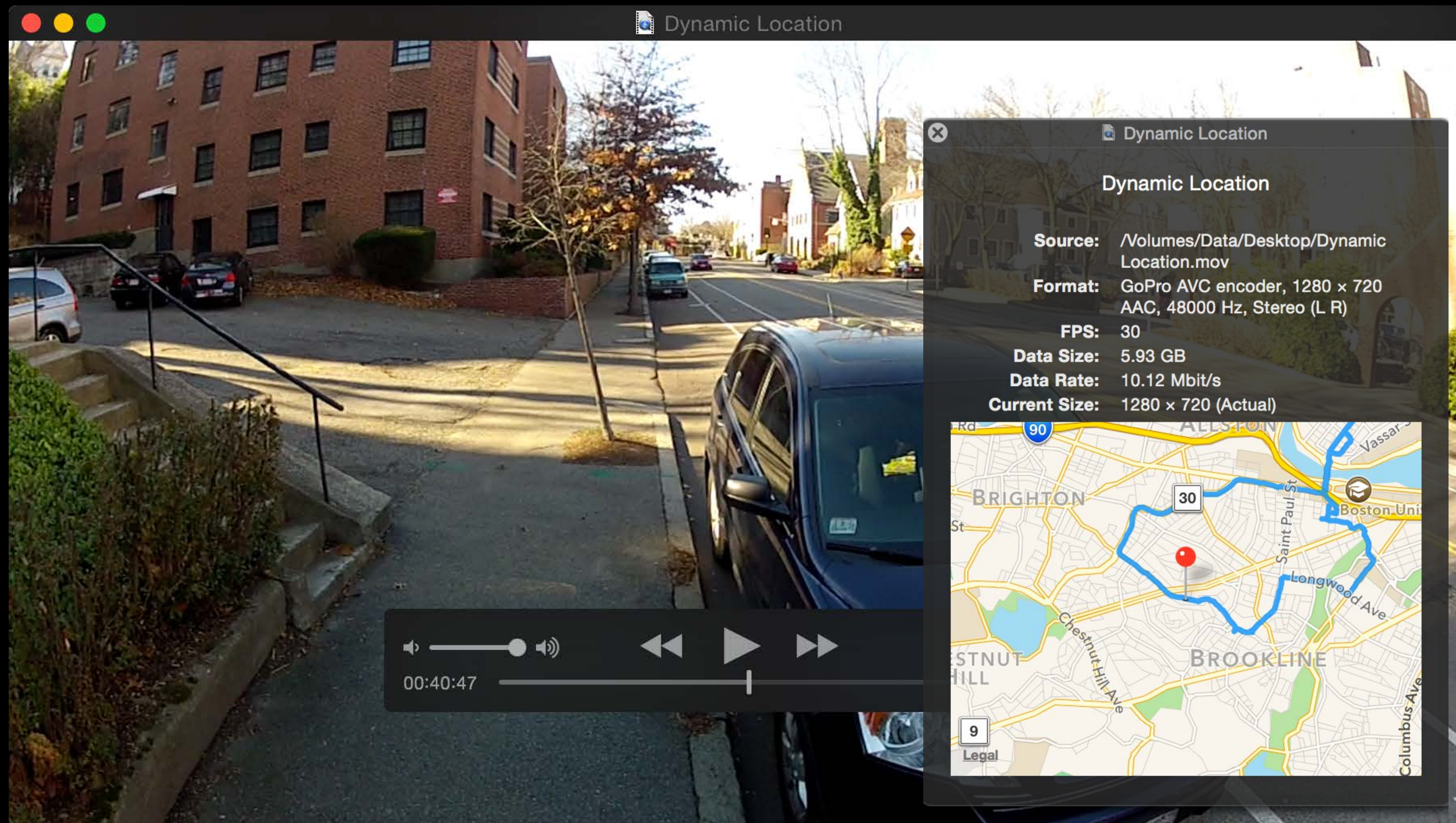


Location



Dynamic Location

NEW



The image shows a video player window titled "Dynamic Location" displaying a street scene. The video is at the 00:40:47 mark. A metadata window is overlaid on the right side of the video, providing technical details about the video file.

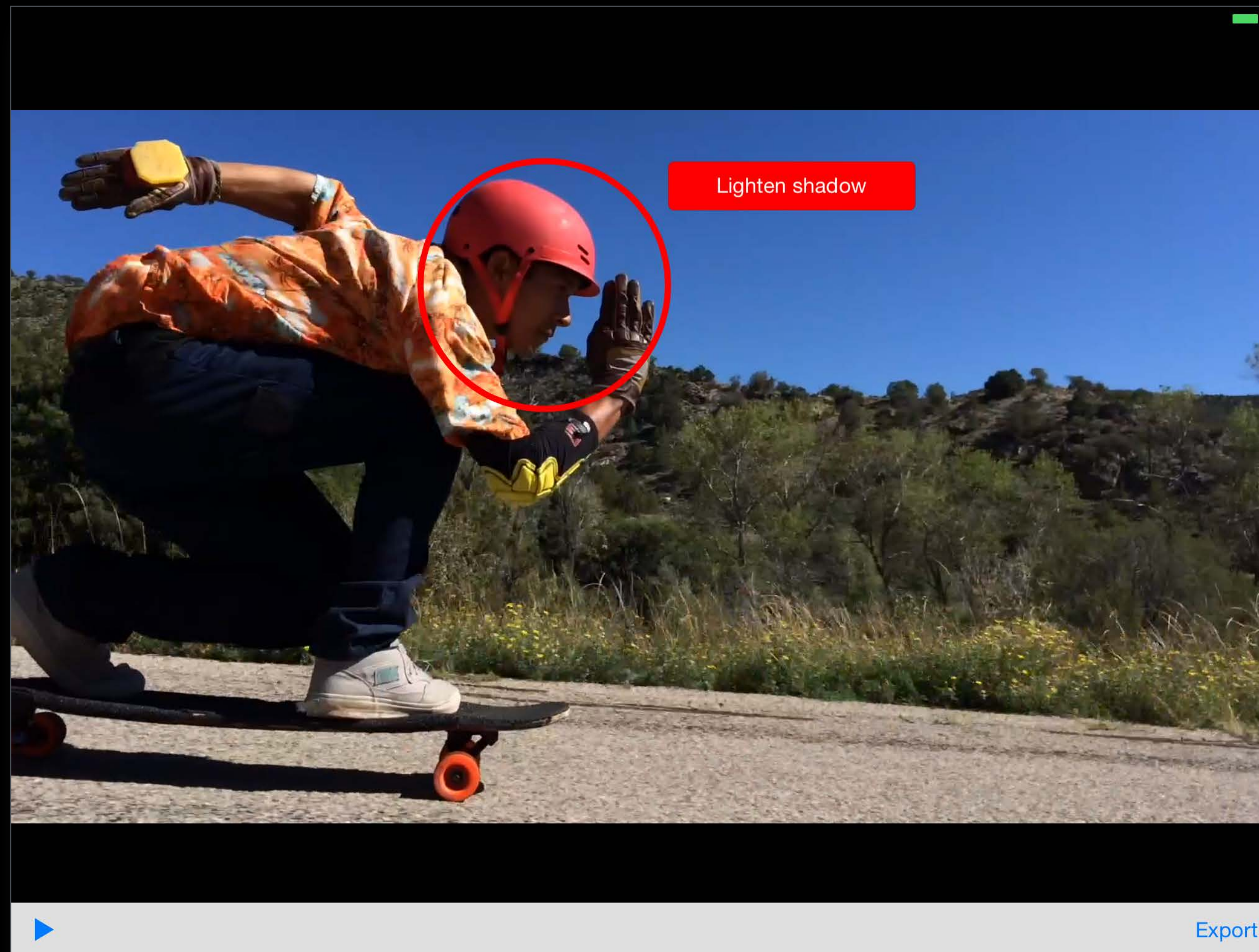
Dynamic Location

- Source:** /Volumes/Data/Desktop/Dynamic Location.mov
- Format:** GoPro AVC encoder, 1280 x 720 AAC, 48000 Hz, Stereo (L R)
- FPS:** 30
- Data Size:** 5.93 GB
- Data Rate:** 10.12 Mbit/s
- Current Size:** 1280 x 720 (Actual)

The metadata window also includes a map showing the video's location in Brighton, Brookline, and Chestnut Hill, with a red pin indicating the current position. The map shows streets like Saint Paul St, Longwood Ave, and Chestnut Hill Ave, and landmarks like Boston University and the 9 Legal area.

Your Metadata

NEW



Agenda

Metadata in AVFoundation

- Inspection
- Authoring

Timed Metadata

- Reading and playback
- Authoring

Privacy

Authoring best practices

Metadata in AVFoundation

Timed Metadata

Privacy

Authoring Best Practices

Model Objects

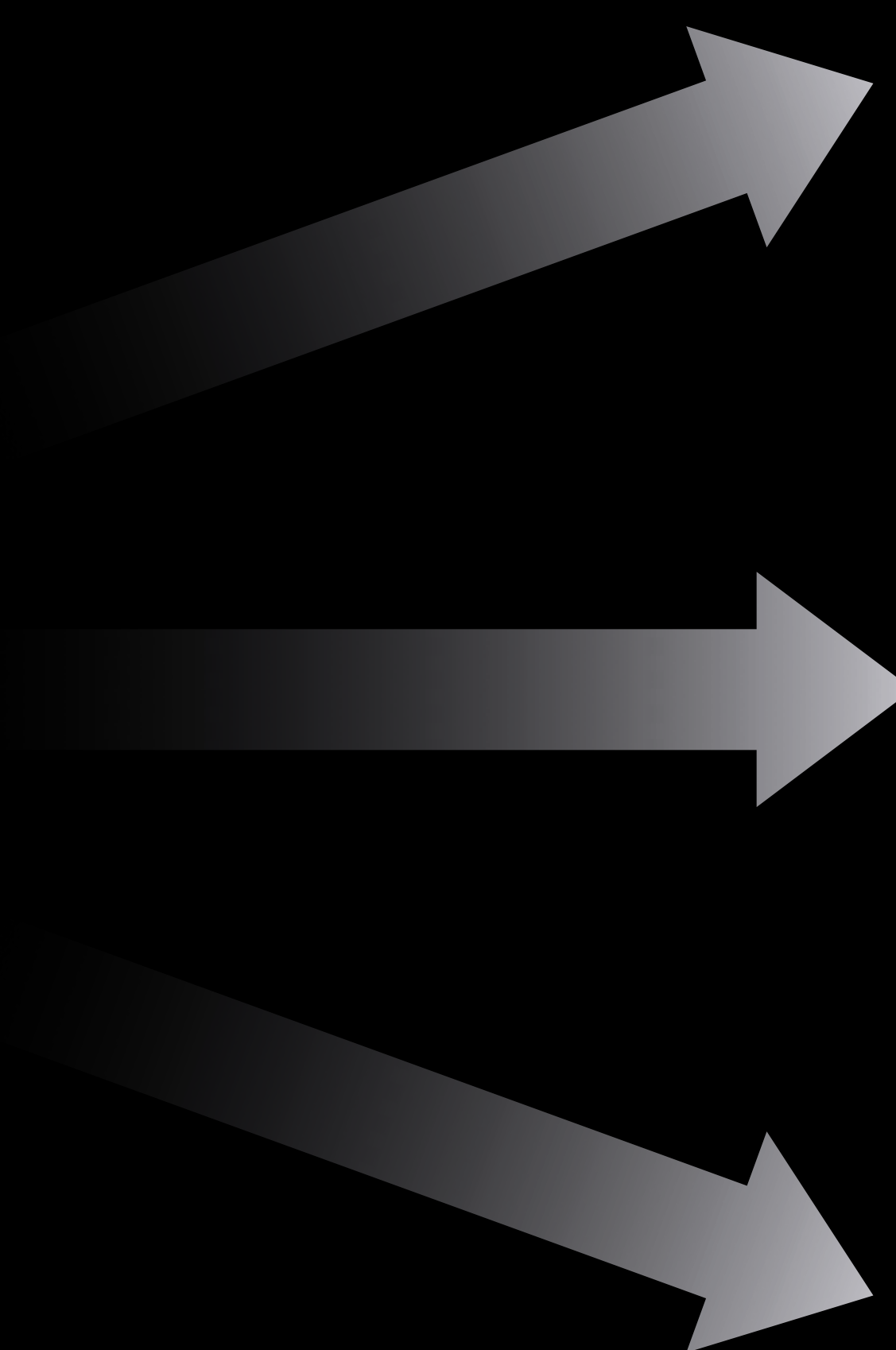
AVFoundation



AVAsset

Model Objects

AVFoundation



AVMetadataItem

Title: "Bridge Song"

This block represents an AVMetadataItem with the title "Bridge Song". It is contained within a green rectangular box.

AVMetadataItem



The Bridge Album
The Bridge Singers

This block represents an AVMetadataItem containing an album cover. The cover image shows a bridge over water under a blue sky. The text "The Bridge Album" is at the top and "The Bridge Singers" is at the bottom of the image. The entire block is in a green rectangular box.

AVMetadataItem

38.44°N
122.71°W

This block represents an AVMetadataItem with geographic coordinates. The text "38.44°N" and "122.71°W" is displayed in two lines. The entire block is in a green rectangular box.

AVMetadataItem

<AVFoundation/AVMetadataItem.h>

AVMetadataItem

AVMetadataItem

<AVFoundation/AVMetadataItem.h>



AVMetadataItem

AVMetadataIdentifieriTunesMetadataSongName

@“Bridge Song”

identifier: NSString

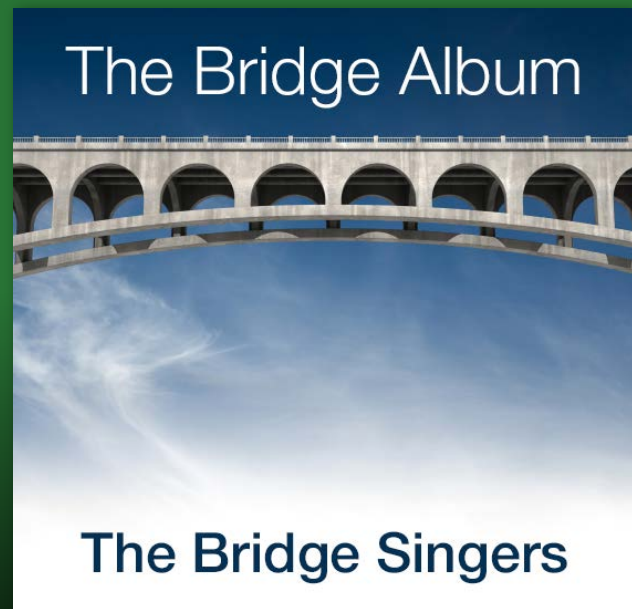
value: id <NSObject, NSCopying>

AVMetadataItem

<AVFoundation/AVMetadataItem.h>

AVMetadataItem

AVMetadataIdentifieriTunesMetadataCoverArt



identifier: NSString

value: id <NSObject, NSCopying>

Identifier = key space + key

Metadata Identifiers

<AVFoundation/AVMetadataIdentifiers.h>



Metadata Identifiers

<AVFoundation/AVMetadataIdentifiers.h>



AVMetadataIdentifieriTunesMetadataSongName

AVMetadataIdentifieriTunesMetadataCopyright

AVMetadataIdentifieriTunesMetadataCoverArt

...

AVMetadataIdentifierQuickTimeMetadataTitle

AVMetadataIdentifierQuickTimeMetadataCopyright

AVMetadataIdentifierQuickTimeMetadataLocationISO6709

...

AVMetadataIdentifier3GPUUserDataTitle

AVMetadataIdentifier3GPUUserDataCopyright

AVMetadataIdentifier3GPUUserDataAuthor

...

AVMetadataItem

Payload data type



AVMetadataItem

AVMetadataIdentifieriTunesMetadataSongName

@“Bridge Song”

kCMMetadataBaseDataType_UTF8

dataType: NSString

<CoreMedia/CMMetadata.h>

AVMetadataItem

Payload data type



AVMetadataItem

AVMetadataIdentifieriTunesMetadataSongName

@“Bridge Song”

stringValue: NSString

@“Bridge Song”

kCMMetadataBaseDataType_UTF8



AVMetadataItem

Payload data type



AVMetadataItem

AVMetadataIdentifieriTunesMetadataSongName

@“Bridge Song”

kCMMetadataBaseDataType_UTF8

stringValue: NSString

@“Bridge Song”

numberValue: NSNumber

nil



AVMetadataItem

Payload data type



AVMetadataItem

AVMetadataIdentifieriTunesMetadataSongName

@“Bridge Song”

kCMMetadataBaseDataType_ **UTF8**

stringValue: NSString

@“Bridge Song”

numberValue: NSNumber

nil

dateValue: NSDate

nil

AVMetadataItem

Payload data type



AVMetadataItem

AVMetadataIdentifieriTunesMetadataSongName

@“Bridge Song”

kCMMetadataBaseDataType_UTF8

stringValue: NSString

@“Bridge Song”

