

[AVFoundation](#) / [Media reading and writing](#) / Creating images from a video asset

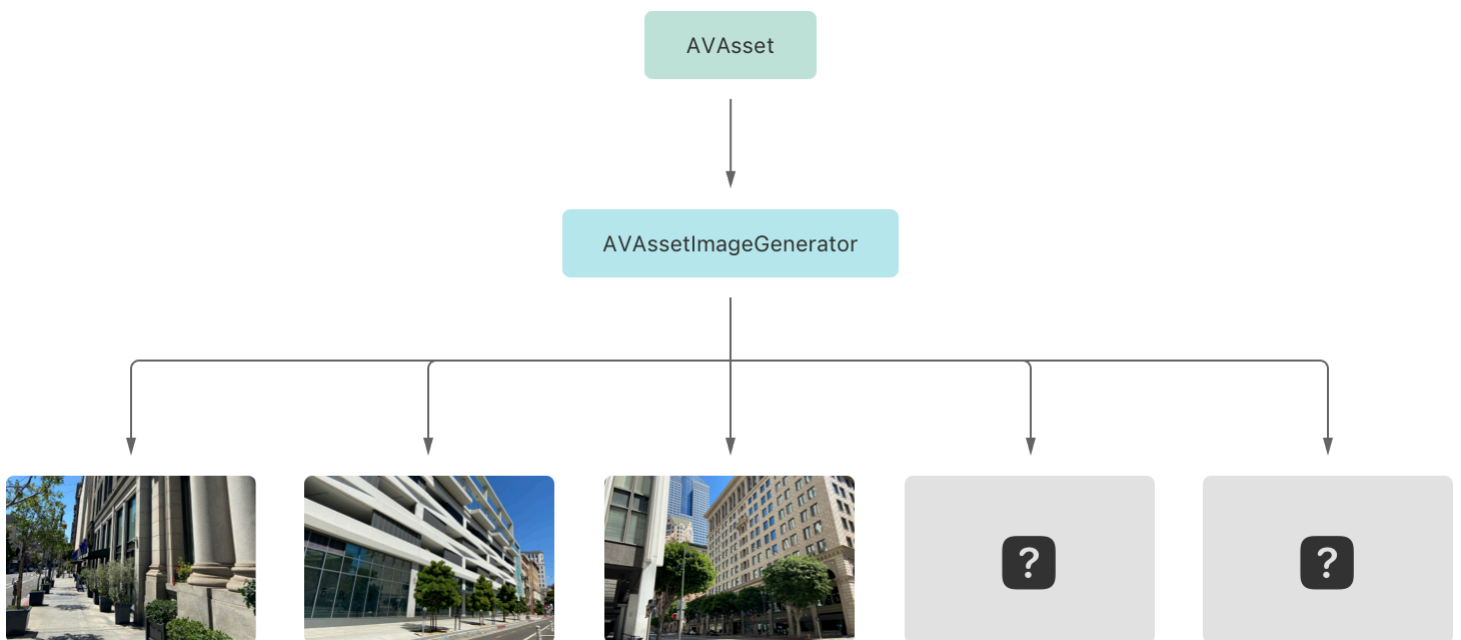
## Article

# Creating images from a video asset

Display images for specific times within the media timeline by generating images from a video's frames.

## Overview

A common requirement when you build video apps is to display images of a video's content. You may want to show a single thumbnail next to a video in a list, or show several images spanning a video's duration that you use to build a visual scrubbing or selection user interface. AVFoundation makes it simple to produce images from your app's video content by using [AVAssetImageGenerator](#).



[AVAssetImageGenerator](#) is a utility class that creates images from assets that contain one or more video tracks. You can use it to generate an image for a single point in time, or a series of images by passing it an array of times.

# Configure an image generator

Create an instance of `AVAssetImageGenerator` by passing it a video asset to retrieve images from. A generator supports creating images from local and remote file-based media, and also from HTTP Live Streaming (HLS) video that provides I-frame only renditions.

```
let asset: AVAsset = // A video asset.  
let generator = AVAssetImageGenerator(asset: asset)
```

## Note

You can generate images from an HLS stream that provides a media playlist with the EXT-X-I-FRAMES-ONLY tag. The durations the playlist specifies between I-frames determines the time resolution of image-generation requests.

An image generator is ready to produce images when you create it, but it also provides configurable properties that you can use to customize its behavior. One property that you typically set is its `maximumSize`. By default, a generator produces images at the video's native resolution, which is often larger than required, and can be CPU and memory intensive. Instead, set a value for its `maximumSize` property to constrain the size of its output. For example, you can constrain the width to a specific value, and specify 0 to proportionally scale the height, by setting a maximum size as follows.

```
// Generate the equivalent of a 150-pixel-wide @2x image.  
generator.maximumSize = CGSize(width: 300, height: 0)
```

# Specify the time precision

By default, an image generator creates images near the time you request, but its actual time may be slightly before or after for efficiency sake. To change the amount of time you allow the generator to deviate from the requested time, set the value of its `requestedTimeToleranceBefore` and `requestedTimeToleranceAfter` properties. For example, to allow image generation to occur over a time range that starts no earlier than your requested time, and no later than two seconds after, set its tolerance values as shown below.

```
// Configure the generator's time tolerance values.  
generator.requestedTimeToleranceBefore = .zero  
generator.requestedTimeToleranceAfter = CMTime(seconds: 2, preferredTimescale: 600)
```

You can request frame-accurate image generation by setting the values of the before and after tolerances to zero. However, configuring the generator this way requires it to perform additional frame decoding, which impacts its image generation speed.

## Generate images

Retrieving and decoding the media data that a generator requires to produce images takes an unknown amount of time. It's not safe to perform image generation synchronously because it blocks the calling thread, which can degrade your app's user experience and may even result in a crash. Instead, perform image generation asynchronously.

To create a single image from the video timeline, call the asynchronous `image(at:)` method, passing it your requested time. The following example shows how to create an image for the first frame in the video.

```
// Generate an image at time zero.
let (image, actualTime) = try await generator.image(at: .zero)
```

Because the time at which it creates an image may vary from your requested time, the method returns the actual image-generation time along with the image in a tuple. If you don't require this information, you can retrieve the image more succinctly as shown below.

```
// Generate an image at time zero. Access the image alone.
let image = try await generator.image(at: .zero).image
```

You can also use an image generator to create a sequence of images from the video timeline. To generate multiple images from a video asset, call the asynchronous `images(for:)` method as shown below.

```
let times: [CMTime] = // An array of times at which to create images.
for await result in generator.images(for: times) {
    switch result {
    case .success(requestedTime: let requested, image: let image, actualTime: let actualTime):
        // Process the image for the requested time.
    case .failure(requestedTime: let requested, error: let error):
        // Handle the failure for the requested time.
    }
}
```

This method returns an asynchronous sequence of type [AVAssetImageGenerator.Images](#) that you iterate over and await results from. If image generation for a requested time succeeds, the result provides the image, along with the requested and actual image generation times. If a requested time fails, the result provides the time and an error that describes the failure.

---

## See Also

### Image generation

```
class AVAssetImageGenerator
    An object that generates images from a video asset.
```