Framework

# Background Assets

Improve or eliminate the time people wait while your app downloads assets.

iOS 16.0+ | iPadOS 16.0+ | Mac Catalyst 16.0+ | macOS 13.0+ | tvOS 18.4+ | visionOS 2.4+

## Overview

Let the system manage your asset downloads on people's devices for you according to your preferences, and optionally, host your assets on Apple servers. For example, a game might specify that the system downloads tutorial assets before first launch and other assets on demand — such as level packs, 3D character models, and textures — as a player progresses during gameplay. The default implementation of Managed Background Assets handles downloads, updates, compression, and more for you. If you choose Apple-Hosted Background Assets, you upload your assets to App Store Connect and maintain them there, similar to app builds.

visionOS availability

For compatible iPad and iPhone apps, Background Assets is available in visionOS 1.0 and later. For apps built for visionOS, Background Assets is available in visionOS 2.4 and later.

To use these features:

1. Group your assets into asset packs containing manifest files that specify a download policy and other details. You can specify that the system downloads essential asset packs before launch, prefetches others without blocking launch, and downloads some only on demand when you need them. For more information, see Creating managed asset packs.

2. Add a managed Background Download extension to your project and configure your project accordingly. Optionally, customize your downloader extension and add code to your app that downloads on-demand assets when needed. For more information, see Downloading Apple-hosted asset packs.

3. For either Apple-hosted or self-hosted asset packs, before you distribute your app, test your code using a local mock serve included in your Xcode install. For more information, see Testing asset packs locally.

4. For Apple-hosted assets, upload your asset packs to App Store Connect before you distribute your app through TestFlight or the App Store. For more information, see the Overview of Apple-hosted asset packs in App Store Connect Help.

Alternatively, you can manage and host asset downloads yourself using the low-level Background Assets APIs. For more information, see Configuring an unmanaged Background Assets project and Downloading essential assets in the background.

Important

Use the Background Assets framework only to download additional assets for your app; don’t use it for any other purposes. For example, don’t collect or transmit data to identify a user or device or to perform advertising or advertising measurement.

# Topics

## Essentials

📄 Creating managed asset packs

Create managed asset packs, choose download options, and upload Apple-hosted asset packs to App Store Connect.

Downloading Apple-hosted asset packs

Configure your project and write the code to download asset packs hosted by Apple.

Testing asset packs locally

Test your system-managed asset packs using a mock server on your Mac.

## Managed asset packs

`struct AssetPack`

An archive of assets that the system downloads together.

`actor AssetPackManager`

An actor that manages asset packs.

`protocol ManagedDownloaderExtension`

An app extension that uses the system implementation to schedule asset-pack downloads automatically.

`BAAppGroupID`

The app group identifier that you share between your app and the extension that uses asset packs.

`BAHasManagedAssetPacks`

A Boolean value that indicates whether you let the system automatically manage your asset packs.

## Apple-hosted managed asset packs

`BAUsesAppleHosting`

A Boolean value that indicates whether you use Apple's service to host your asset packs.

## Self-hosted unmanaged asset packs

`struct AssetPackManifest`

A representation of a manifest that lists asset packs that are available to download.

## Unmanaged asset downloads

Configuring an unmanaged Background Assets project

Manage and download individual assets yourself by configuring your app and extension targets.

{} Downloading essential assets in the background

Fetch the assets your app requires before its first launch using an app extension and the Background Assets framework.

`BAManifestURL`

The location URL of the app’s manifest file that contains the names and sizes of assets.

`BAInitialDownloadRestrictions`

The restrictions that apply to the set of assets that download immediately after app installation.

`BAEssentialMaxInstallSize`

The combined, maximum size of the essential assets that the system downloads before it launches your app in bytes.

`BAMaxInstallSize`

The combined, maximum size, in bytes, of the non-essential assets that download immediately after app installation.

`class BADownloadManager`

An object that manages the queue of scheduled asset downloads.

`protocol BADownloaderExtension`

An interface for reacting to app life-cycle events and processing concluded asset downloads while your app isn’t running.

`protocol BADownloaderExtensionConfiguration`

`class BAURLDownload`

An object that represents a remote asset to download.

`class BADownload`

An object that represents an in-progress or concluded asset download.

## Errors

`enum ManagedBackgroundAssetsError`

An error for a managed asset pack.

`let BAErrorDomain: String`

```
enum BAErrorCode
```