numberValue: NSNumber

nil

dateValue: NSDate

nil

dataValue: NSData

nil

AVMetadataItem

Payload data type

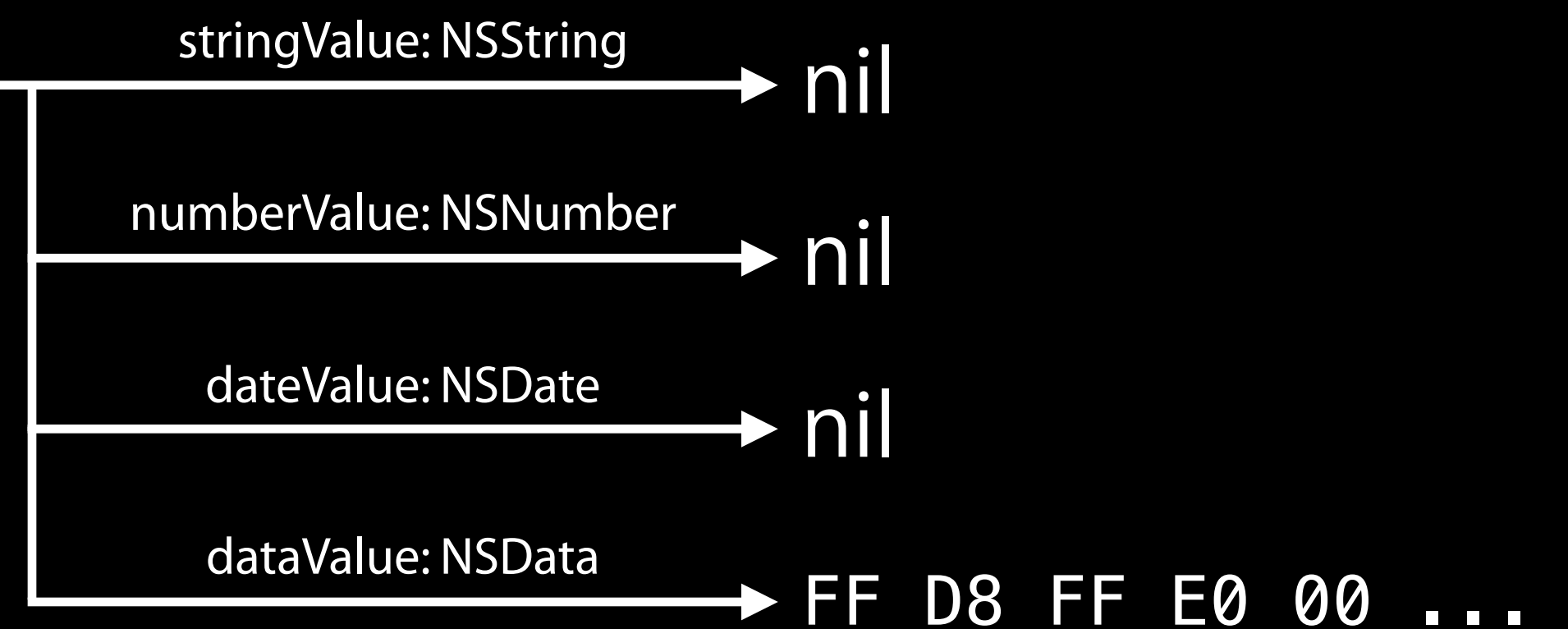


AVMetadataItem

AVMetadataIdentifieriTunesMetadataCoverArt



kCMMetadataBaseDataType_JPEG



AVMetadataItem

Payload data type

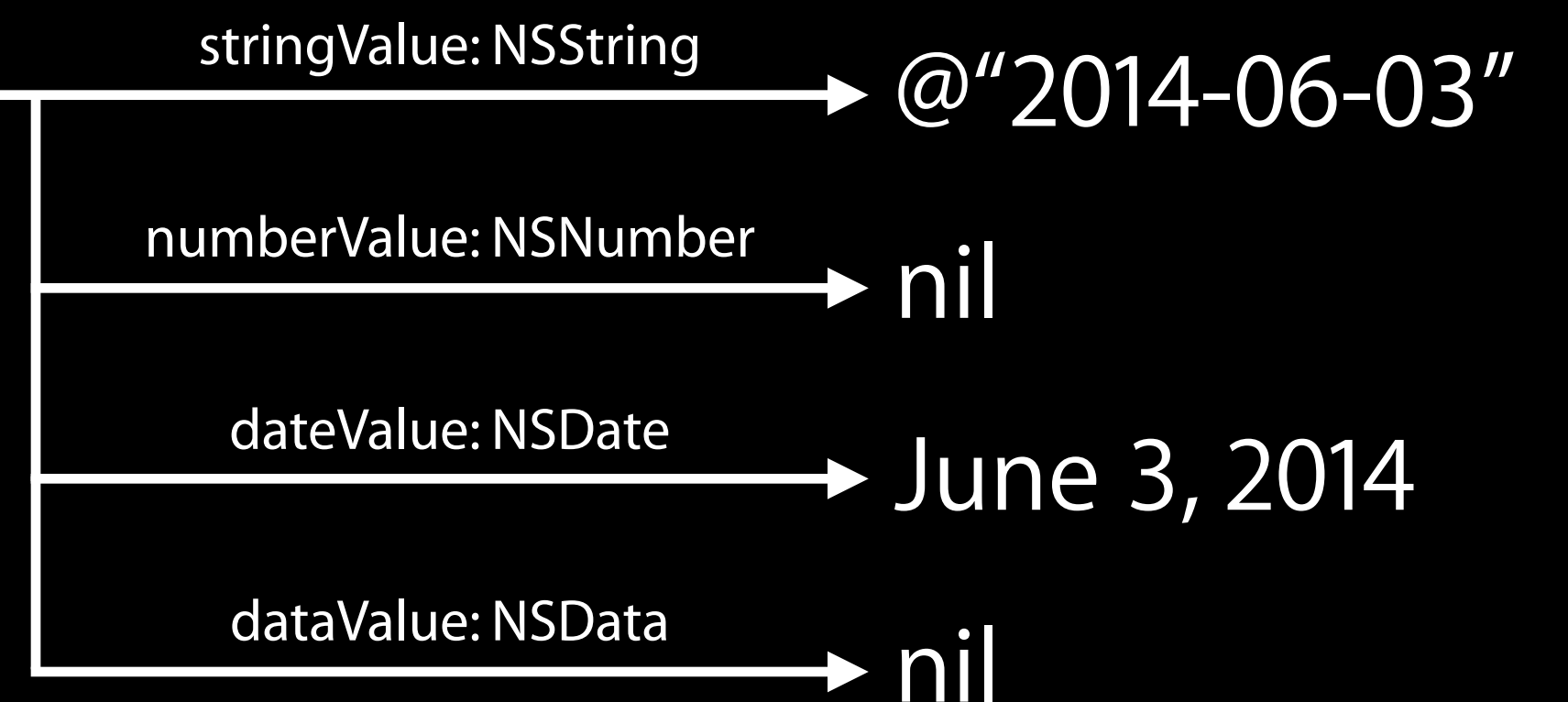


AVMetadataItem

AVMetadataIdentifierQuickTimeMetadataCreationDate

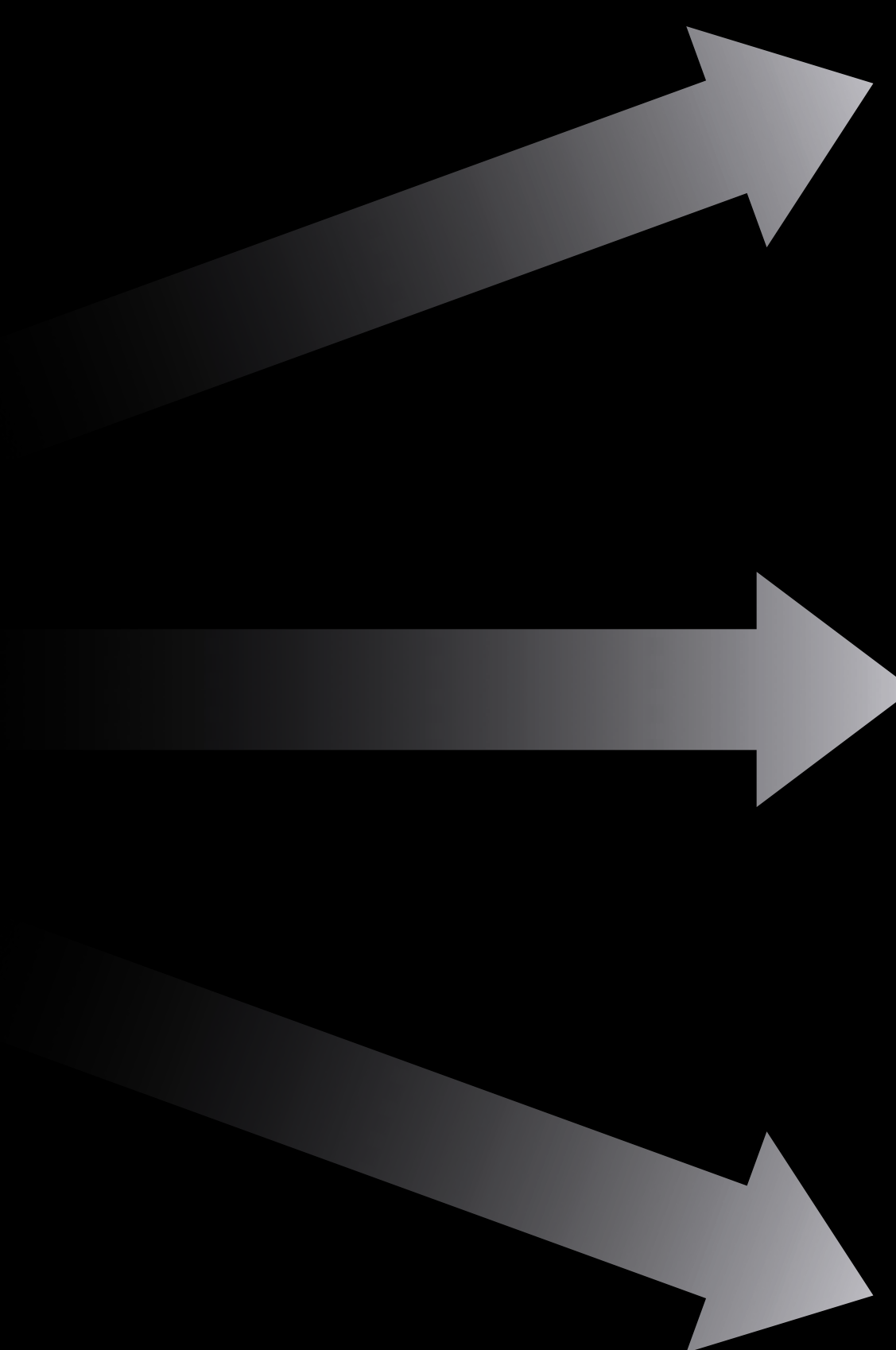
@"2014-06-03"

kCMMetadataBaseDataType_UTF8



Model Objects

AVFoundation



AVMetadataItem

Title: "Bridge Song"

This block is a green rectangle containing the text 'AVMetadataItem' at the top and 'Title: "Bridge Song"' below it.

AVMetadataItem



The Bridge Album
The Bridge Singers

This block is a green rectangle containing the text 'AVMetadataItem' at the top. Below it is a small image of an album cover for 'The Bridge Album' by 'The Bridge Singers'. The cover features a bridge over water under a blue sky.

AVMetadataItem

38.44°N
122.71°W

This block is a green rectangle containing the text 'AVMetadataItem' at the top and the coordinates '38.44°N' and '122.71°W' below it.

Read iTunes Metadata

<AVFoundation/AVAsset.h>

```
@property (nonatomic, readonly) NSArray *metadata;
```

```
@property (nonatomic, readonly) NSArray *availableMetadataFormats;
```

```
- (NSArray *)metadataForFormat:(NSString *)format;
```

Get all iTunes metadata:

```
iTunesArray = [asset metadataForFormat: AVMetadataFormatiTunesMetadata];
```

Get title(s) (AVMetadataItem.h, AVMetadataItemArrayFiltering protocol):

```
titleArray = [AVMetadataItem metadataItemsFromArray: iTunesArray  
              filteredByIdentifier: AVMetadataIdentifieriTunesMetadataSongName];
```


Read iTunes Metadata

<AVFoundation/AVAsset.h>

```
@property (nonatomic, readonly) NSArray *metadata;
```

```
@property (nonatomic, readonly) NSArray *availableMetadataFormats;
```

```
- (NSArray *)metadataForFormat:(NSString *)format;
```

Get all iTunes metadata:

```
iTunesArray = [asset metadataForFormat: AVMetadataFormatiTunesMetadata];
```

Get title(s) (AVMetadataItem.h, AVMetadataItemArrayFiltering protocol):

```
titleArray = [AVMetadataItem metadataItemsFromArray: iTunesArray  
             filteredByIdentifier: AVMetadataIdentifieriTunesMetadataSongName];
```

Read iTunes Metadata

<AVFoundation/AVAsset.h>

```
@property (nonatomic, readonly) NSArray *metadata;
```

```
@property (nonatomic, readonly) NSArray *availableMetadataFormats;
```

```
- (NSArray *)metadataForFormat:(NSString *)format;
```

Get all iTunes metadata:

```
iTunesArray = [asset metadataForFormat: AVMetadataFormatiTunesMetadata];
```

Get title(s) (AVMetadataItem.h, AVMetadataItemArrayFiltering protocol):

```
titleArray = [AVMetadataItem metadataItemsFromArray: iTunesArray  
             filteredByIdentifier: AVMetadataIdentifieriTunesMetadataSongName];
```

Read iTunes Metadata

<AVFoundation/AVAsset.h>

```
@property (nonatomic, readonly) NSArray *metadata;
```

```
@property (nonatomic, readonly) NSArray *availableMetadataFormats;
```

```
- (NSArray *)metadataForFormat:(NSString *)format;
```

Get all iTunes metadata:

```
iTunesArray = [asset metadataForFormat: AVMetadataFormatiTunesMetadata];
```

Get title(s) (AVMetadataItem.h, AVMetadataItemArrayFiltering protocol):

```
titleArray = [AVMetadataItem metadataItemsFromArray: iTunesArray  
              filteredByIdentifier: AVMetadataIdentifieriTunesMetadataSongName];
```


Read iTunes Metadata

<AVFoundation/AVAsynchronousKeyValueLoading.h>

```
AVMetadataItem *title = [titleArray firstObject];
```

```
[title loadValuesAsynchronouslyForKeys: @[ @"value" ] completionHandler: ^{  
    NSError *error = nil;  
  
    if ([title statusOfValueForKey: @"value" error: &error] ==  
        AVKeyValueStatusLoaded) {  
        NSString *title = [title stringValue];  
        // use title string  
    } else {  
        // handle error  
    }  
}];
```

Read iTunes Metadata

<AVFoundation/AVAsynchronousKeyValueLoading.h>

```
AVMetadataItem *title = [titleArray firstObject];

[title loadValuesAsynchronouslyForKeys: @[ @"value" ] completionHandler: ^{
    NSError *error = nil;

    if ([title statusOfValueForKey: @"value" error: &error] ==
        AVKeyValueStatusLoaded) {
        NSString *title = [title stringValue];
        // use title string
    } else {
        // handle error
    }
}];
```

Read iTunes Metadata

<AVFoundation/AVAsynchronousKeyValueLoading.h>

```
AVMetadataItem *title = [titleArray firstObject];

[title loadValuesAsynchronouslyForKeys: @[ @"value" ] completionHandler: ^{
    NSError *error = nil;

    if ([title statusOfValueForKey: @"value" error: &error] ==
        AVKeyValueStatusLoaded) {
        NSString *title = [title stringValue];
        // use title string
    } else {
        // handle error
    }
}];
```


Read iTunes Metadata

<AVFoundation/AVAsynchronousKeyValueLoading.h>

```
AVMetadataItem *title = [titleArray firstObject];

[title loadValuesAsynchronouslyForKeys: @[ @"value" ] completionHandler: ^{
    NSError *error = nil;

    if ([title statusOfValueForKey: @"value" error: &error] ==
        AVKeyValueStatusLoaded) {
        NSString *title = [title stringValue];
        // use title string
    } else {
        // handle error
    }
}];
```

AVMetadataItem

Localization

An asset can have the same metadata in multiple languages

AVMetadataItem

Localization

An asset can have the same metadata in multiple languages

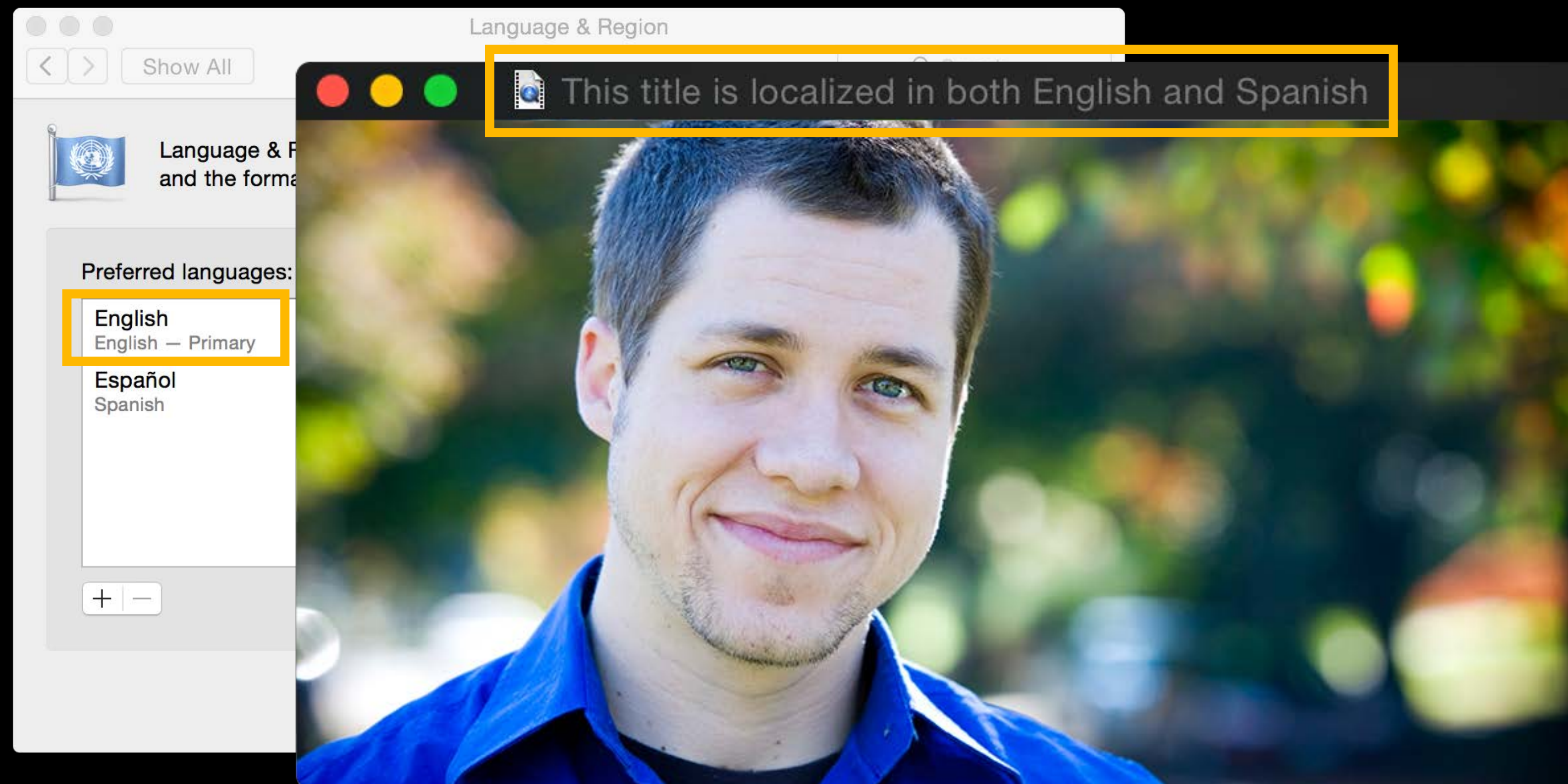
Example: `AVMetadataIdentifierQuickTimeUserDataFullName`

