<https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/index.html>

# Addressables

The Addressables system provides tools and scripts to organize and package content for your application and an API to load and release assets at runtime.  
Addressables 系统提供用于组织和打包应用程序内容的工具和脚本，以及用于在运行时加载和释放资产的 API。

When you make an asset "Addressable," you can use that asset's address to load it from anywhere. Whether that asset resides in the local application or on a content delivery network, the Addressable system locates and returns it.  
当您将资产设置为“Addressables”时，您可以使用该资产的地址从任何位置加载它。无论该资产驻留在本地应用程序还是内容交付网络上，Addressables系统都会找到并返回它。

Adopt the Addressables system to help improve your project in the following areas:  
采用 Addressables系统来帮助在以下方面改进您的项目：

* **Flexibility**: Addressables give you the flexibility to adjust where you host your assets. You can install assets with your application or download them on demand. You can change where you access a specific asset at any stage in your project without rewriting any game code.  
  **灵活性**：Addressables使您可以灵活地调整资产的托管位置。您可以随意使用应用程序安装的自带资源或按需下载它们。您可以在项目的任何阶段更改访问特定资源的位置，而无需重写任何游戏代码。
* **Dependency management**: The system automatically loads all dependencies of any assets you load, so that all meshes, shaders, animations, and other assets load before the system returns the content to you.  
  **依赖关系管理**：系统会自动加载您加载的任何资产的所有依赖关系，以便在系统将内容返回给您之前加载所有网格、着色器、动画和其他资产。
* **Memory management**: The system unloads assets as well as loads them, counting references automatically and providing a robust profiler to help you spot potential memory problems.  
  **内存管理**：系统卸载和加载资产，自动计算引用，并提供强大的性能分析器来帮助您发现潜在的内存问题。
* **Content packing**: Because the system maps and understands complex dependency chains, it package AssetBundles efficiently, even when you move or rename assets. You can prepare assets for both local and remote deployment, to support downloadable content and reduced application size.  
  **内容打包**：由于系统映射并理解复杂的依赖关系链，因此即使您移动或重命名资产，它也能高效地打包资产包。您可以为本地和远程部署准备资产，以支持可下载的内容并减小应用程序大小。

For an introduction to the Addressables system see [Simplify your content management with Addressables](https://unity.com/how-to/simplify-your-content-management-addressables).

**NOTE**

The Addressables system builds upon Unity AssetBundles. If you want to use AssetBundles in your projects without writing your own detailed management code, you should use Addressables.  
Addressables系统基于 Unity AssetBundles 构建。如果要在项目中使用 AssetBundle，而无需编写自己的详细管理代码，则应使用Addressables。

## Adding Addressables to an existing Project 向现有项目添加Addressables系统

You can adopt Addressables in an existing Unity Project by installing the Addressables package. To do this, you need to assign addresses to your assets and refactor any runtime loading code. See [Upgrading to the Addressables system](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsMigrationGuide.html) for more information.  
您可以通过安装Addressables package 在现有 Unity 项目中采用Addressables。为此，您需要为资产分配地址并重构任何运行时加载代码。 有关详细信息，请参阅[升级到Addressables系统](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsMigrationGuide.html)。

Although you can integrate Addressables at any stage in a project’s development, Unity recommends that you start using Addressables immediately in new projects to avoid unnecessary code refactoring and content planning changes later in development.  
尽管您可以在项目开发的任何阶段集成Addressables，但 Unity 建议您立即在新项目中开始使用Addressables，以避免在开发后期进行不必要的代码重构和内容规划更改。

# Getting started 开始

Once you have [installed the Addressables package](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsGettingStarted.html#installation) in your Unity Project, you can get started.  
在 Unity 项目中[安装Addressables包](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsGettingStarted.html#installation)后，就可以开始了。

The basic steps to using Addressables include:  
使用可寻址对象的基本步骤包括：

* Make your assets Addressable  
  使您的资产可寻址
* Reference and load those assets in code using the Addressables API  
  使用可寻址对象 API 在代码中引用和加载这些资产
* Build your Addressable assets  
  构建可寻址资产

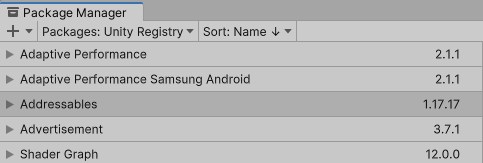
See the [Space Shooter project](https://github.com/Unity-Technologies/Addressables-Sample/tree/master/Basic/SpaceShooter) in the [Addressables-Sample](https://github.com/Unity-Technologies/Addressables-Sample) repository for an example of a project set up to use Addressable assets.

