Article

Improving the player experience for games with large downloads

Provide ample content in your base installation and then use on-demand resources and the Background Assets API to handle additional content.

Overview

As games become larger and more visually impressive, it's important to balance initial download time with a smooth gameplay experience and responsible disk usage. Providing a small base installation that merely begins another installation phase doesn't make for a great experience when players are excited to play your game. Instead, provide enough content in your initial download that a player can evaluate your game, and keep your initial download to under 30 minutes. Apple provides two technologies that your game can use to provide the right amount of content in your base installation and help you download additional content in the background: on-demand resources and the Background Assets framework.

Maximize your app bundle's size

If your game targets iOS 18, iPadOS 18, or any version of tvOS, your game's app bundle can be up to 4 GB in size, as outlined in the table below. You need to deliver any additional content through on-demand resources, the Background Assets framework, or your own in-game solution.

Minimum deployment target	Maximum bundle size
iOS 17, iPadOS 17, or earlier	2 GB
iOS 18 or iPadOS 18	4 GB
tvOS (any version)	4 GB

Include additional content in your game's base installation

With on-demand resources, you can specify additional content to download from the App Store while your game installs. Alternatively, you can use the Background Assets framework to write an extension that downloads content from your own website or CDN. With either solution, the player sees a single progress indicator while the system downloads your game's app bundle and any additional first-launch content. A purely in-app solution can't include additional content in the base installation.

Apply tags to use on-demand resources

On-demand resources are assets hosted by Apple. To use them, split your game's content into asset packs, apply one or more tags to each pack, and submit your asset packs to the App Store along with your game. For more information on applying tags to your assets, see Creating and Assigning Tags.

Identify which tags apply to essential and early-game content

After you assign tags to your asset packs, use the Prefetch view to identify which tags you should include in your game's base installation. Asset packs matching these tags are included in the base installation progress and are available when your game first launches. You can also specify an additional set of tags to automatically download after the base installation completes. This prefetched content might not finish downloading before the user launches your game. For more information, see Prefetching Tags.

Use the on-demand resources API for later-game and optional content

Your game can programmatically download asset packs using the `NSBundleResourceRequest` API. This allows you to save disk space by only downloading content after the player unlocks it through progression or In-App Purchases. Because asset packs can have multiple tags, splitting your content into smaller asset packs can further reduce your game's disk space usage. Use the `NSBundleResourceRequest` API to request tags, which prompt the system to download any matching asset packs as necessary.

For content unlocked by progress, requesting tags before they're needed can make the content available to the player without waiting. In case the download doesn't complete before the player needs the content, use the `progress` property to keep the user informed.

When the player completes part of your game's content, use the `setPreservationPriority(_:forTags:)` method of the `Bundle` class to inform the system that the asset packs

containing that content are no longer necessary.

Stay within the on-demand resources limits

If your game targets iOS 18, iPadOS 18, or tvOS 18, it can have up to 70 GB of assets in up to 1,000 asset packs, an unlimited subset of which you can include in your initial install tags or prefetch tags. The table below shows the resource limits:

Item	Size prior to iOS 17, iPadOS 17, and tvOS 17	Size when targeting iOS 17, iPadOS 17, or tvOS 17	Size when targeting iOS 18, iPadOS 18, or tvOS 18
Tag	512 MB	512 MB	8 GB
Asset packs	1,000	1,000	1,000
Initial install tags	2 GB	2 GB	No limit
Initial install and prefetch tags	4 GB	4 GB	No limit
In-use on-demand resources	2 GB	2 GB	No limit
Hosted on-demand resources	20 GB	40 GB	70 GB

Use Background Assets for content outside the App Store

Use the Background Assets framework if you want to host your game's additional content outside the App Store. To use Background Assets, write a Background Assets extension that tells the system how to download assets from your website or CDN. Your extension can also react to authentication challenges and changes in download status. Your game uses the `BADownloadManager` API to stay informed of completed downloads and schedule new downloads.

Related sessions from WWDC22 and WWDC23

Session 110403: Meet Background Assets

Session 10108: What's New in Background Assets

Add information about your essential content

The `BAEssentialDownloadAllowance` key tells the App Store how much additional content your Background Assets Extension downloads during installation. Your App Store listing includes this information in the displayed download size. The `BAEssentialMaxInstallSize` key tells the system how much space those assets require when uncompressed.

Extract your assets to the correct location

The system places downloaded files in a purgeable location. When the system finishes downloading your essential content, it informs your game if the user launched it immediately or it informs your extension. The process that receives the notification must extract the assets to a stable location where your game can find them. Choose a location within your game's `Caches` directory. Don't extract your assets to the `Documents` directory because this causes them to get included in device backups. To find the appropriate `Caches` directory, use the `FileManager` API:

```
@implementation MyGameBADownloaderExtension

- (void)backgroundDownload:(BADownload *)download finishedWithFileURL:(NSURL *)fileU
    NSFileManager *fileManager = [NSFileManager defaultManager];
    NSURL *cacheRoot = [fileManager URLForDirectory:NSCachesDirectoryinDomains:NSUse

    // Create a subdirectory with the same name as this downloaded asset.
    NSString *downloadedFileName = fileURL.lastPathComponent.stringByDeletingPathExt
    NSURL *extractionDir = [cacheRoot URLByAppendingPathComponent:downloadedFileName

    NSError *error;
    BOOL success = [fileManager createDirectoryAtURL:extractionDir withIntermediateD
    if (success) {
        success = extractCompressedFile(fileURL, extractionDir);
        if (!success) {
            os_log_fault(OS_LOG_DEFAULT, "Couldn't extract download from URL %@ to U
        }
    } else {
        os_log_fault(OS_LOG_DEFAULT, "Couldn't create cache directory at URL %@", ex
```

```
    }

    // Clean up the downloaded file after trying to extract it.
    success = [fileManager removeItemAtURL:fileURL error:&error];
    if (!success) {
        os_log(OS_LOG_DEFAULT, "Couldn't clean up download at URL %@", fileURL);
    }
}

@end
```

Redownload essential assets on failure

If an essential download fails, the system skips it, and installation of your app proceeds. Therefore, reschedule the asset as nonessential so that it begins downloading. The system delivers the failure notification to your game if it's running or else to your extension.

```
- (void)backgroundDownload:(BADownload *)download failedWithError:(NSError *)error {
    // If the `BAManifestURL` fails to download, the system notifies the extension.
    // The type of the manifest isn't a `BAURLDownload`, so you can key off of
    // the download's type to filter it out.

    // If the failed download is essential, you can re-enqueue it in the background
    // so it can download at a later time.
    if (download.isEssential) {
        os_log(OS_LOG_DEFAULT, "Rescheduling failed essential download with identifi
        download = [download copyAsNonEssential];
    }

    if (![BADownloadManager sharedManager] scheduleDownload:download error:error]) {
        os_log_fault("Couldn't reschedule essential download with identifier '%@': %
    }
}
```

See Also

Essentials

Initializing and configuring Game Center

Enable Game Center, configure features, and test them locally in your Xcode project.

Authenticating a player

Confirm player credentials and device capabilities and check for account restrictions.

`Game Center Entitlement`

A Boolean value that indicates whether users of the app may see and compare achievements on a leaderboard, invite friends, and start multiplayer games.