AVMetadataItem

Localization

An asset can have the same metadata in multiple languages

Example: `AVMetadataIdentifierQuickTimeUserDataFullName`

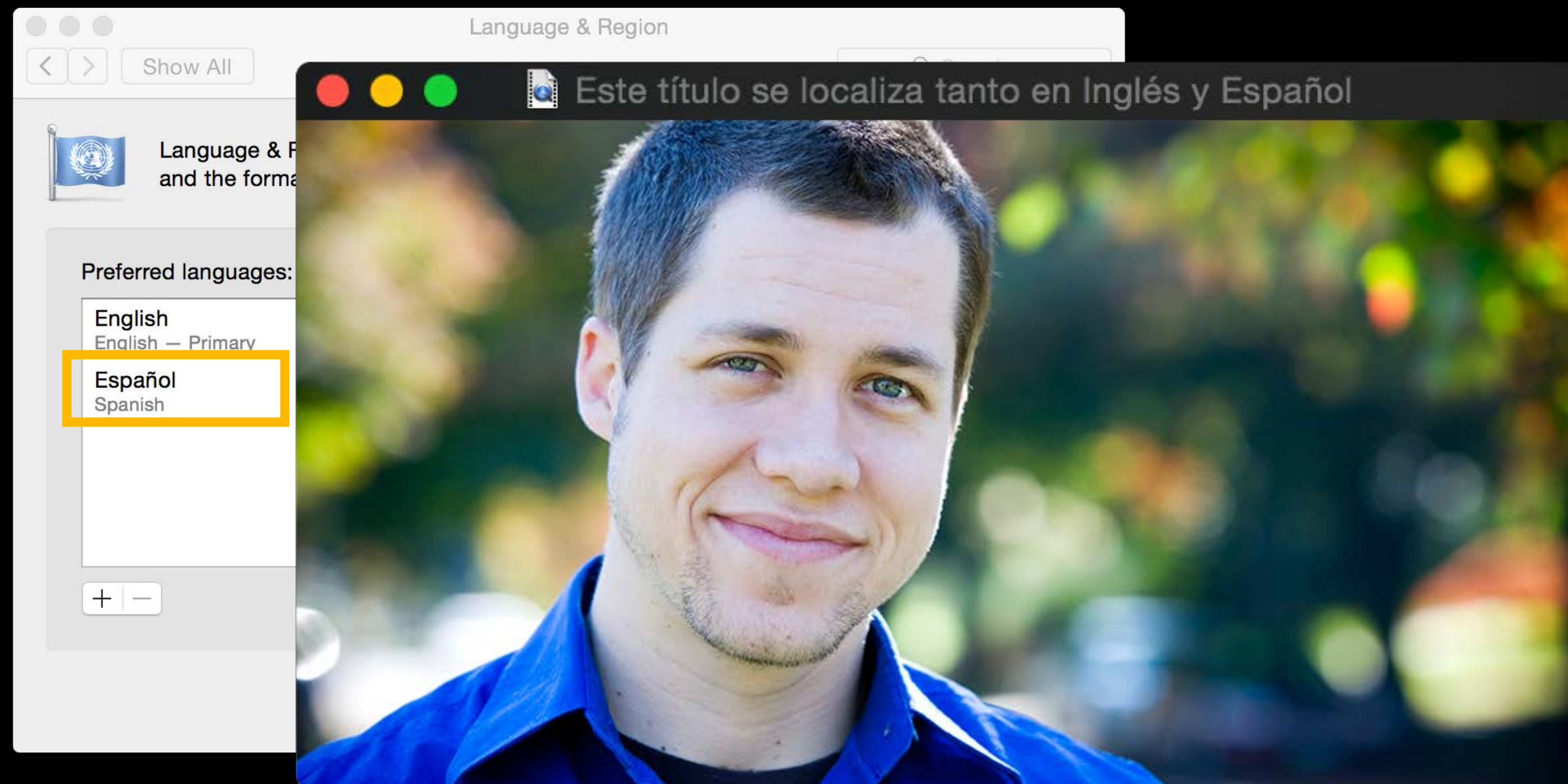


AVMetadataItem

Localization

An asset can have the same metadata in multiple languages

Example: `AVMetadataIdentifierQuickTimeUserDataFullName`

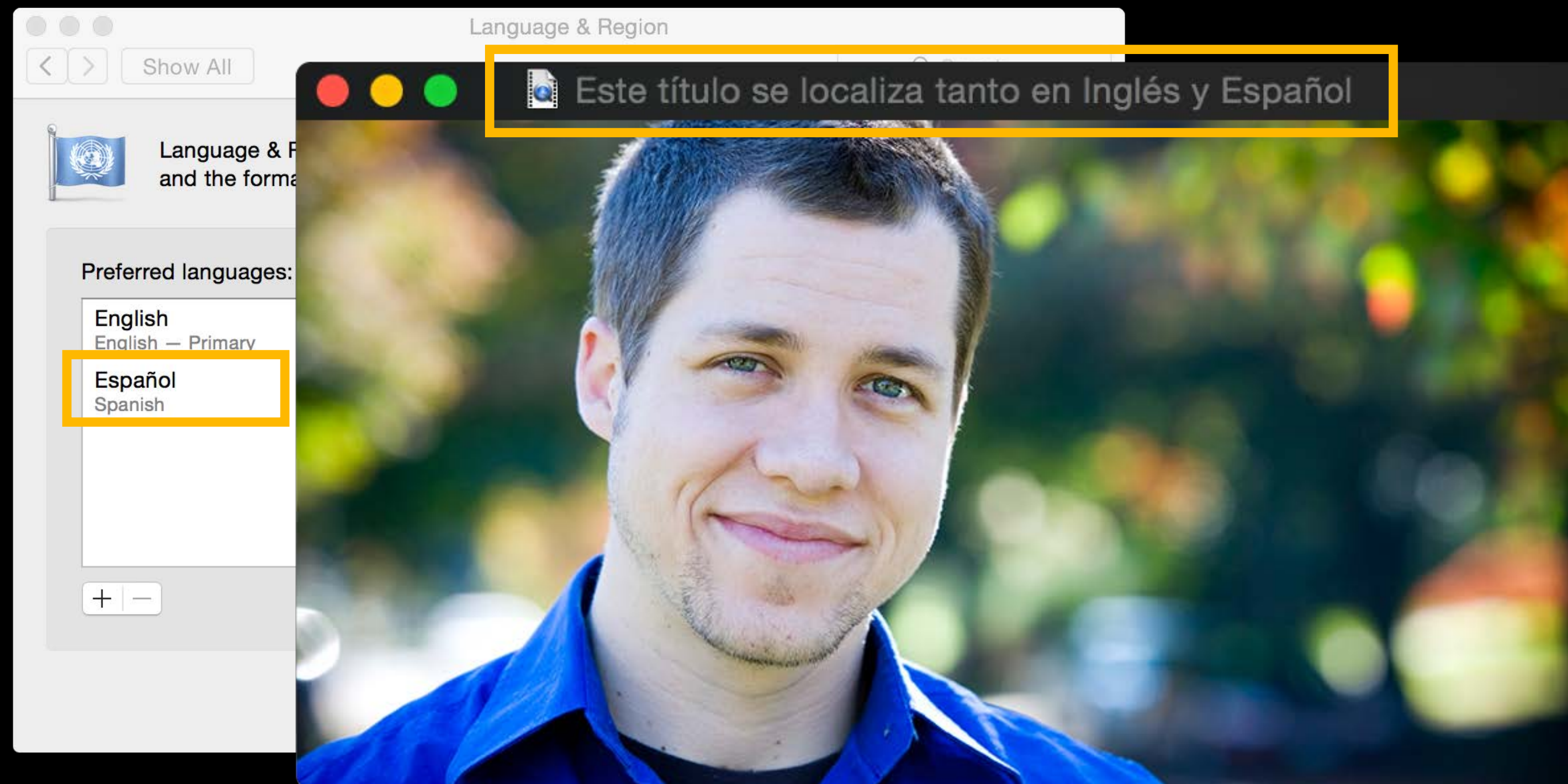


AVMetadataItem

Localization

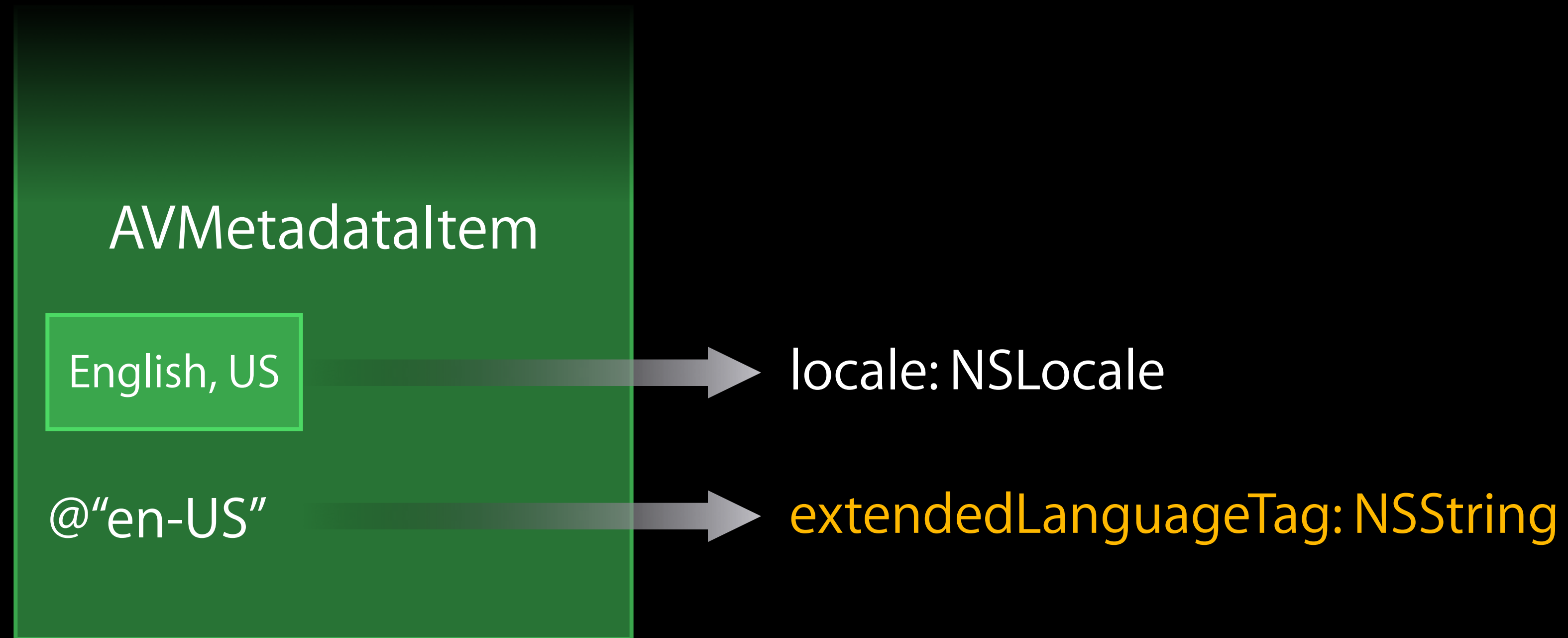
An asset can have the same metadata in multiple languages

Example: `AVMetadataIdentifierQuickTimeUserDataFullName`



AVMetadataItem

Localization



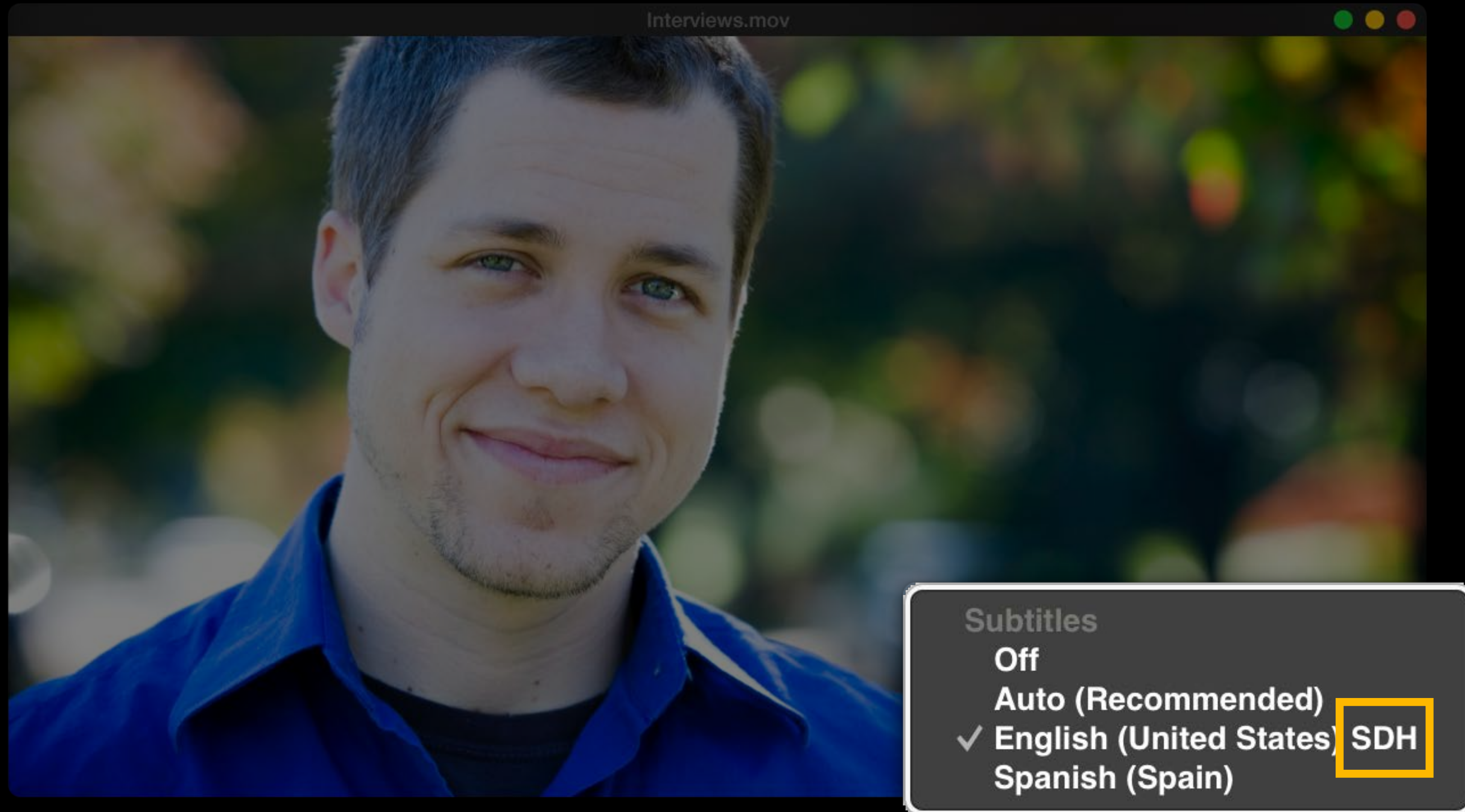
Accessibility Support

Tagged characteristics



Accessibility Support

Tagged characteristics



Read Tagged Characteristics

<AVFoundation/AVAssetTrack.h>

```
@property (nonatomic, readonly) NSArray *metadata;
```

```
@property (nonatomic, readonly) NSArray *availableMetadataFormats;
```

```
- (NSArray *)metadataForFormat:(NSString *)format;
```

Get all QuickTime user data:

```
userData = [track metadataForFormat: AVMetadataFormatQuickTimeUserData];
```

Get tagged characteristics:

```
characteristics = [AVMetadataItem metadataItemsFromArray: userData  
                  filteredByIdentifier:  
                  AVMetadataIdentifierQuickTimeUserDataTaggedCharacteristic];
```


Read Tagged Characteristics

<AVFoundation/AVAssetTrack.h>

```
@property (nonatomic, readonly) NSArray *metadata;
```

```
@property (nonatomic, readonly) NSArray *availableMetadataFormats;
```

```
- (NSArray *)metadataForFormat:(NSString *)format;
```

Get all QuickTime user data:

```
userData = [track metadataForFormat: AVMetadataFormatQuickTimeUserData];
```

Get tagged characteristics:

```
characteristics = [AVMetadataItem metadataItemsFromArray: userData  
                  filteredByIdentifier:  
                  AVMetadataIdentifierQuickTimeUserDataTaggedCharacteristic];
```

Read Tagged Characteristics

<AVFoundation/AVAssetTrack.h>

```
@property (nonatomic, readonly) NSArray *metadata;
```

```
@property (nonatomic, readonly) NSArray *availableMetadataFormats;
```

```
- (NSArray *)metadataForFormat:(NSString *)format;
```

Get all QuickTime user data:

```
userData = [track metadataForFormat: AVMetadataFormatQuickTimeUserData];
```

Get tagged characteristics:

```
characteristics = [AVMetadataItem metadataItemsFromArray: userData  
                  filteredByIdentifier:  
                  AVMetadataIdentifierQuickTimeUserDataTaggedCharacteristic];
```

Read Tagged Characteristics

<AVFoundation/AVAssetTrack.h>

```
@property (nonatomic, readonly) NSArray *metadata;
```

```
@property (nonatomic, readonly) NSArray *availableMetadataFormats;
```

```
- (NSArray *)metadataForFormat:(NSString *)format;
```

Get all QuickTime user data:

```
userData = [track metadataForFormat: AVMetadataFormatQuickTimeUserData];
```

Get tagged characteristics:

```
characteristics = [AVMetadataItem metadataItemsFromArray: userData  
                  filteredByIdentifier:  
                  AVMetadataIdentifierQuickTimeUserDataTaggedCharacteristic];
```

Metadata Identifiers

Common identifiers

AVMetadataIdentifieriTunesMetadataSongName

AVMetadataIdentifieriTunesMetadataCopyright

AVMetadataIdentifieriTunesMetadataCoverArt

...

AVMetadataIdentifierQuickTimeMetadataTitle

AVMetadataIdentifierQuickTimeMetadataCopyright

AVMetadataIdentifierQuickTimeMetadataLocationISO6709

...

AVMetadataIdentifier3GPUUserDataTitle

AVMetadataIdentifier3GPUUserDataCopyright

AVMetadataIdentifier3GPUUserDataAuthor

...

Metadata Identifiers

Common identifiers

AVMetadataIdentifieriTunesMetadataSongName

AVMetadataIdentifieriTunesMetadataCopyright

AVMetadataIdentifieriTunesMetadataCoverArt

...