##### NOTE

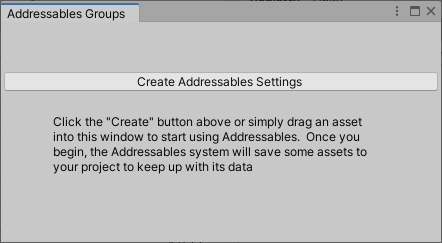
This Getting Started topic doesn't discuss the various ways you can organize your Addressable content. For information on that topic, see [Organizing Addressable Assets](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsDevelopmentCycle.html#organizing-addressable-assets).  
此入门主题不讨论组织可寻址内容的各种方法。有关该主题的信息，请参阅[Organizing Addressable Assets](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsDevelopmentCycle.html#organizing-addressable-assets)。

## Installation 安装

To install the Addressables package in your project, use the Unity Package Manager:

1. Open the Package Manager (menu: **Window > Package Manager**).
2. Set the package list to display packages from the **Unity Registry**.  
   
3. Select the Addressables package in the list.
4. Click **Install** (at the bottom, right-hand side of the Package Manager window).

To set up the Addressables system in your Project after installation, open the **Addressables Groups** window and click **Create Addressables Settings**.

  
Before initializing the Addressables system in a Project

When you run the **Create Addressables Settings** command, the Addressables system creates a folder called, AddressableAssetsData, in which it stores settings files and other assets it uses to keep track of your Addressables setup. You should add the files in this folder to your source control system. Note that Addressables can create additional files as you change your Addressables configuration. See [Addressables Settings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html) for more information about the settings themselves.  
运行**Create Addressables Settings**命令时，可寻址对象系统会创建一个名为 AddressableAssetsData 的文件夹，该文件夹存储用于跟踪可寻址设置的设置文件和其他资产。应将此文件夹中的文件添加到源代码管理系统（Git或SVN）中。请注意，可寻址对象可以在您更改可寻址对象配置时创建其他文件。有关设置本身的详细信息，请参阅[Addressables Settings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html)。

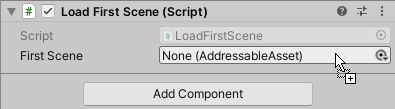
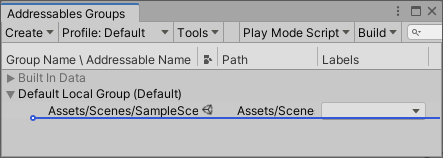
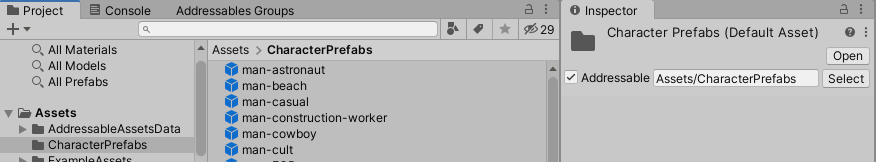
##### NOTE

For instructions on installing a specific version of Addressables or for general information about managing the packages in a Project, see [Packages](https://docs.unity3d.com/2019.4/Documentation/Manual/PackagesList.html).  
有关安装特定版本的可寻址对象的说明或有关在项目中管理包的一般信息，请参阅[Packages](https://docs.unity3d.com/2019.4/Documentation/Manual/PackagesList.html)。

## Making an asset Addressable

## 设置可寻址资产

You can mark an asset as Addressable in the following ways:

* Check the **Addressable** box in the asset's Inspector:  
  
* Drag or assign the asset to an AssetReference field in an Inspector:  
  
* Drag the asset into a group on the **Addressables Groups** window:  
  
* Put the asset in a Project folder that's marked as Addressable:  
  

