Article

Configuring an unmanaged Background Assets project

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

Overview

To opt out of Managed Background Assets, add a Self-Hosted, Unmanaged extension target to your project, configure the App Groups capability for both the app and extension target, and add some Background Asset keys to the app's information property list. Then the system notifies your extension when system events occur so that your extension can initiate downloads.

> **Note**
>
> For information about Apple-Hosted Background Assets, see Downloading Apple-hosted asset packs.

The system launches the extension during the first install and subsequent updates, before a person launches the app, and periodically in the background when the app isn't running. The sequence of events is:

1. A person installs or updates your app on the device.
2. The system prevents the app from launching and begins downloading your manifest file using the URL you provide.
3. During the manifest download, the system reports progress to the App Store.
4. When the download completes, the system launches your app extension, sending it a content request with the location of the manifest file on disk.

5. The extension uses the manifest file, which should contains the asset URLs and file sizes, to return a set of download requests to the system.

6. The system pauses, or terminates, the extension and begins the downloads.

7. When the downloads complete, the system notifies the extension and allows the app to launch.

The flow for periodic content requests is identical to the app install and updates, except the system determines when periodic events occur, depending on a person's usage and their system settings. For example, the system factors in whether a person enables the Low Power Mode and Background App Refresh settings.

> **Note**
>
> It's your responsibility to create manifest files for your self-hosted, unmanaged assets (using your format of choice) that your code parses to get the URLs and file sizes to the system.

Add a Self-Hosted, Unmanaged extension to your project

Choose New > Target, select the Background Download template under Application Extension, and click Next. In the dialog, enter a product name, choose Self-Hosted, Unmanaged as the extension type, and click Finish. In the next dialog, click Activate to use the extension scheme Xcode creates.

If you don't have an Xcode project for your app, first create one from an Application template under the platform you support, such as iOS or macOS. For more information, see Creating an Xcode project for an app.

Add the App Groups capability

Add your app and extension targets to the same app group so that they can communicate and share data.

Add the App Groups capability to both your app and extension target. For macOS apps, also add the App Sandbox capability to both targets. For more information, see Adding capabilities to your app.

Then, add both targets to the same app group. In the project editor, select the app target, and then add a unique ID for the group under App Groups on the Signing & Capabilities pane. Xcode automatically selects the new group ID. Select the extension target, then go to App Groups, click Refresh, and select the same group ID.

The app and extension are now in the same app group and can share the asset files. For more information on configuring app groups, and additional steps for macOS apps, see Configuring app groups.

Add required information property list keys

Configure Background Assets for your app target by setting information property list keys. In the project editor, select the app target and click the Info tab. Then, add the following keys to the information property list file:

`BAManifestURL`
: Set this key to the URL for your manifest file. You provide the manifest file that contains the URLs and file sizes for the assets you want to download. After installing your app on a device, the system uses the `BAManifestURL` key to download the manifest file before it launches your extension.

`BAInitialDownloadRestrictions`
: Use this dictionary to provide the constraints on your downloads that the system uses after it installs your app on the device. Be as accurate as possible when setting these dictionary keys:

  `BADownloadAllowance`
  : Set this key to the upper bounds of the download size of nonessential asset files combined, not individual files, in bytes. If you compress the files, use the compressed file sizes that the system downloads, not the uncompressed file sizes.

  `BADownloadDomainAllowList`
  : Set this key to the domain names that you want the extension to download assets from in DNS format. To use a wildcard domain name, prefix the string with an asterisk (*). For example, the `*.example.com` wildcard matches `assets.example.com` and `download.example.com`.

  `BAEssentialDownloadAllowance`
  : Set this key to the upper bounds of the download size of the essential download files only in bytes that download before the system launches your app. Use the compressed file sizes, not the uncompressed file sizes.

`BAEssentialMaxInstallSize`
: Set this key to the combined, maximum size of the essential assets only that download before the system launches your app. Use the uncompressed size of the files for this value.

`BAMaxInstallSize`
: Set this key to the combined, maximum size of the nonessential assets that download after the system downloads essential assets. Use the uncompressed size of the files for this value.

For more examples of these information property list keys, see the Downloading essential assets in the background sample code project. For more information on editing the information property list file, see Managing your app's information property list values.

See Also

Unmanaged asset downloads

{} 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.