AVMetadataIdentifierQuickTimeMetadataTitle

AVMetadataIdentifierQuickTimeMetadataCopyright

AVMetadataIdentifierQuickTimeMetadataLocationISO6709

...

AVMetadataIdentifier3GPUUserDataTitle

AVMetadataIdentifier3GPUUserDataCopyright

AVMetadataIdentifier3GPUUserDataAuthor

...

Metadata Identifiers

Common identifiers

AVMetadataIdentifieriTunesMetadata**SongName**

AVMetadataIdentifieriTunesMetadataCopyright

AVMetadataIdentifieriTunesMetadataCoverArt

...

AVMetadataIdentifierQuickTimeMetadata**Title**

AVMetadataIdentifierQuickTimeMetadataCopyright

AVMetadataIdentifierQuickTimeMetadataLocationISO6709

...

AVMetadataIdentifier3GPUUserData**Title**

AVMetadataIdentifier3GPUUserDataCopyright

AVMetadataIdentifier3GPUUserDataAuthor

...

AVMetadataCommonIdentifier**Title**

Metadata Identifiers

Common identifiers

AVMetadataIdentifieriTunesMetadataSongName

AVMetadataIdentifieriTunesMetadataCopyright

AVMetadataIdentifieriTunesMetadataCoverArt

...

AVMetadataIdentifierQuickTimeMetadataTitle

AVMetadataCommonIdentifierTitle

AVMetadataIdentifierQuickTimeMetadataCopyright

AVMetadataIdentifierQuickTimeMetadataLocationISO6709

...

AVMetadataIdentifier3GPUUserDataTitle

AVMetadataIdentifier3GPUUserDataCopyright

AVMetadataIdentifier3GPUUserDataAuthor

...

Metadata Identifiers

Common identifiers

AVMetadataIdentifieriTunesMetadataSongName

AVMetadataIdentifieriTunesMetadataCopyright

AVMetadataIdentifieriTunesMetadataCoverArt

...

AVMetadataIdentifierQuickTimeMetadataTitle

AVMetadataIdentifierQuickTimeMetadataCopyright

AVMetadataIdentifierQuickTimeMetadataLocationISO6709

...

AVMetadataIdentifier3GPUUserDataTitle

AVMetadataIdentifier3GPUUserDataCopyright

AVMetadataIdentifier3GPUUserDataAuthor

...

AVMetadataCommonIdentifierTitle

AVMetadataCommonIdentifierCopyrights

Common Metadata

Read asset title

Get all “common” metadata:

```
commonArray = [asset commonMetadata];
```

Get title(s):

```
titleArray = [AVMetadataItem metadataItemsFromArray: commonArray  
              filteredByIdentifier: AVMetadataCommonIdentifierTitle];
```

Also `AVAssetTrack.commonMetadata`

Authoring

Asset metadata

```
@property (nonatomic, copy) NSArray *metadata; // NSArray of AVMetadataItem
```

```
AVAssetExportSession
```

```
AVAssetWriter
```

```
AVCaptureMovieFileOutput
```

```
AVCaptureAudioFileOutput
```

Authoring

Track metadata

```
@property (nonatomic, copy) NSArray *metadata; // NSArray of AVMetadataItem  
AVAssetWriterInput
```

AVMutableMetadataItem

AVMutableMetadataItem

```
AVMutableMetadataItem *SDH1 = [AVMutableMetadataItem metadataItem];  
AVMutableMetadataItem *SDH2 = [AVMutableMetadataItem metadataItem];
```

AVMutableMetadataItem

```
AVMutableMetadataItem *SDH1 = [AVMutableMetadataItem metadataItem];  
AVMutableMetadataItem *SDH2 = [AVMutableMetadataItem metadataItem];
```

```
SDH1.identifier = AVMetadataIdentifierQuickTimeUserDataTaggedCharacteristic;  
SDH1.value = AVMediaCharacteristicTranscribesSpokenDialogForAccessibility;
```

```
SDH2.identifier = AVMetadataIdentifierQuickTimeUserDataTaggedCharacteristic;  
SDH2.value = AVMediaCharacteristicDescribesMusicAndSoundForAccessibility;
```

AVMutableMetadataItem

```
AVMutableMetadataItem *SDH1 = [AVMutableMetadataItem metadataItem];  
AVMutableMetadataItem *SDH2 = [AVMutableMetadataItem metadataItem];
```

```
SDH1.identifier = AVMetadataIdentifierQuickTimeUserDataTaggedCharacteristic;  
SDH1.value = AVMediaCharacteristicTranscribesSpokenDialogForAccessibility;
```

```
SDH2.identifier = AVMetadataIdentifierQuickTimeUserDataTaggedCharacteristic;  
SDH2.value = AVMediaCharacteristicDescribesMusicAndSoundForAccessibility;
```

```
AVAssetWriterInput *subtitleInput = ...;  
subtitleInput.metadata = @[ SDH1, SDH2 ];
```

Authoring

AVAssetExportSession

Passes through metadata from source asset

Value set on `metadata` property is written instead of metadata from source asset

Authoring

AVAssetExportSession

Passes through metadata from source asset

Value set on **metadata** property is written instead of metadata from source asset

Augment or modify

```
AVAssetExportSession *exportSession = ...
```

```
AVMetadataItem *myCustomMetadata = ...
```

```
NSMutableArray *metadata = [exportSession.asset.metadata mutableCopy];
```

```
[metadata addObject: myCustomMetadata];
```

```
exportSession.metadata = metadata;
```

Authoring

HTTP live streaming



New tag—EXT-X-SESSION-DATA

- DATA-ID (reverse-DNS)
- URI or VALUE
- LANGUAGE

```
#EXT-X-SESSION-DATA:DATA-ID="com.example.title",LANGUAGE="en",VALUE="This is an example"
```

```
#EXT-X-SESSION-DATA:DATA-ID="com.example.title",LANGAUGE="sp",VALUE="Este es un ejemplo"
```

More Information

Sample code: `avmetadataeditor`

HTTP Live Streaming: <https://developer.apple.com/streaming/>

Metadata in AVFoundation

Timed Metadata

Privacy

Authoring Best Practices

Chapters



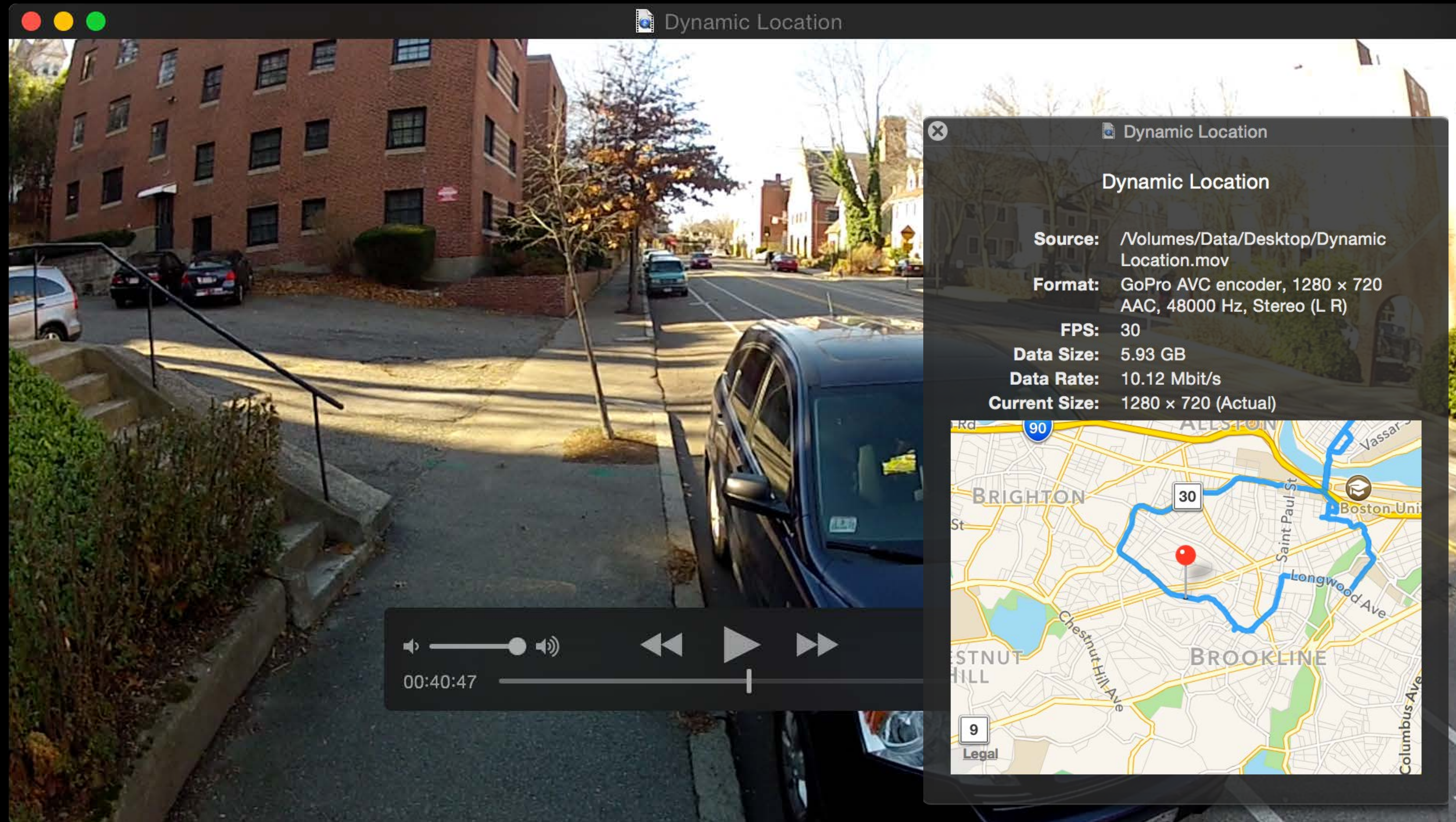
The image shows a video player window titled "Interviews.mov". The main video area displays a young man with short brown hair, wearing a light blue button-down shirt, smiling broadly in an outdoor setting with a green field and a blue sky with clouds. On the right side of the player, a chapter list is overlaid, listing the following chapters and their start times:

Chapter Name	Start Time
Main Titles	00:00
Flight	00:06
Invisibility	00:26
Teleportation	00:50
End Titles	01:06

The "Teleportation" chapter is currently selected, indicated by a white play button icon to its left. Below the chapter list, there is a small thumbnail of the man smiling, and a list of credits including "Interviews" and several names.

Dynamic Location

NEW



The image shows a video player window titled "Dynamic Location". The main video frame displays a street scene with a brick building on the left, a sidewalk, and a blue car in the foreground. A playback control bar at the bottom shows a progress indicator at 00:40:47 and standard navigation buttons. A semi-transparent metadata overlay is positioned on the right side of the video frame. The overlay contains the following information:

- Dynamic Location**
- Source:** /Volumes/Data/Desktop/Dynamic Location.mov
- Format:** GoPro AVC encoder, 1280 x 720 AAC, 48000 Hz, Stereo (L R)
- FPS:** 30
- Data Size:** 5.93 GB
- Data Rate:** 10.12 Mbit/s
- Current Size:** 1280 x 720 (Actual)

Below the metadata, there is a map showing the location in Brighton and Brookline, with a red pin indicating the current position. The map includes labels for streets like Saint Paul St, Longwood Ave, and Chestnut Hill Ave, and landmarks like Boston Uni.

Demo

Timed location

Shalini Sahoo

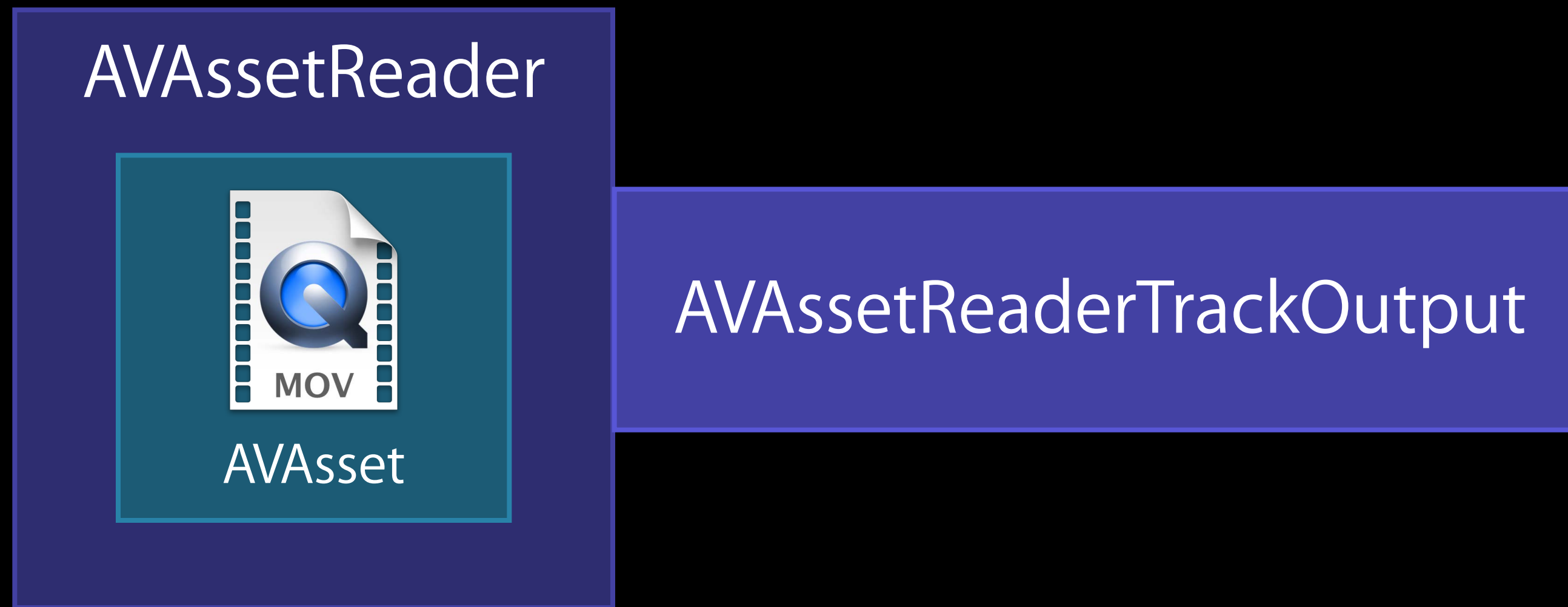
AVFoundation Engineer

Drawing Path on Map

AVAssetReader



Drawing Path on Map

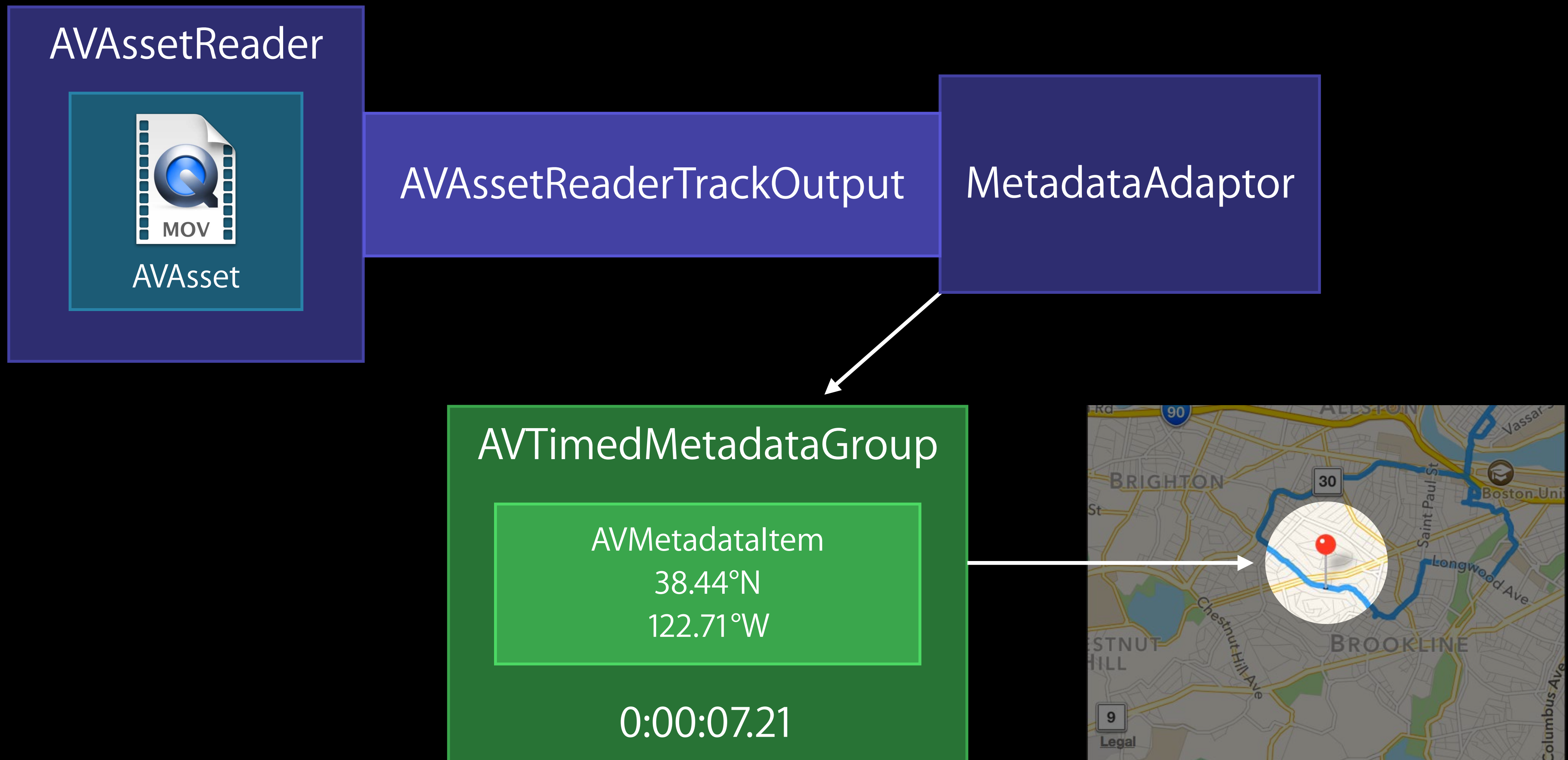


Drawing Path on Map



Drawing Path on Map

NEW



AVTimedMetadataGroup

<AVFoundation/AVTimedMetadataGroup.h>

AVTimedMetadataGroup

items: NSArray (AVMetadataItem)

timeRange: CMTimeRange

Drawing Path on Map



<AVFoundation/AVAssetReaderOutput.h>

Setup

```
AVAssetTrack *GPSTrack = ...;
trackOutput = [[AVAssetReaderTrackOutput alloc] initWithTrack: GPSTrack
                                                         outputSettings: nil];
metadataAdaptor = [[AVAssetReaderOutputMetadataAdaptor alloc]
                  initWithAssetReaderTrackOutput: trackOutput];
```