Once you make an asset Addressable, the Addressables system adds it to a default group (unless you place it in a specific group). Addressables packs assets in a group into [AssetBundles](https://docs.unity3d.com/2019.4/Documentation/Manual/AssetBundlesIntro.html) according to your group settings when you make a [content build](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Builds.html). You can load these assets using the [Addressables API](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadingAddressableAssets.html).  
将资产设置为可寻址后，可寻址对象系统会将其添加到默认组中（除非您将其放置在特定组中）。可寻址对象会在您进行[content build](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Builds.html)时，根据您的组设置将组中的资产打包到 [AssetBundles](https://docs.unity3d.com/2019.4/Documentation/Manual/AssetBundlesIntro.html)中。您可以使用 [Addressables API](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadingAddressableAssets.html) 加载这些资产。

##### NOTE

If you make an asset in a [Resources folder](https://docs.unity3d.com/2019.4/Documentation/Manual/SpecialFolders.html) Addressable, Unity moves the asset out of the Resources folder. You can move the asset to a different folder in your Project, but you cannot store Addressable assets in a Resources folder.  
如果将[Resources](https://docs.unity3d.com/2019.4/Documentation/Manual/SpecialFolders.html)文件夹中的资源设置为可寻址，Unity 会将资源移出资源文件夹。您可以将资源移动到项目中的其他文件夹，但不能将可寻址资源存储在Resources文件夹中。

## Using an Addressable Asset 使用可寻址资产

To load an Addressable Asset, you can:

* [Use an AssetReference referencing the asset](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsGettingStarted.html#using-assetreferences)
* [Use its address string](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsGettingStarted.html#loading-by-address)
* [Use a label assigned to the asset](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsGettingStarted.html#loading-by-label)

See [Loading assets](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadingAddressableAssets.html) for more detailed information about loading Addressable assets.

Loading Addressable assets uses asynchronous operations. See [Operations](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsAsyncOperationHandle.html) for information about the different ways to tackle asynchronous programming in Unity scripts.

##### TIP

You can find more involved examples of how to use Addressable assets in the [Addressables-Sample repo](https://github.com/Unity-Technologies/Addressables-Sample).

### Using AssetReferences

To use an AssetReference, add an AssetReference field to a MonoBehaviour or ScriptableObject. After you create an object of that type, you can assign an asset to the field in your object's Inspector window.  
要使用 AssetReference，请将 AssetReference 字段添加到 MonoBehavior 或 ScriptableObject。创建该类型的对象后，您可以在对象的“检查器”窗口中将资源分配给该字段。

##### NOTE

If you assign a non-Addressable asset to an AssetReference field, Unity automatically makes that asset Addressable and adds it to your default Addressables group. AssetReferences also let you use Addressable assets in a Scene that isn't itself Addressable.  
如果将不可寻址资源分配给“资产引用”字段，Unity 会自动将该资源设置为可寻址，并将其添加到默认的可寻址对象组中。资产引用还允许您在本身不可寻址的场景中使用可寻址资源。

Unity does not load or release the referenced asset automatically; you must load and release the asset using the Addressables API:  
Unity 不会自动加载或释放引用的资源;您必须使用可寻址对象 API 加载和释放资产：

using UnityEngine;

using UnityEngine.AddressableAssets;

using UnityEngine.ResourceManagement.AsyncOperations;

internal class LoadWithReference : MonoBehaviour

{

// Assign in Editor

public AssetReference reference;

// Start the load operation on start

void Start() {

AsyncOperationHandle handle = reference.LoadAssetAsync<GameObject>();

handle.Completed += Handle\_Completed;

}

// Instantiate the loaded prefab on complete

private void Handle\_Completed(AsyncOperationHandle obj) {

if (obj.Status == AsyncOperationStatus.Succeeded) {

Instantiate(reference.Asset, transform);

} else {

Debug.LogError($"AssetReference {reference.RuntimeKey} failed to load.");

}

}

// Release asset when parent object is destroyed

private void OnDestroy() {

reference.ReleaseAsset();

}

}

See [Loading an AssetReference](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadingAddressableAssets.html#loading-an-assetreference) for additional information about loading AssetReferences.

