**Playing Media**

IP-516\_AppleTV App

**Version number:** 0.1  
**Main author:** AZ, JB  
**Last revision:** 21.10.2016  
**Status:** in work  
**Filename:** PlayingMedia  
**Client:** Michael Keller, UPC

Authors

|  |  |
| --- | --- |
| **Name** | **Initials** |
| Andrea Zirn | AZ |
|  |  |
|  |  |

Revision list

| **Version** | **Date** | **Comment** | **Author** |
| --- | --- | --- | --- |
| 0.1 | 29.11.2016 | Initial version | AZ |
|  |  |  |  |
|  |  |  |  |

Table of Content

Authors 2

Revision list 2

Table of Content 3

1 Heading 1 3

1.1 Heading 2 3

1.1.1 Heading 3 3

# AVFoundation[[1]](#footnote-1)

Build a full-featured video playback with AVKit and AVFoundation.

## Asset Model

The framework models assets using the AVAsset class, which is an abstract, immutable type representing a single media resource. An instance of AVAsset can model local file-based media, such as a QuickTime movie or an MP3 audio file, but can also represent an asset progressively downloaded from a remote host or streamed using HTTP Live Streaming (HLS). It works with media in two important ways

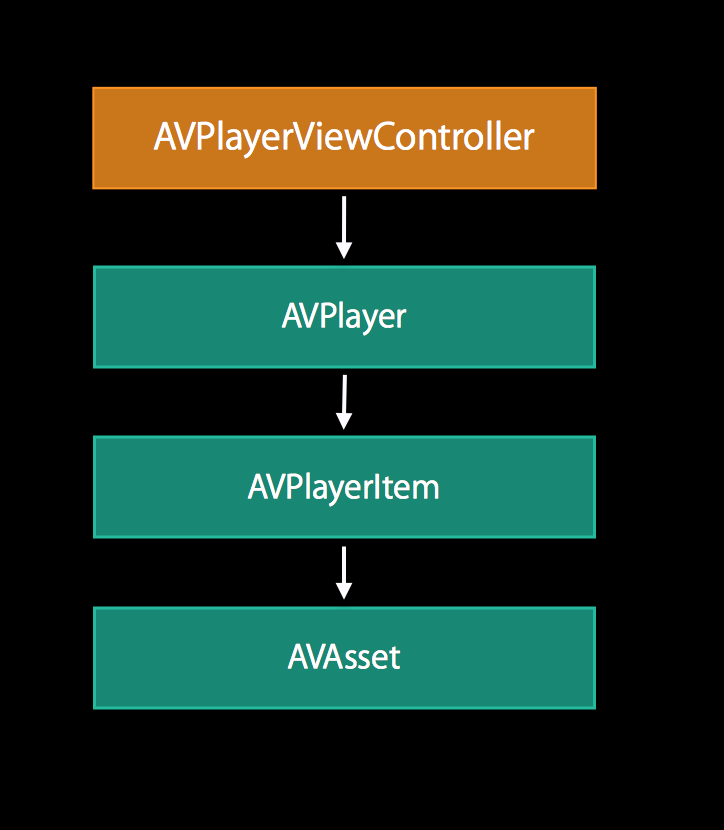
1. it provides a level of independence from the media format. It gives you a consistent interface for managing and interacting with your media regardless of its underlying type.
2. It provides a level of independence from the media’s location. You create an asset instance by initializing it with the media’s URL. This could be a local URL, such as one contained within your app bundle or elsewhere on the file system, or it could also be a resource such as an HLS stream hosted on a remote server.

AVAsset is a container object composed of one or more instances of AVAssetTrack, which models the asset’s uniformly typed media streams. The most commonly used track types are audio and video tracks.

You retrieve an asset’s collection of tracks using its tracks property. If you want to perform operations on a subset rather than on its complete collection, AVAsset provides methods to retrieve subsets of tracks based on criteria such as identifier, media type, or characteristic.

Creating an asset does not automatically load its properties or prepare it for any particular use. Instead, the loading of an asset’s property values is deferred until they are requested. So, the Assets represent the media you want to play, but are only part of the picture

# Playing Media

The Assets are only one part of the construct that is needed for playing media in your tvOS application. The object on the Figure 1 are also necessary.

Figure

## AVPlayer

The AVPlayer is the central class driving the playback use case. It is a controller object that manages playback and timing of a media asset. It is used to play a single media asset at a time. If you want to create and manage a queue of media assets to be played sequentially the subclass AVQueuePlayer can be used.

## AVPlayerItem

Because the AVAssets only models the static aspects of the media, such as its duration, it needs an instance of its dynamic counterpart, the AVPlayerItem. This object models the timing and presentation state of an asset played by AVPlayer.

## AVKit and AVPlayerLayer

Since AVPlayer and AVPlayerItem are nonvisual objects on their own and unable to present and asset’s video on screen there is an option needey to display the content. This can either way be the AVKit or the AVPlayerLayer.

The AVKit is the best way to present your video content by using the AVKit framework’s AVPlayerViewController in tvOS. These objects present the video content, along with playback controls and other media features.

Another option is the AVPlayerLayer. It is use when you’re building a custom interface for your player. This player doesn’t present any playback controls but simply presents the visual content pf a player onscreen. Eventually, you are responsible to build the playback transport controls to play, pause, and seek through the media.

## Observing Playback State

AVPlayer and AVPlayerItem are dynamic objects whose state changes frequently. You often want to take actions in response to these changes, and the way you do so is by using Key-Value Observing (KVO). Using KVO makes it easy for you to observe state changes to AVPlayer and AVPlayerItem and take actions in response.

One of the most important AVPlayerItem properties to observe is its status. The status indicates if the player item is ready for playback and generally available for use. When you first create a player item, its status has a value of AVPlayerItemStatusUnknown, meaning its media hasn’t been loaded or been enqueued for playback. When you associate a player item with AVPlayer, it immediately begins enqueuing the item’s media and preparing it for playback. The player item becomes ready for use when its status changes to AVPlayerItemStatusReadyToPlay.

1. https://developer.apple.com/library/content/documentation/AudioVideo/Conceptual/MediaPlaybackGuide/Contents/Resources/en.lproj/ExploringAVFoundation/ExploringAVFoundation.html#//apple\_ref/doc/uid/TP40016757-CH4-SW9 (29.11.2016)

   Figure 1: http://devstreaming.apple.com/videos/wwdc/2014/503xx50xm4n63qe/503/503\_mastering\_modern\_media\_playback.pdf [↑](#footnote-ref-1)