Read data

```
AVTimedMetadataGroup *metadataGroup = nil;
do {
    metadataGroup = [metadataAdaptor nextTimedMetadataGroup];
    // Draw next segment of path...
} while (metadataGroup != nil);
```

Drawing Path on Map



<AVFoundation/AVAssetReaderOutput.h>

Setup

```
AVAssetTrack *GPSTrack = ...;
trackOutput = [[AVAssetReaderTrackOutput alloc] initWithTrack: GPSTrack
                                                         outputSettings: nil];
metadataAdaptor = [[AVAssetReaderOutputMetadataAdaptor alloc]
                  initWithAssetReaderTrackOutput: trackOutput];
```

Read data

```
AVTimedMetadataGroup *metadataGroup = nil;
do {
    metadataGroup = [metadataAdaptor nextTimedMetadataGroup];
    // Draw next segment of path...
} while (metadataGroup != nil);
```

Drawing Path on Map



<AVFoundation/AVAssetReaderOutput.h>

Setup

```
AVAssetTrack *GPSTrack = ...;
trackOutput = [[AVAssetReaderTrackOutput alloc] initWithTrack: GPSTrack
                                                         outputSettings: nil];
metadataAdaptor = [[AVAssetReaderOutputMetadataAdaptor alloc]
                  initWithAssetReaderTrackOutput: trackOutput];
```

Read data

```
AVTimedMetadataGroup *metadataGroup = nil;
do {
    metadataGroup = [metadataAdaptor nextTimedMetadataGroup];
    // Draw next segment of path...
} while (metadataGroup != nil);
```

Drawing Path on Map

Finding the right track

A white rounded square containing the word "NEW" in a colorful, outlined font.

```
NSArray *metadataTracks = [myAsset tracksWithMediaType: AVMediaTypeMetadata];

for (AVAssetTrack *candidateTrack in metadataTracks) {
    for (id formatDescription in [candidateTrack formatDescriptions]) {
        CFArrayRef identifiers =
            CMMetadataFormatDescriptionGetIdentifiers( formatDescription );

        if ([identifiers containsObject:
            AVMetadataIdentifierQuickTimeMetadataLocationIS06709) {
            GPSTrack = candidateTrack;
            break;
        }
    }
}
```


Drawing Path on Map

Finding the right track



```
NSArray *metadataTracks = [myAsset tracksWithMediaType: AVMediaTypeMetadata];

for (AVAssetTrack *candidateTrack in metadataTracks) {
    for (id formatDescription in [candidateTrack formatDescriptions]) {
        CFArrayRef identifiers =
            CMMetadataFormatDescriptionGetIdentifiers( formatDescription );

        if ([identifiers containsObject:
            AVMetadataIdentifierQuickTimeMetadataLocationIS06709) {
            GPSTrack = candidateTrack;
            break;
        }
    }
}
```

Drawing Path on Map

Finding the right track

A white rounded square containing the word "NEW" in a colorful, outlined font.

```
NSArray *metadataTracks = [myAsset tracksWithMediaType: AVMediaTypeMetadata];

for (AVAssetTrack *candidateTrack in metadataTracks) {
    for (id formatDescription in [candidateTrack formatDescriptions]) {
        CFArrayRef identifiers =
            CMMetadataFormatDescriptionGetIdentifiers( formatDescription );

        if ([identifiers containsObject:
            AVMetadataIdentifierQuickTimeMetadataLocationIS06709) {
            GPSTrack = candidateTrack;
            break;
        }
    }
}
```

Drawing Path on Map

Finding the right track

A white rounded square containing the word "NEW" in a colorful, outlined font.

```
NSArray *metadataTracks = [myAsset tracksWithMediaType: AVMediaTypeMetadata];

for (AVAssetTrack *candidateTrack in metadataTracks) {
    for (id formatDescription in [candidateTrack formatDescriptions]) {
        CFArrayRef identifiers =
            CMMetadataFormatDescriptionGetIdentifiers( formatDescription );

        if ([identifiers containsObject:
            AVMetadataIdentifierQuickTimeMetadataLocationIS06709) {
            GPSTrack = candidateTrack;
            break;
        }
    }
}
```

Updating Location During Playback

AVPlayerItem



Updating Location During Playback

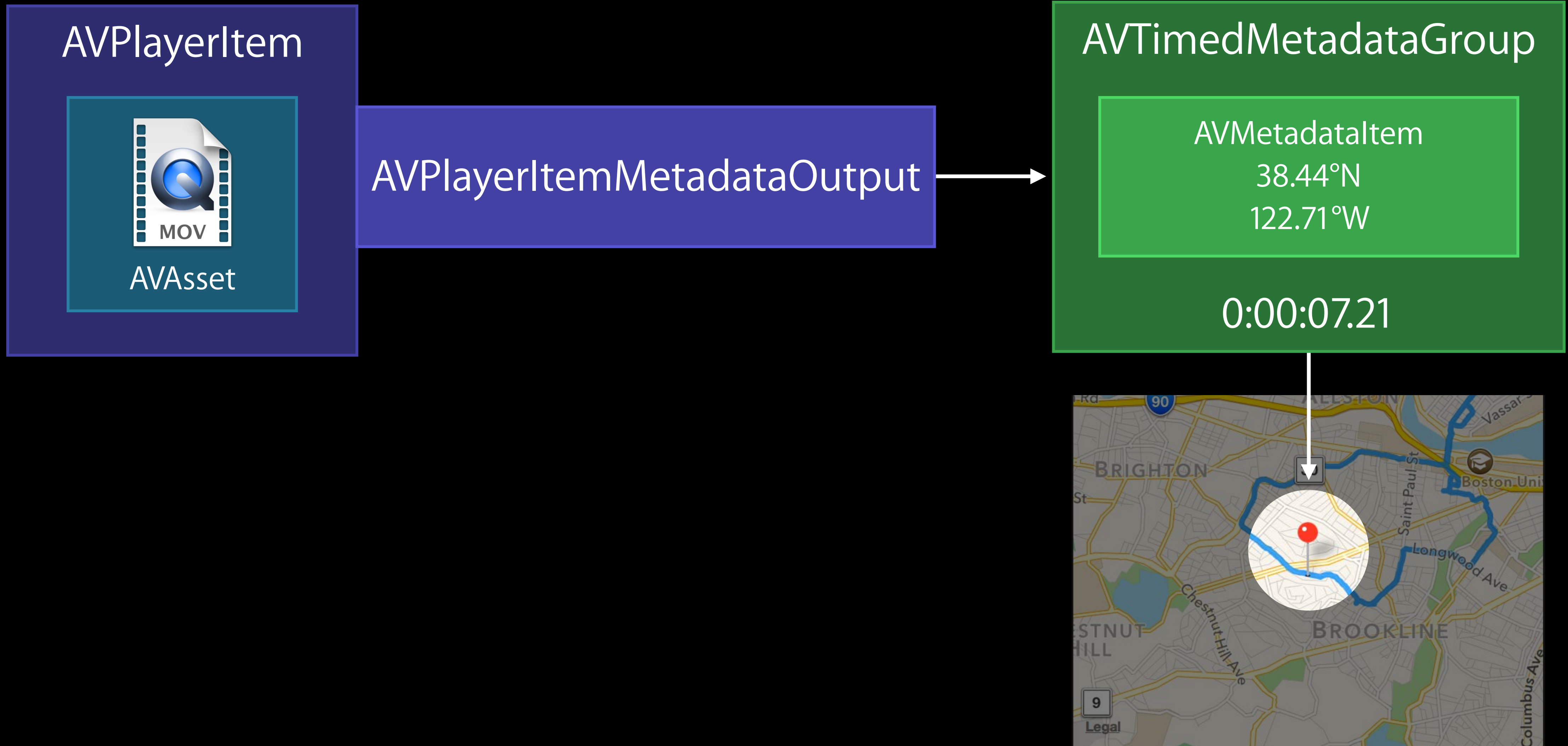


AVPlayerItem



AVPlayerItemMetadataOutput

Updating Location During Playback



Updating Location During Playback

AVPlayerItemMetadataOutput



```
<AVFoundation/AVPlayerItemOutput.h>
```

```
output = [[AVPlayerItemMetadataOutput alloc] initWithIdentifiers:
           @[ AVMetadataIdentifierQuickTimeMetadataLocationIS06709 ]];
```

```
id <AVPlayerItemMetadataOutputPushDelegate> myDelegate = ...;
[output setDelegate: myDelegate queue: mySerialDispatchQueue];
```

```
AVPlayerItem *playerItem = ...;
[playerItem addOutput: output];
```

```
AVPlayer *player = [[AVPlayer alloc] init];
[player replaceCurrentItemWithPlayerItem: item]; // after config is done
```

Updating Location During Playback

AVPlayerItemMetadataOutput



```
<AVFoundation/AVPlayerItemOutput.h>
```

```
output = [[AVPlayerItemMetadataOutput alloc] initWithIdentifiers:
          @[ AVMetadataIdentifierQuickTimeMetadataLocationIS06709 ]];
```

```
id <AVPlayerItemMetadataOutputPushDelegate> myDelegate = ...;
[output setDelegate: myDelegate queue: mySerialDispatchQueue];
```

```
AVPlayerItem *playerItem = ...;
[playerItem addOutput: output];
```

```
AVPlayer *player = [[AVPlayer alloc] init];
[player replaceCurrentItemWithPlayerItem: item]; // after config is done
```


Updating Location During Playback

AVPlayerItemMetadataOutput



```
<AVFoundation/AVPlayerItemOutput.h>
```

```
output = [[AVPlayerItemMetadataOutput alloc] initWithIdentifiers:
          @[ AVMetadataIdentifierQuickTimeMetadataLocationIS06709 ]];
```

```
id <AVPlayerItemMetadataOutputPushDelegate> myDelegate = ...;
[output setDelegate: myDelegate queue: mySerialDispatchQueue];
```

```
AVPlayerItem *playerItem = ...;
[playerItem addOutput: output];
```

```
AVPlayer *player = [[AVPlayer alloc] init];
[player replaceCurrentItemWithPlayerItem: item]; // after config is done
```

Updating Location During Playback

AVPlayerItemMetadataOutput



```
<AVFoundation/AVPlayerItemOutput.h>
```

```
output = [[AVPlayerItemMetadataOutput alloc] initWithIdentifiers:
          @[ AVMetadataIdentifierQuickTimeMetadataLocationIS06709 ]];
```

```
id <AVPlayerItemMetadataOutputPushDelegate> myDelegate = ...;
[output setDelegate: myDelegate queue: mySerialDispatchQueue];
```

```
AVPlayerItem *playerItem = ...;
[playerItem addOutput: output];
```

```
AVPlayer *player = [[AVPlayer alloc] init];
[player replaceCurrentItemWithPlayerItem: item]; // after config is done
```

Updating Location During Playback

NEW

AVPlayerItemMetadataOutputPushDelegate protocol

```
- (void) metadataOutput: (AVPlayerItemMetadataOutput *)output
  didOutputTimedMetadataGroups: (NSArray *)metadataGroups
    fromPlayerItemTrack: (AVPlayerItemTrack *)track
{
    AVMetadataItem *item = [[metadataGroups firstObject].items firstObject];
    NSArray *itemKeys = @[ @"value", @"dataType" ];
    [item loadValuesAsynchronouslyForKeys: itemKeys completionHandler:
    ^{
        if ([item.dataType isEqualToString: (__bridge NSString *)
            kCMMetadataDataType_QuickTimeMetadataLocation_ISO6709]) {
            // Grab location from ISO6709 string (item.stringValue)
        }
        // dispatch to main thread, update UI with location
    }
    }
```

Updating Location During Playback

NEW

AVPlayerItemMetadataOutputPushDelegate protocol

```
- (void) metadataOutput: (AVPlayerItemMetadataOutput *)output
  didOutputTimedMetadataGroups: (NSArray *)metadataGroups
    fromPlayerItemTrack: (AVPlayerItemTrack *)track
{
    AVMetadataItem *item = [[metadataGroups firstObject].items firstObject];
    NSArray *itemKeys = @[ @"value", @"dataType" ];
    [item loadValuesAsynchronouslyForKeys: itemKeys completionHandler:
    ^{
        if ([item.dataType isEqualToString: (__bridge NSString *)
            kCMMDetadataDataType_QuickTimeMetadataLocation_ISO6709]) {
            // Grab location from ISO6709 string (item.stringValue)
        }
        // dispatch to main thread, update UI with location
    }
    }
```


Updating Location During Playback

NEW

AVPlayerItemMetadataOutputPushDelegate protocol

```
- (void) metadataOutput: (AVPlayerItemMetadataOutput *)output
  didOutputTimedMetadataGroups: (NSArray *)metadataGroups
    fromPlayerItemTrack: (AVPlayerItemTrack *)track
{
    AVMetadataItem *item = [[metadataGroups firstObject].items firstObject];
    NSArray *itemKeys = @[ @"value", @"dataType" ];
    [item loadValuesAsynchronouslyForKeys: itemKeys completionHandler:
    ^{
        if ([item.dataType isEqualToString: (__bridge NSString *)
            kCMMDetadataDataType_QuickTimeMetadataLocation_ISO6709]) {
            // Grab location from ISO6709 string (item.stringValue)
        }
        // dispatch to main thread, update UI with location
    }
}
```

Updating Location During Playback

NEW

AVPlayerItemMetadataOutputPushDelegate protocol

```
- (void) metadataOutput: (AVPlayerItemMetadataOutput *)output
  didOutputTimedMetadataGroups: (NSArray *)metadataGroups
    fromPlayerItemTrack: (AVPlayerItemTrack *)track
{
    AVMetadataItem *item = [[metadataGroups firstObject].items firstObject];
    NSArray *itemKeys = @[ @"value", @"dataType" ];
    [item loadValuesAsynchronouslyForKeys: itemKeys completionHandler:
    ^{
        if ([item.dataType isEqualToString: (__bridge NSString *)
            kCMMDetadataDataType_QuickTimeMetadataLocation_ISO6709]) {
            // Grab location from ISO6709 string (item.stringValue)
        }
        // dispatch to main thread, update UI with location
    }
    }
```

Updating Location During Playback

NEW

AVPlayerItemMetadataOutputPushDelegate protocol

```
- (void) metadataOutput: (AVPlayerItemMetadataOutput *)output
    didOutputTimedMetadataGroups: (NSArray *)metadataGroups
        fromPlayerItemTrack: (AVPlayerItemTrack *)track
{
    AVMetadataItem *item = [[metadataGroups firstObject].items firstObject];
    NSArray *itemKeys = @[ @"value", @"dataType" ];
    [item loadValuesAsynchronouslyForKeys: itemKeys completionHandler:
    ^{
        if ([item.dataType isEqualToString: (__bridge NSString *)
            kCMMetadataDataType_QuickTimeMetadataLocation_ISO6709]) {
            // Grab location from ISO6709 string (item.stringValue)
        }
        // dispatch to main thread, update UI with location
    }
    }
```

Evolution of Playback Interfaces

`AVPlayerItemMetadataOutput` replaces `AVPlayerItem.timedMetadata`

Existing clients of `timedMetadata` property

- Consider adopting `AVPlayerItemMetadataOutput`
- Check identifiers for HLS content

Reading Chapters

<AVFoundation/AVAsset.h>, AVAssetChapterInspection category

– (NSArray *)chapterMetadataGroupsBestMatchingPreferredLanguages: (NSArray *)preferredLanguages;

AVMetadataIdentifierQuickTimeUserDataChapter

QuickTime Movie, M4V

Reading Chapters



<AVFoundation/AVAsset.h>, AVAssetChapterInspection category

– (NSArray *)chapterMetadataGroupsBestMatchingPreferredLanguages: (NSArray *)preferredLanguages;

AVMetadataIdentifierQuickTimeUserDataChapter

QuickTime Movie, M4V

HTTP Live Streaming (Video on Demand only)