### Loading by address

You can use the address string to load an Asset:

using UnityEngine;

using UnityEngine.AddressableAssets;

using UnityEngine.ResourceManagement.AsyncOperations;

internal class LoadWithAddress : MonoBehaviour

{

// Assign in Editor or in code

public string address;

// Retain handle to release asset and operation

private AsyncOperationHandle<GameObject> handle;

// Start the load operation on start

void Start() {

handle = Addressables.LoadAssetAsync<GameObject>(address);

handle.Completed += Handle\_Completed;

}

// Instantiate the loaded prefab on complete

private void Handle\_Completed(AsyncOperationHandle<GameObject> operation) {

if (operation.Status == AsyncOperationStatus.Succeeded) {

Instantiate(operation.Result, transform);

} else {

Debug.LogError($"Asset for {address} failed to load.");

}

}

// Release asset when parent object is destroyed

private void OnDestroy() {

Addressables.Release(handle);

}

}

Remember that every time you load an Asset, you must also release it.

See [Loading a single asset](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadingAddressableAssets.html#loading-a-single-asset) for more information.

### Loading by label

You can load sets of assets that have the same label in one operation:

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.AddressableAssets;

using UnityEngine.ResourceManagement.AsyncOperations;

internal class LoadWithLabels : MonoBehaviour

{

// Label strings to load

public List<string> keys = new List<string>() { "characters", "animals" };

// Operation handle used to load and release assets

AsyncOperationHandle<IList<GameObject>> loadHandle;

// Load Addressables by Label

void Start() {

float x = 0, z = 0;

loadHandle = Addressables.LoadAssetsAsync<GameObject>(

keys, // Either a single key or a List of keys

addressable => {

//Gets called for every loaded asset

if (addressable != null) {

Instantiate<GameObject>(addressable,

new Vector3(x++ \* 2.0f, 0, z \* 2.0f),

Quaternion.identity,

transform);

if (x > 9) {

x = 0;

z++;

}

}

}, Addressables.MergeMode.Union, // How to combine multiple labels

false); // Whether to fail if any asset fails to load

loadHandle.Completed += LoadHandle\_Completed;

}

private void LoadHandle\_Completed(AsyncOperationHandle<IList<GameObject>> operation) {

if (operation.Status != AsyncOperationStatus.Succeeded)

Debug.LogWarning("Some assets did not load.");

}

private void OnDestroy() {

// Release all the loaded assets associated with loadHandle

Addressables.Release(loadHandle);

}

}

See [Loading multiple assets](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadingAddressableAssets.html#loading-multiple-assets) for more information.

## Managing Addressable assets 管理可寻址资产

To manage your Addressable assets, use the Addressables **Groups** window. Use this window to create Addressables groups, move assets between groups, and assign addresses and labels to assets.  
要管理可寻址资产，请使用Addressables **Groups**窗口。使用此窗口可以创建可寻址对象组、在组之间移动资产以及为资产分配地址和标签。

When you first install and set up the Addressables package, it creates a default group for Addressable assets. The Addressables system assigns any assets you mark as Addressable to this group by default. In the early stages of a Project, you might find it acceptable to keep your assets in this single group, but as you add more content, you should consider creating additional groups so that you have better control over which resources your application loads and keeps in memory at any given time.  
首次安装和设置可寻址包时，它会为可寻址资产创建一个默认组。默认情况下，可寻址对象系统会将您标记为可寻址的任何资产分配给此组。在项目的早期阶段，您可能会发现将资产保留在此单个组中是可以接受的，但是随着添加更多内容，应考虑创建其他组，以便更好地控制应用程序在任何给定时间加载并保留在内存中的资源

Key group settings include:  
Group中关键的设定包括：

* Build path: where to save your content after a content build.
* Load path: where your app or game looks for built content at runtime.

##### NOTE

You can (and usually should) use Profile variables to set these paths. See [Profiles](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html) for more information.

* Bundle mode: how to package the content in the group into a bundle. You can choose the following options:
  + One bundle containing all group assets
  + A bundle for each entry in the group (particularly useful if you mark entire folders as Addressable and want their contents built together)
  + A bundle for each unique combination of labels assigned to group assets
* Content update restriction: Setting this value appropriately allows you to publish smaller content updates. See [Content update builds](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html) for more information. If you always publish full builds to update your app and don't download content from a remote source, you can ignore this setting.

For more information on strategies to consider when deciding how to organize your assets, see [Organizing Addressable assets](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsDevelopmentCycle.html#organizing-addressable-assets).

For more information on using the Addressables Groups window, see [Groups](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html).

## Building Addressable assets 创建可寻址资产

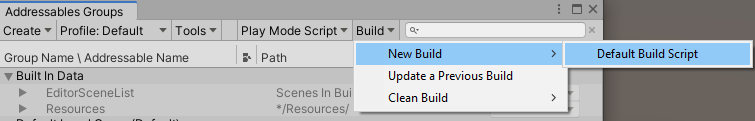
The Addressables content build step converts the assets in your Addressables groups into AssetBundles based on the [group settings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/GroupSettings.html) and the current platform set in the Editor.

In Unity 2021.2+, you can configure the Addressables system to build your Addressables content as part of every Player build or you can build your content separately before making a Player build. See [Building Addressables content with Player builds](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Builds.html#build-with-player) for more information about configuring these options.

If you configure Unity to build your content as part of the Player build, use the normal **Build** or **Build and Run** buttons on the Editor [Build Settings](https://docs.unity3d.com/2019.4/Documentation/Manual/PublishingBuilds.html) window to start a build. Unity builds your Addressables content as a pre-build step before it builds the Player.

In earlier versions of Unity, or if you configure Unity to build your content separately, you must make an Addressables build using the **Build** menu on the **Addressables Groups** window as described in [Making builds]. The next time you build the Player for your project, it uses the artifacts produced by the last Addressables content build run for the current platform. See [Build scripting] for information about automating your Addressables build process.

To initiate a content build from the Addressables Groups window:



1. Open the Addressables Groups window (menu: **Windows > Asset Management > Addressables > Groups**).
2. Choose an option from the **Build** menu:
   * **New Build**: perform a build with a specific build script. Use the **Default Build Script** if you don't have your own custom one.
   * **Update a Previous Build**: builds an update based on an existing build. To update a previous build, the Addressables system needs the addressables\_content\_state.bin file produced by the earlier build. You can find this file in the Assets/AddressableAssetsData/Platform folder of your Unity Project. See [Content Updates](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html) for more information about updating content.
   * **Clean Build**: deletes cached build files.

By default, the build creates files in the locations defined in your [Profile](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html) settings for the **LocalBuildPath** and **RemoteBuildPath** variables. The files that Unity uses for your player builds include AssetBundles (.bundle), [catalog JSON and hash files](#_Build_artifacts), and settings files.

##### WARNING

In most cases, you should not change the local build or load paths from their default values. If you do, you must copy the local build artifacts from your custom build location to the project's [StreamingAssets](https://docs.unity3d.com/2019.4/Documentation/Manual/StreamingAssets.html) folder before making a Player build. Altering these paths also precludes building your Addressables as part of the Player build.

If you have groups that you build to the **RemoteBuildPath**, it is your responsibility to upload those AssetBundles, [catalog, and hash files](#_Build_artifacts) to your hosting server. (If your Project doesn't use remote content, set all groups to use the local build and load paths.)

A content build also creates the following files that Addressables doesn't use directly in a player build:

* addressables\_content\_state.bin: used to make a content update build. If you support dynamic content updates, you must save this file after each content release. Otherwise, you can ignore this file.
* AddressablesBuildTEP.json: logs build performance data. See [Build Profiling](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildProfileLog.html).

See [Building Addressable content](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Builds.html) for more information about how to set up and perform a content build.

### Starting a full content build 完整内容构建

To make a full content build:

1. Set the desired **Platform Target** on the **Build Settings** window.
2. Open the **Addressables Groups** window (menu: **Asset Management > Addressables > Groups**).
3. Choose the\_\_ New Build > Default Build Script\_\_ command from the Build menu of the **Groups** window.

The build process starts.

After the build is complete, you can perform a player build and upload any remote files from your **RemoteBuildPath** to your hosting server.