MP3

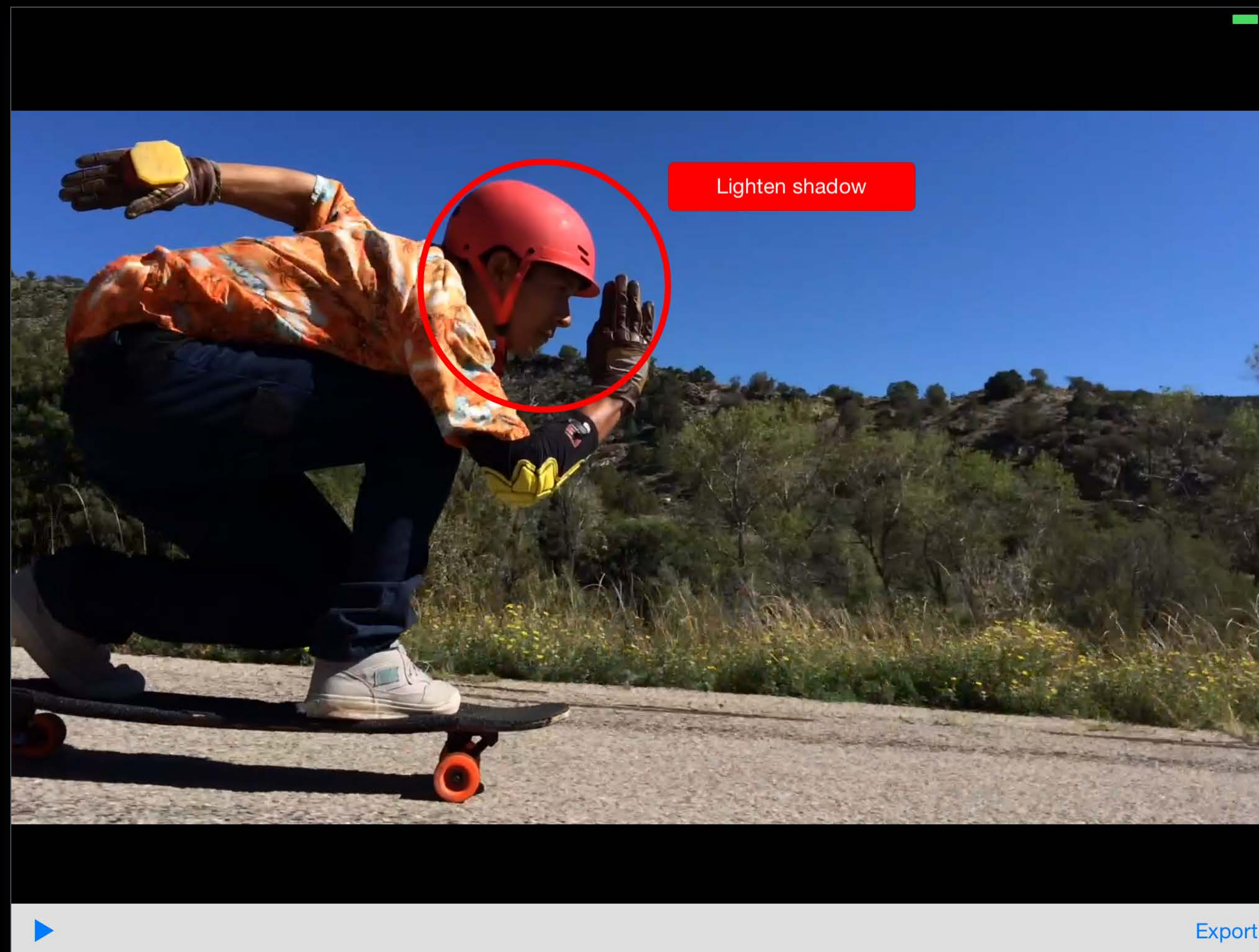
More Information

Sample code: *AVLocationPlayer*

More about *AVAssetReader*: WWDC 2011 — Working with Media in AVFoundation

Your Metadata

NEW



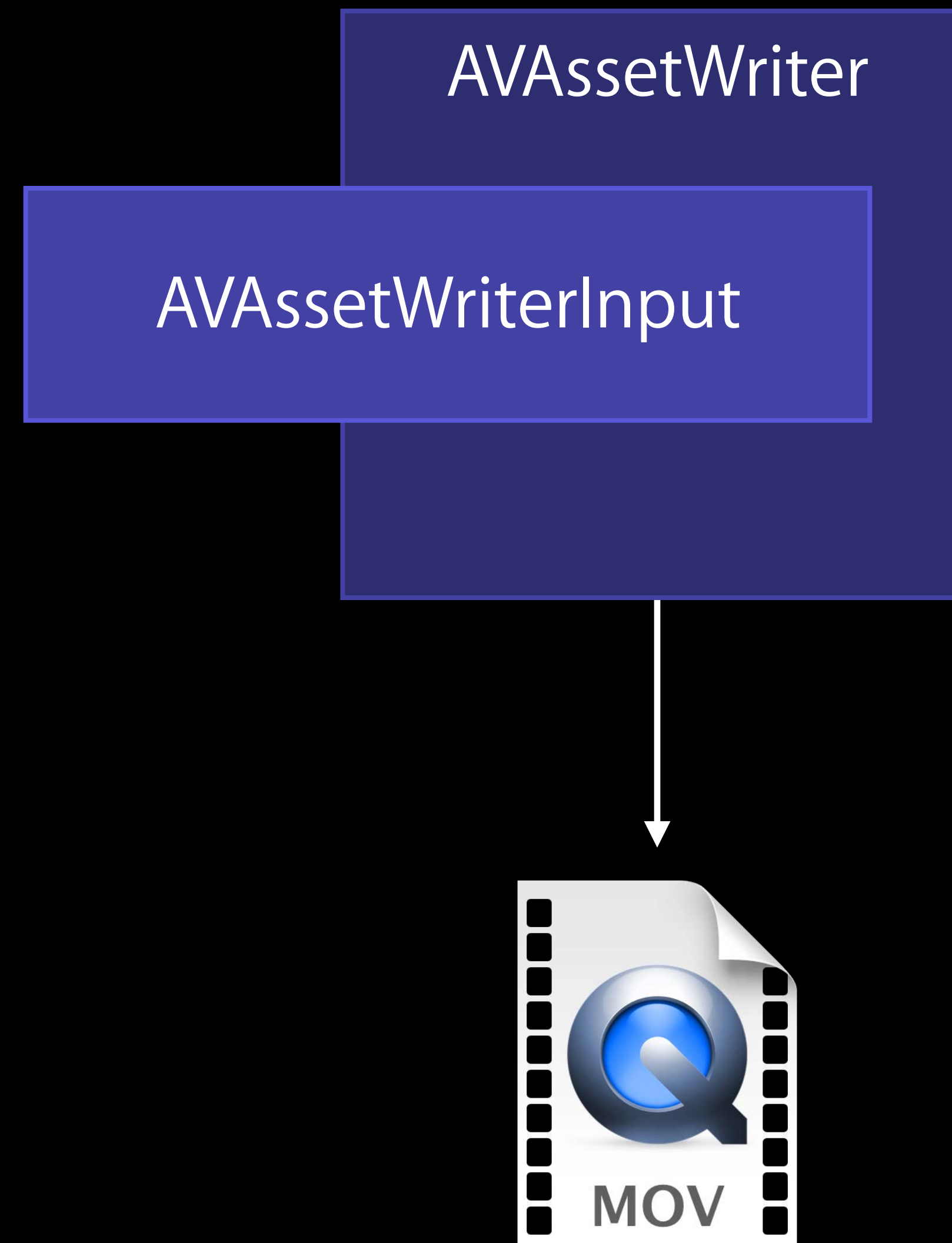
Demo

AVTimedAnnotationWriter

Shalini Sahoo

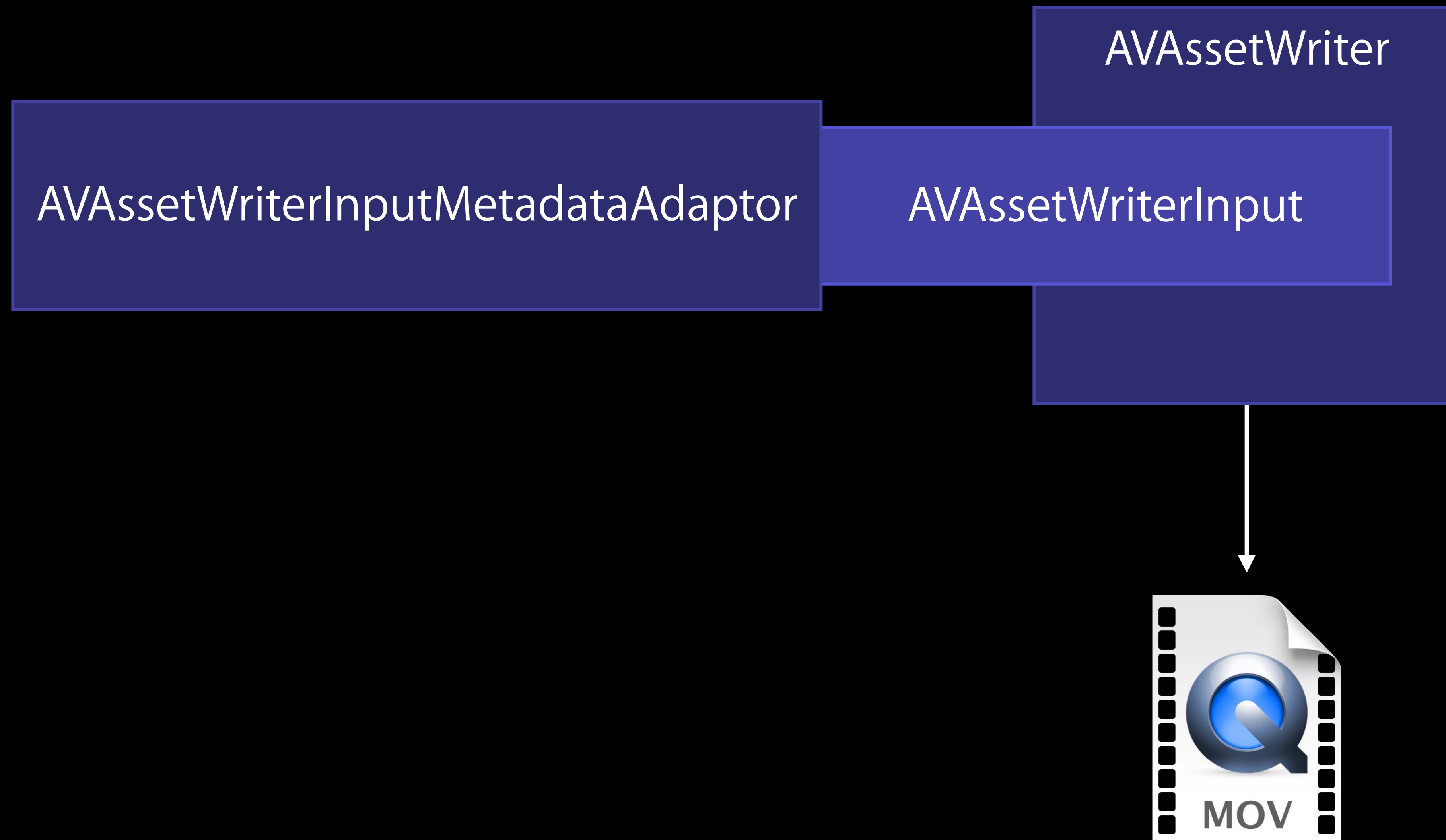
AVFoundation Engineer

Writing Annotations to Movie File



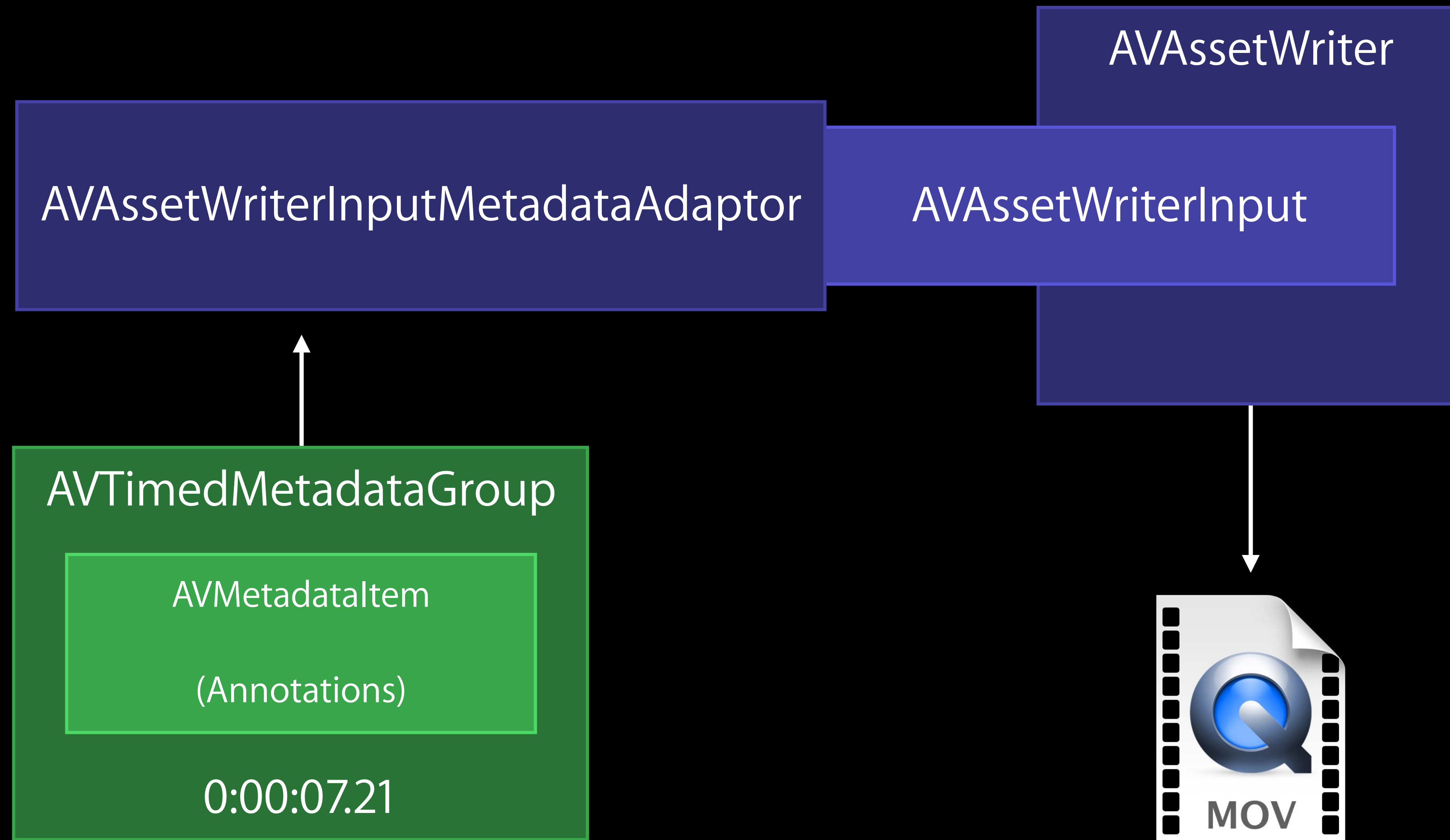
Writing Annotations to Movie File

NEW



Writing Annotations to Movie File

NEW



Writing Annotations to Movie File



<AVFoundation/AVAssetWriterInput.h>

Setup

```
CMFormatDescriptionRef sourceFormat = ...; // required
input = [[AVAssetWriterInput alloc] initWithMediaType:AVMediaTypeMetadata
        outputSettings: nil
        sourceFormatHint: sourceFormat];
```

```
metadataAdaptor = [[AVAssetWriterInputMetadataAdaptor alloc]
        initWithAssetWriterInput: input];
```

Append metadata

```
AVTimedMetadataGroup *metadataGroup = ...;
[metadataAdaptor appendTimedMetadataGroup: metadataGroup];
```

Writing Annotations to Movie File



<AVFoundation/AVAssetWriterInput.h>

Setup

```
CMFormatDescriptionRef sourceFormat = ...; // required
input = [[AVAssetWriterInput alloc] initWithMediaType:AVMediaTypeMetadata
        outputSettings: nil
        sourceFormatHint: sourceFormat];
```

```
metadataAdaptor = [[AVAssetWriterInputMetadataAdaptor alloc]
        initWithAssetWriterInput: input];
```

Append metadata

```
AVTimedMetadataGroup *metadataGroup = ...;
[metadataAdaptor appendTimedMetadataGroup: metadataGroup];
```

Writing Annotations to Movie File



<AVFoundation/AVAssetWriterInput.h>

Setup

```
CMFormatDescriptionRef sourceFormat = ...; // required
input = [[AVAssetWriterInput alloc] initWithMediaType:AVMediaTypeMetadata
        outputSettings: nil
        sourceFormatHint: sourceFormat];
```

```
metadataAdaptor = [[AVAssetWriterInputMetadataAdaptor alloc]
        initWithAssetWriterInput: input];
```

Append metadata

```
AVTimedMetadataGroup *metadataGroup = ...;
[metadataAdaptor appendTimedMetadataGroup: metadataGroup];
```

Writing Annotations to Movie File

Source format

Efficient storage and playback

Writing Annotations to Movie File

Source format

Efficient storage and playback

If from AVAssetReader

```
sourceFormat = [trackOutput.track.formatDescriptions firstObject];
```

Writing Annotations to Movie File



Source format

Efficient storage and playback

If from AVAssetReader

```
sourceFormat = [trackOutput.track.formatDescriptions firstObject];
```

If from another source

```
AVTimedMetadataGroup *metadataGroup = ...; // contains every identifier,  
                                             // data type, and extended  
                                             // language tag to be appended  
sourceFormat = [metadataGroup copyFormatDescription];  
// ...  
CFRelease(sourceFormat);
```

Create Metadata Format Description

<CoreMedia/CMFormatDescription.h>



```
OSStatus CMMetadataFormatDescriptionCreateWithMetadataSpecifications(  
    CFAllocatorRef allocator,  
    CMMetadataFormatType metadataType,  
    CFArrayRef metadataSpecifications,  
    CMMetadataFormatDescriptionRef *outDesc);
```

Create Metadata Format Description



<CoreMedia/CMFormatDescription.h>

```
OSStatus CMMetadataFormatDescriptionCreateWithMetadataSpecifications(  
    CFAllocatorRef allocator,  
    CMMetadataFormatType metadataType,  
    CFArrayRef metadataSpecifications,  
    CMMetadataFormatDescriptionRef *outDesc);
```

kCMMetadataFormatType_Boxed

Create Metadata Format Description



<CoreMedia/CMFormatDescription.h>

```
OSStatus CMMetadataFormatDescriptionCreateWithMetadataSpecifications(  
    CFAllocatorRef allocator,  
    CMMetadataFormatType metadataType,  
    CFArrayRef metadataSpecifications, // CFArray of CFDictionary  
    CMMetadataFormatDescriptionRef *outDesc);
```

Dictionary keys:

```
kCMMetadataFormatDescriptionMetadataSpecificationKey_Identifier  
kCMMetadataFormatDescriptionMetadataSpecificationKey_DataType
```

Optional:

```
kCMMetadataFormatDescriptionMetadataSpecificationKey_ExtendedLanguageTag
```


Writing Annotations to Movie File

Track-specific timed metadata

```
AVAssetWriterInput *annotationMetadataInput = ...;
```

```
AVAssetWriterInput *videoInput = ...;
```

```
[annotationMetadataInput addTrackAssociationWithTrackOfInput: videoInput  
                                type: AVTrackAssociationTypeMetadataReferent];
```

Custom Identifiers

<AVFoundation/AVMetadataItem.h>



Identifier = key space + key

```
+ (NSString *)identifierForKey:(id)key keySpace:(NSString *)keySpace;
```

Custom Identifiers



<AVFoundation/AVMetadataItem.h>

Identifier = key space + key

```
+ (NSString *)identifierForKey:(id)key keySpace:(NSString *)keySpace;
```

Key space must be four characters

```
key = @"com.example.circle.radius";  
keySpace = AVMetadataKeySpaceQuickTimeMetadata; // @"mdta";
```

Custom Identifiers



<AVFoundation/AVMetadataItem.h>

Identifier = key space + key

```
+ (NSString *)identifierForKey:(id)key keySpace:(NSString *)keySpace;
```

Key space must be four characters

```
key = @"com.example.circle.radius";
```

```
keySpace = AVMetadataKeySpaceQuickTimeMetadata; // @"mdta";
```

```
myIdentifier = [AVMetadataItem identifierForKey:key keySpace:keySpace];
```

```
mutableMetadataItem.identifier = myIdentifier;
```

Custom Data Types

<CoreMedia/CMMetadata.h>



UTF8

JPEG

Custom Data Types

<CoreMedia/CMMetadata.h>



UTF8

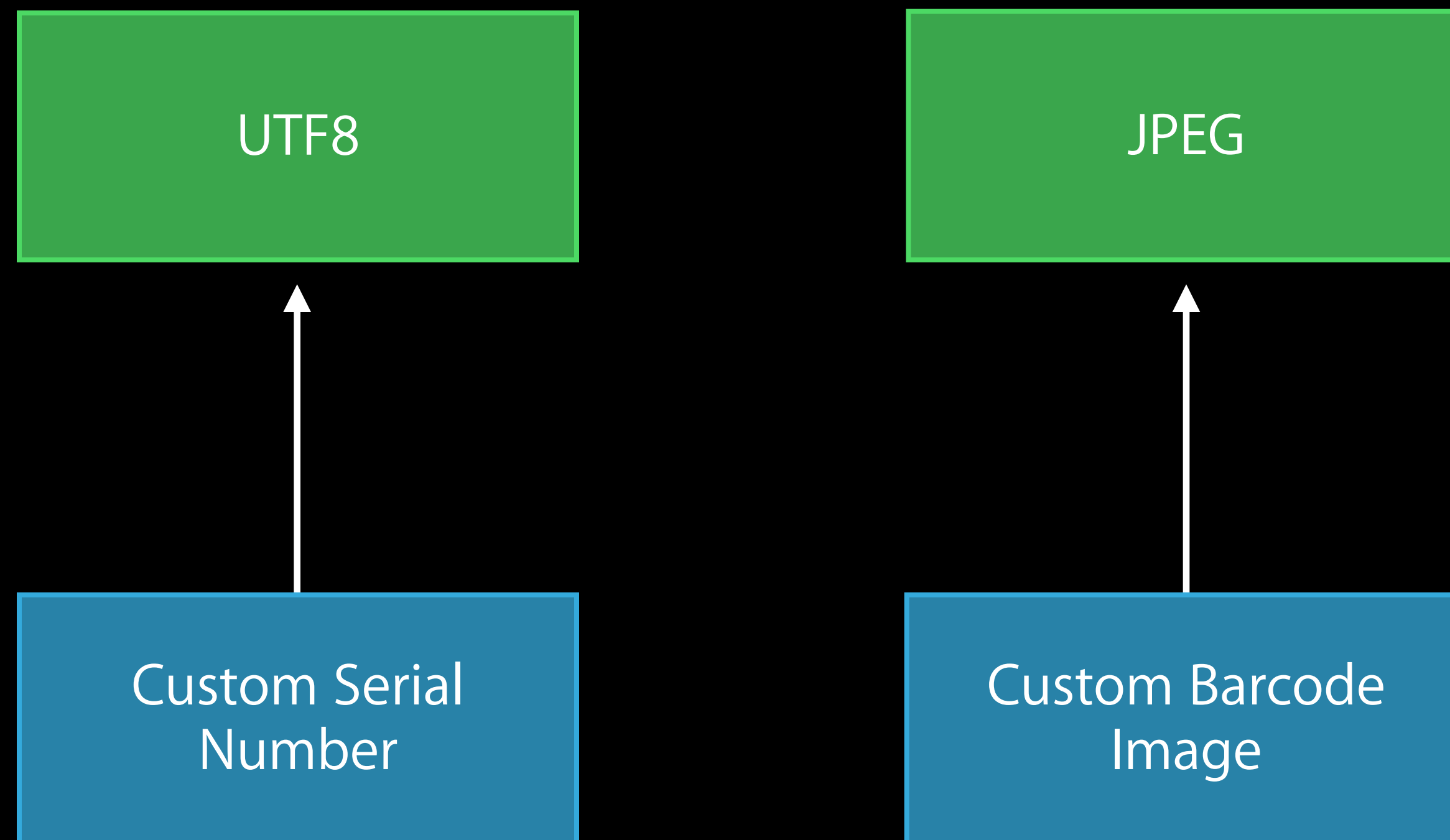
JPEG

Custom Serial
Number

Custom Barcode
Image

Custom Data Types

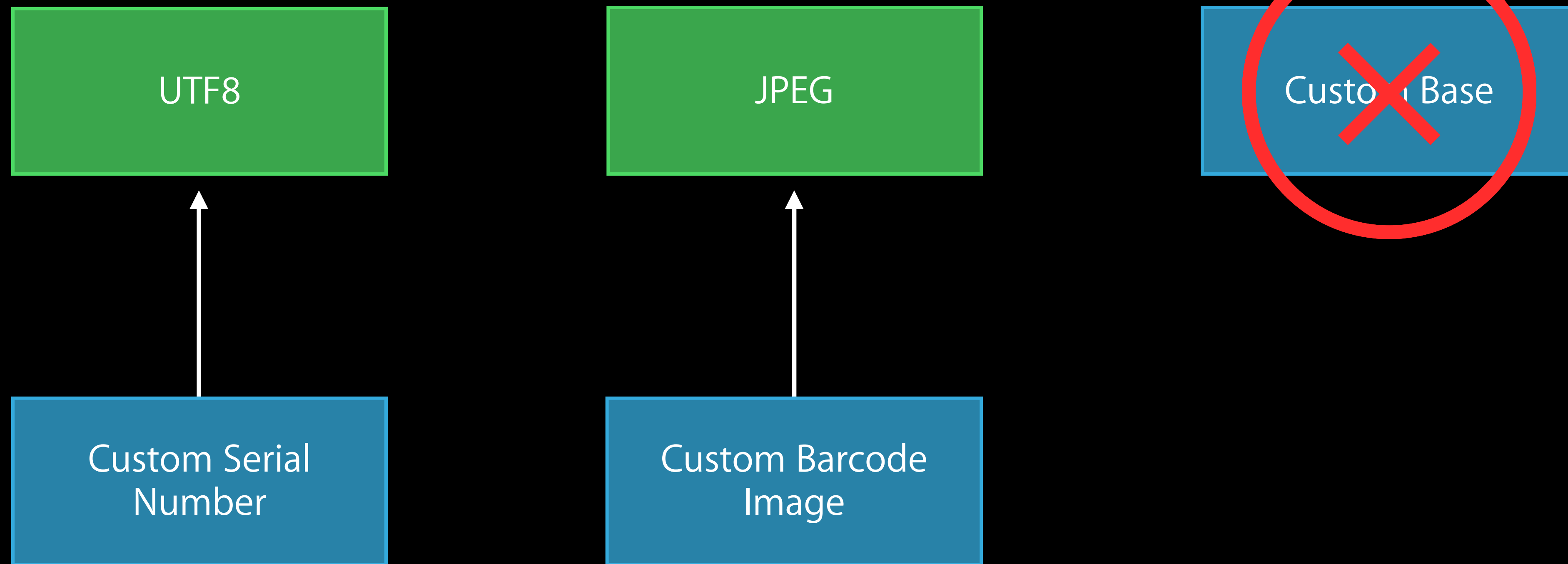
<CoreMedia/CMMetadata.h>



CMMetadataDataTypeRegistryRegisterDataType

Custom Data Types

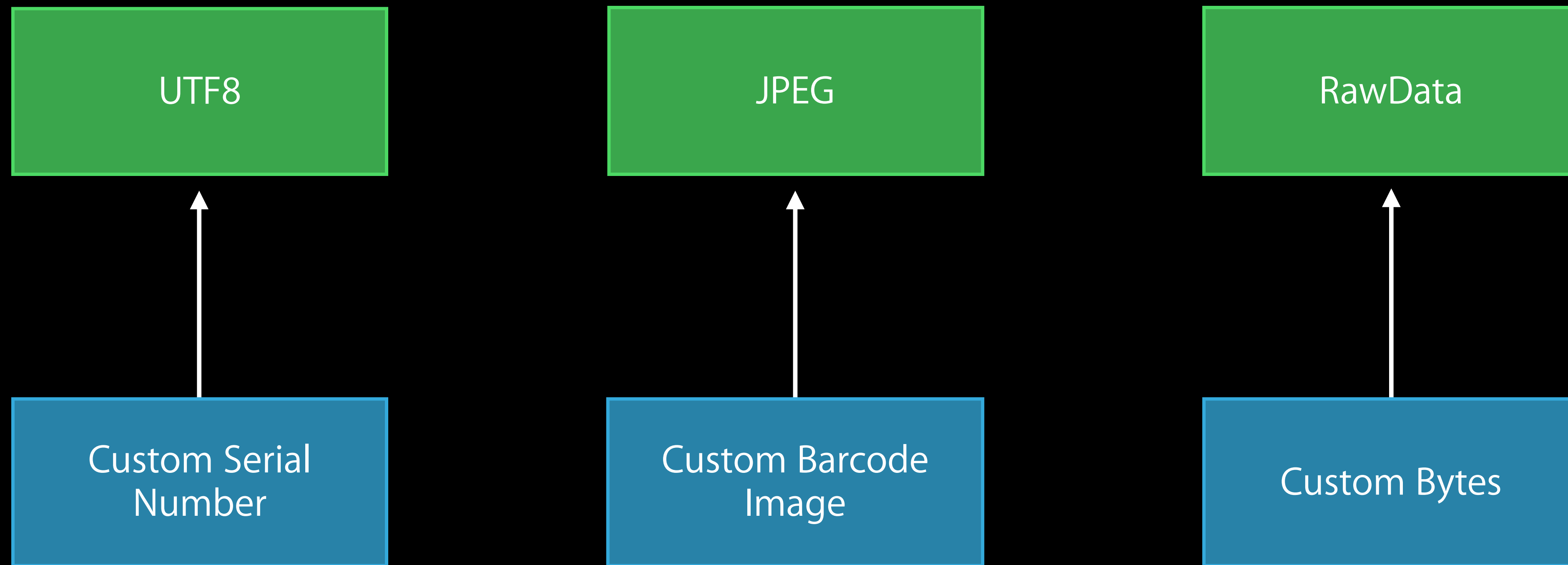
<CoreMedia/CMMetadata.h>



CMMetadataDataTypeRegistryRegisterDataType

Custom Data Types

<CoreMedia/CMMetadata.h>



CMMetadataDataTypeRegistryRegisterDataType

Rules for Using AVAssetWriter

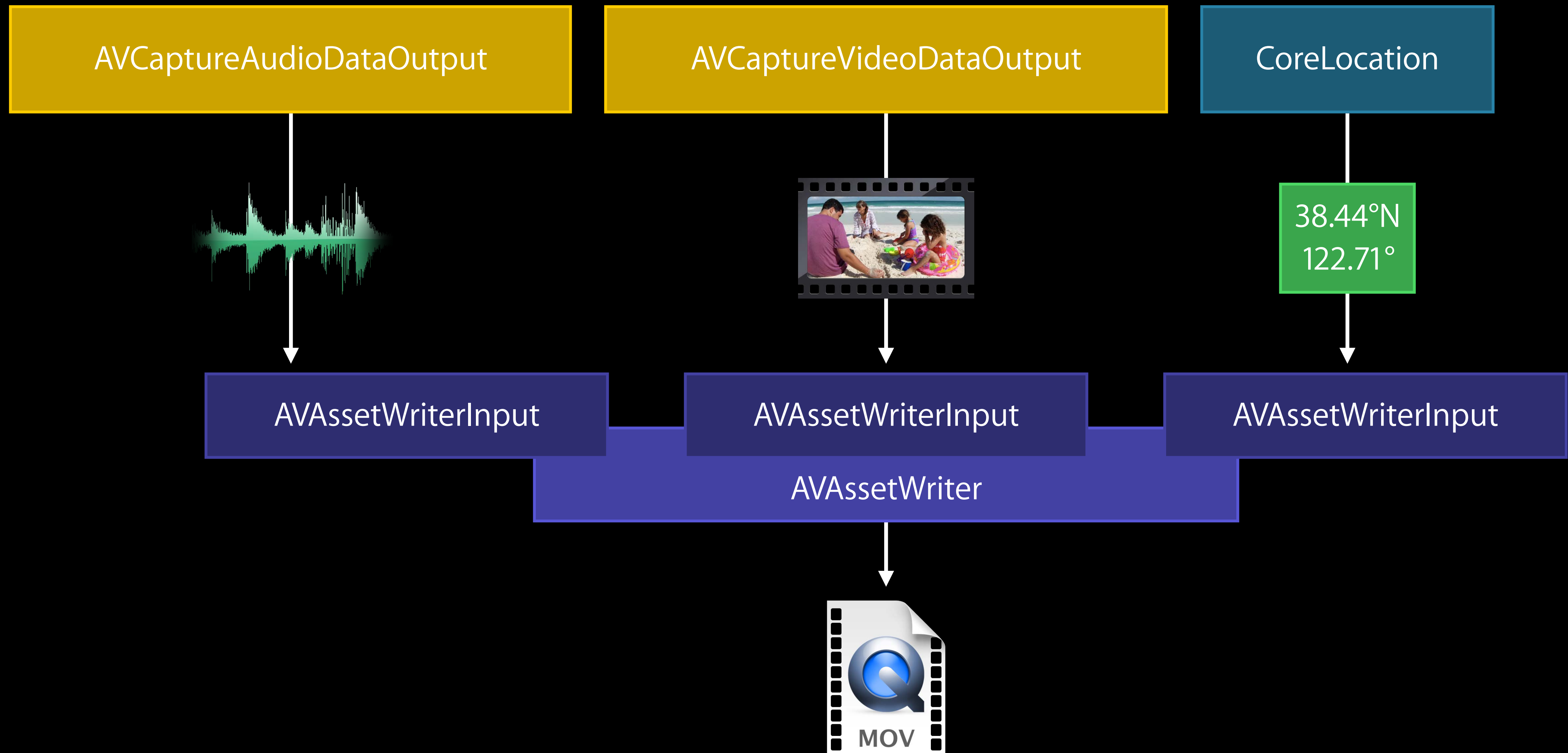
For each appended AVMetadataItem

- Non-nil, valid “identifier”
- Non-nil “dataType”
- Non-nil “value”
- “Value” compatible with “dataType”

Metadata AVAssetWriterInput created with format hint

Format hint is comprehensive

Capturing Dynamic Location



More Information

Sample code: [AVCaptureLocation](#)

Sample code: [AVTimedAnnotationWriter](#)

More about [AVAssetWriter](#): [WWDC 2011 — Working with Media in AVFoundation](#)

Export

AVAssetExportSession

Passes through timed metadata items

Authoring Chapters

HTTP live streaming



Use EXT-X-SESSION-DATA:

```
#EXT-X-SESSION-DATA:DATA-ID="com.apple.hls.chapters",LANGUAGE="en",URI=<chapter JSON>
```

Metadata in AVFoundation

Timed Metadata

Privacy

Authoring Best Practices

Privacy Implications of Metadata

Metadata can contain user-identifiable information

- Location

Movie files are persistent, can be distributed

Stripping User-Identifiable Metadata

AVMetadataItemFilter

<AVFoundation/AVMetadataItem.h>

One privacy-oriented filter:

```
+ (AVMetadataItemFilter *)metadataItemFilterForSharing;
```

Strips out

- User-identifying data (e.g., location)
- Your custom identifiers

Leaves in

- Structural data (e.g., chapters)
- Commercial-related data (e.g., Apple ID)

Using AVMetadataItemFilter

AVAssetWriter

Filter items before adding to AVAssetWriter/Input:

```
NSArray *originalMetadataItems = ...;

filter = [AVMetadataItemFilter metadataItemFilterForSharing];
newItems = [AVMetadataItem metadataItemsFromArray: originalMetadataItems
                                                filteredByMetadataItemFilter: filter];

AVAssetWriter *assetWriter = ...;
assetWriter.metadata = newItems;
```

Using AVMetadataItemFilter

AVAssetExportSession

Set filter on session:

```
AVAssetExportSession *myExportSession = ...;  
filter = [AVMetadataItemFilter metadataItemFilterForSharing];  
myExportSession.metadataItemFilter = filter;
```

Filters both static and timed metadata

Filters metadata from source asset

Does not filter value of AVAssetExportSession.metadata

Export may take more time when filter is used

Metadata in AVFoundation

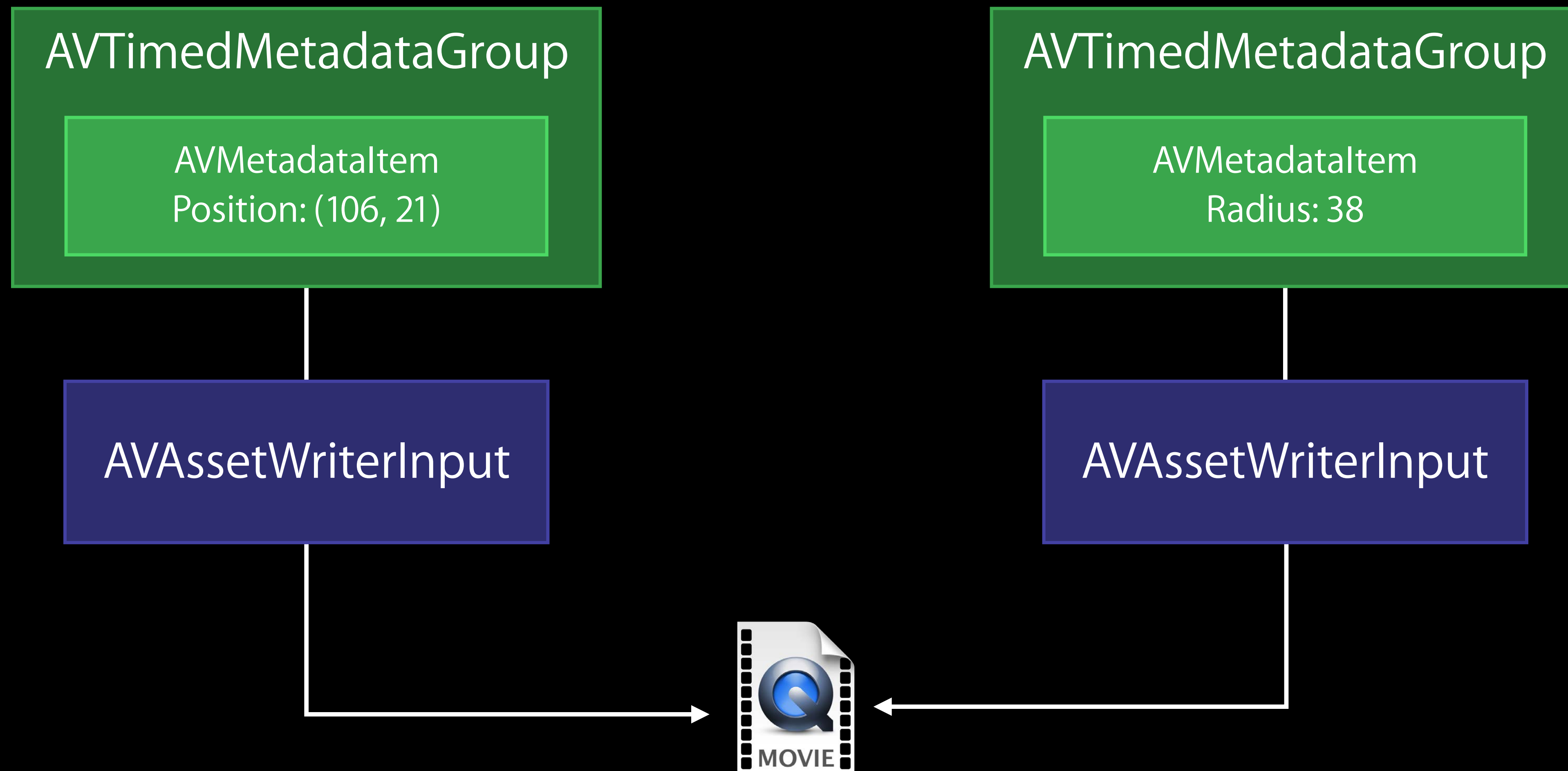
Timed Metadata

Privacy

Authoring Best Practices

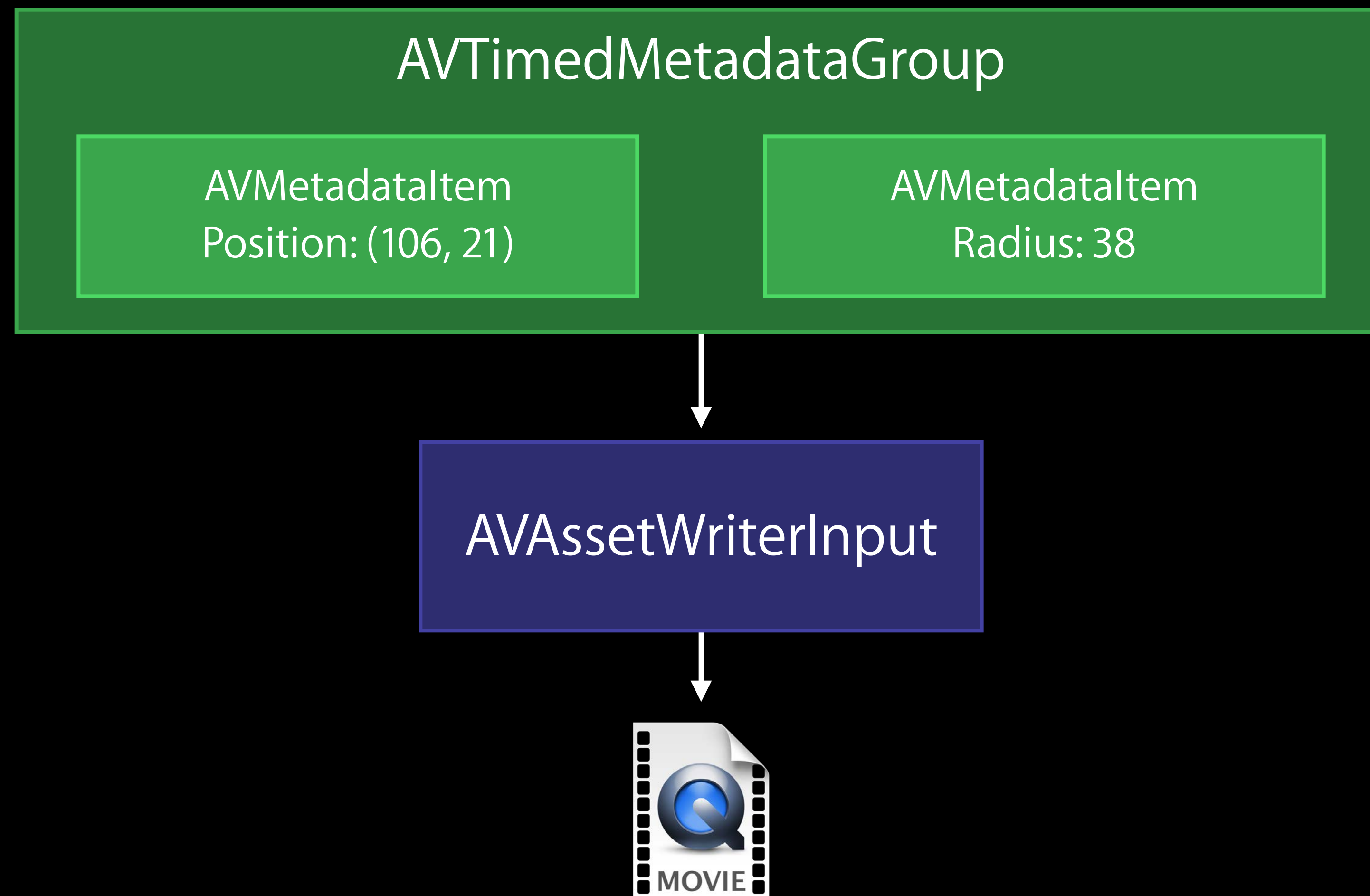
Multiple Streams of Timed Metadata

AVAssetWriter



Multiple Streams of Timed Metadata

AVAssetWriter



Multiple Streams of Timed Metadata

AVAssetWriter

Do combine if

- Used together during playback
- Identical timing

Do not combine if

- Only one has track association

Timed Metadata Group Duration

AVAssetWriter

Group

(1 item)

0:00 - ?



Timed Metadata Group Duration

AVAssetWriter



Timed Metadata Group Duration

AVAssetWriter



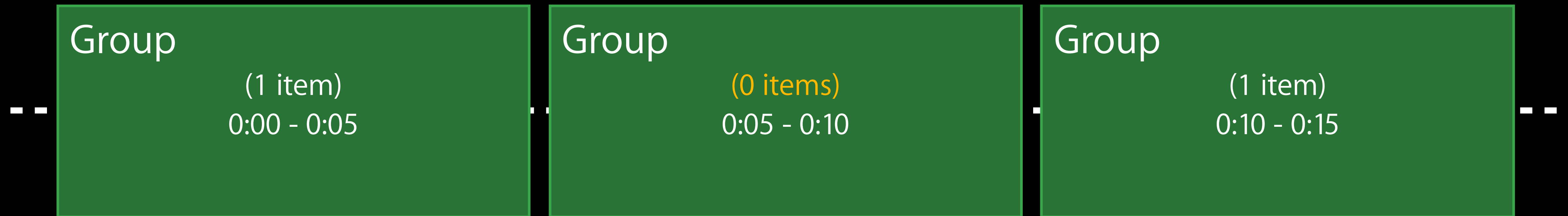
Use "invalid" duration:

```
CMTimeRange timeRange = CMTimeMake( kCMTimeZero, kCMTimeInvalid );
```

```
group = [[AVTimedMetadataGroup alloc] initWithItems: @[ metadataItem ]  
        timeRange: timeRange];
```

Writing Gaps in Metadata

AVAssetWriter



Be explicit:

```
CMTimeRange timeRange = CMTimeMake( CMTimeMake( 5, 1 ), kCMTIME_INVALID );
```

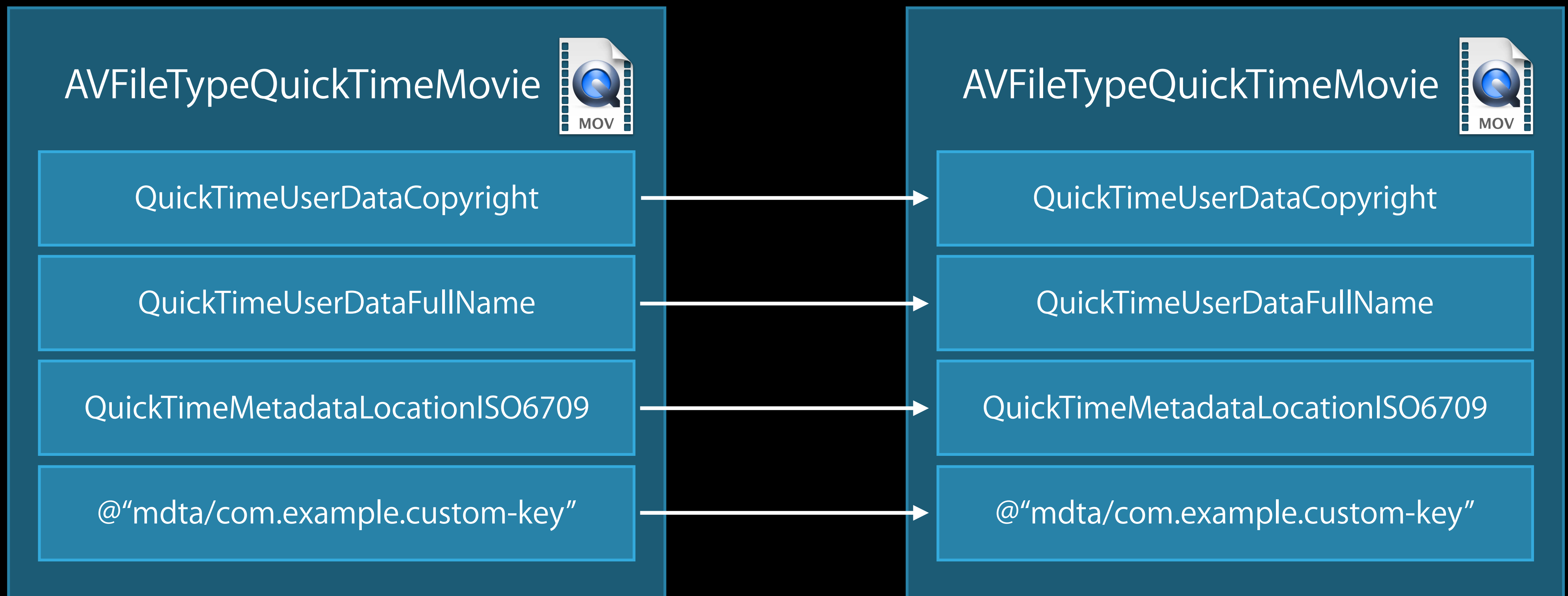
```
emptyGroup = [[AVTimedMetadataGroup alloc] initWithItems: @[]  
              timeRange: timeRange];
```

Output File Type

Same as source

Source

Output

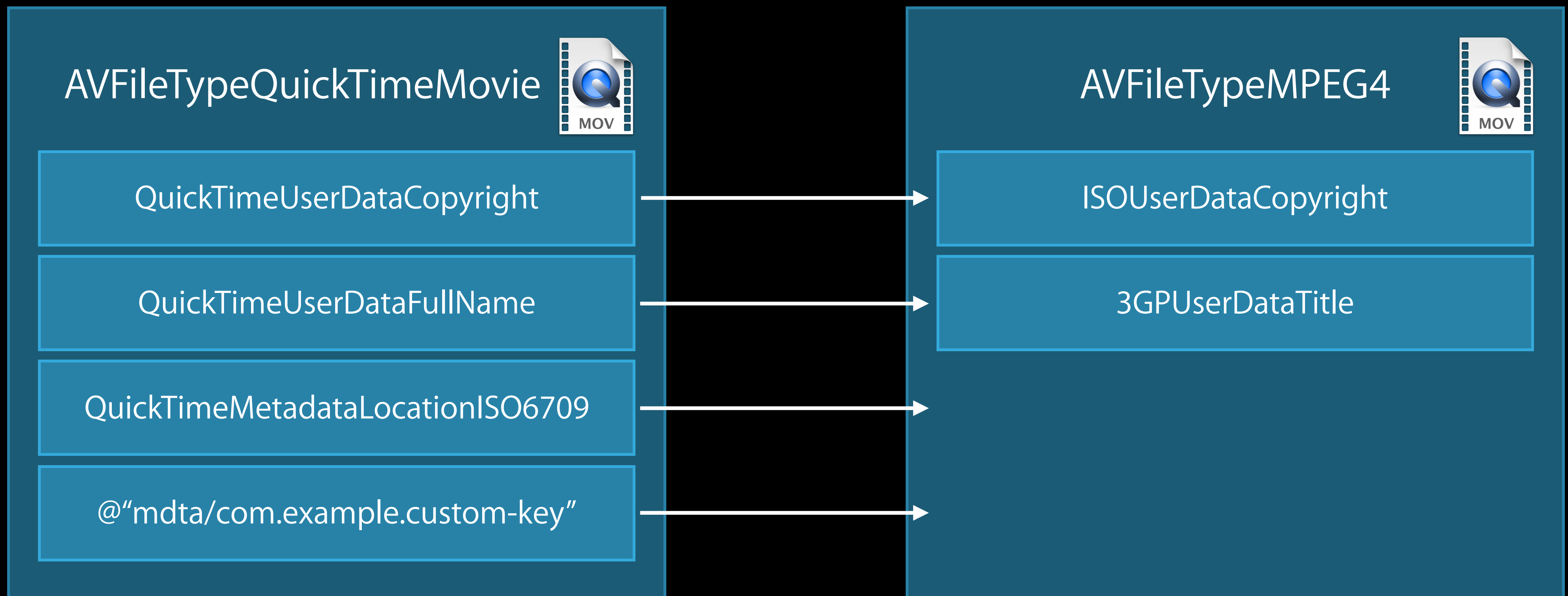


Output File Type

Different from source

Source

Output

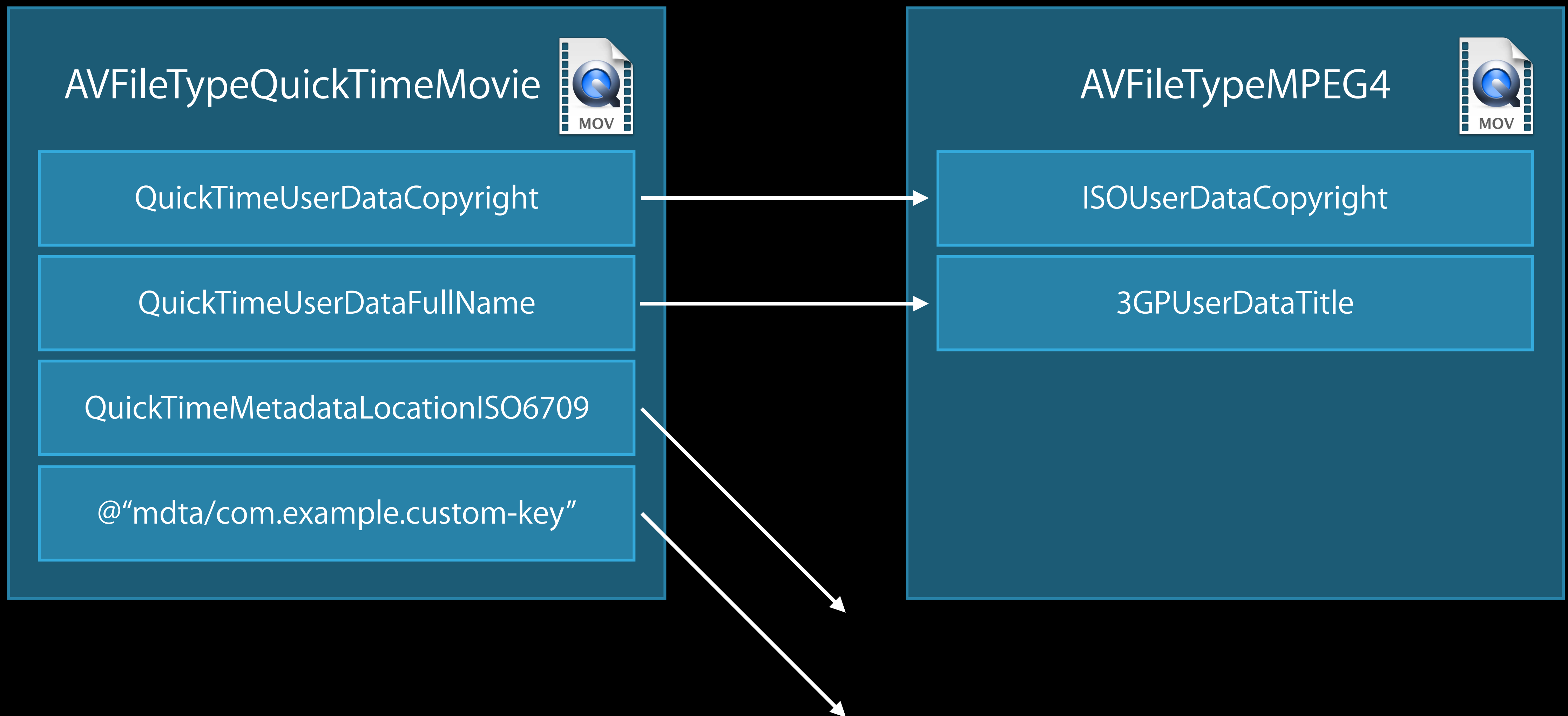


Output File Type

Different from source

Source

Output



Choosing a File Type

AVAssetWriter and AVAssetExportSession

Options

- Use same file type as source

```
[URL getResourceValue: &UTI forKey: NSURLTypeIdentifierKey error:&error];
```

- Use QuickTime Movie
 - Avoid conversions
 - Supports timed metadata

Check results

- AVAsset/AVAssetTrack

Output File Type

Manual conversion

Custom identifiers -> equivalent built-in identifiers

ID3 -> iTunes

- `<AVFoundation/AVMetadataIdentifiers.h>`

Summary

Metadata in AVFoundation

- Inspection
- Authoring

Timed metadata

- Reading and playback
- Authoring

Privacy

Authoring best practices

More Information

Evangelism

evangelism@apple.com

Documentation

AVFoundation Programming Guide

<http://developer.apple.com/library/ios/documentation/AudioVideo/Conceptual/AVFoundationPG/>

Apple Developer Forums

<http://devforums.apple.com>

Related Sessions

-
- | | | |
|--|----------|-------------------|
| ● Mastering Modern Media Playback | Mission | Tuesday 11:30AM |
| ● Camera Capture: Manual Controls | Marina | Wednesday 11:30AM |
| ● Direct Access to Video Encoding and Decoding | Nob Hill | Thursday 11:30AM |
-

Labs

-
- AVFoundation Lab Media Lab A Tuesday 3:15PM

 - AVFoundation and Camera Capture Lab Media Lab A Wednesday 12:45PM

 - HTTP Live Streaming Lab Media Lab A Thursday 9:00AM

 - AVFoundation and Camera Capture Lab Media Lab A Thursday 2:00PM

 WWDC14