##### IMPORTANT

If you plan to publish remote content updates without rebuilding your application, you must preserve the addressables\_content\_state.bin file for each published build. Without this file, you can only create a full content build and player build, not an update. See [Content update builds](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html) for more information.  
如果计划在不重新生成应用程序的情况下发布远程内容更新，则必须为每个已发布的版本保留addressables\_content\_state.bin文件。如果没有此文件，则只能创建完整的内容构建和播放器构建，而不能创建更新。有关详细信息，请参阅[Content update builds](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html)。

## Remote content distribution 远程内容分发

You can use Addressables to support remote distribution of content through a Content Delivery Network (CDN) or other hosting service. Unity provides the Unity Cloud Content Delivery (CCD) service for this purpose, but you can use any CDN or host you prefer.  
您可以使用寻址对象支持通过内容分发网络（CDN） 其他托管服务远程分发内容。Unity 为此提供了 Unity 云内容交付（CCD）服务，但您可以使用您喜欢的任何 CDN 或主机。

Before building content for remote distribution, you must:  
在构建用于远程分发的内容之前，您必须：

* Enable the **Build Remote Catalog** option in your AddressableAssetSettings (access using menu: **Windows > Asset Management > Addressables > Settings**).  
  启用**Build Remote Catalog**选项（使用菜单访问：**Windows > Asset Management > Addressables > Settings**）。
* Configure the **RemoteLoadPath** in the [Profile](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html) you use to publish content to reflect the remote URL at which you plan to access the content.  
  在用于发布内容的 [Profile](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html)中配置 **RemoteLoadPath** ，用以关联您计划访问内容的远程 URL。
* For each Addressables group containing assets you want to deliver remotely, set the **Build Path** to **RemoteBuildPath** and the **Load Path** to **RemoteLoadPath**.  
  对于包含要远程交付的资产的每个可寻址对象组，将**Build Path**设置为**RemoteBuildPath**，将**Load Path**设置为 **RemoteLoadPath**。
* Set desired **Platform Target** on the Unity **Build Settings** window.  
  在Unity **Build Settings**窗口中设置所需的**Platform Target**。

After you make a content build (using the Addressables **Groups** window) and a player build (using the **Build Settings** window), you must upload the files created in the folder designated by your profile's **RemoteBuildPath** to your hosting service. The files to upload include:  
进行内容构建（使用Addressables **Groups**窗口）和播放器构建（使用Unity **Build Settings**窗口）后，必须将配置文件的 RemoteBuildPath 指定的文件夹中创建的文件上传到托管服务。要上传的文件包括：

* AssetBundles (name.bundle)
* Catalog (catalog\_timestamp.json)
* Hash (catalog\_timestamp.hash)

See [Distributing remote content](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/RemoteContentDistribution.html) for more information.

## Incremental content updates 增量内容更新

When you distribute content remotely, you can reduce the amount of data your users need to download for an update by publishing incremental content update builds. An incremental update build allows you to publish remote bundles which contain only the assets that have changed since you last published an update rather than republishing everything. The assets in these smaller, updated bundles override the existing assets.  
远程分发内容时，可以通过发布增量内容更新版本来减少用户为更新下载的数据量。增量更新版本允许您发布远程捆绑包，这些捆绑包仅包含自上次发布更新以来已更改的资产，而不是重新发布所有内容。这些较小的更新捆绑包中的资源会覆盖现有资源。

##### IMPORTANT

You must turn on the [Build Remote Catalog](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html#catalog) option before you publish a player build if you want to have the option to publish incremental updates. Without a remote catalog, an installed application doesn't check for updates.  
如果要选择发布增量更新，则必须在发布播放器内部版本之前打开 [Build Remote Catalog](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html#catalog)选项。如果没有远程目录，已安装的应用程序不会检查更新。

For more detailed information about content updates, including examples, see [Content update builds](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html).

### Starting a content update build 创建更新内容

To make a content update, rather than a full build:  
要进行内容更新，而不是完整版本，请执行以下操作：

1. On the **Build Settings** window, set the **Platform Target** to match the target of the previous content build that you are now updating.
2. Open the **Addressables Groups** window (menu: **Asset Management > Addressables > Groups**).
3. From the **Tools** menu, run the **Check for Content Update Restrictions** command. The **Build Data File** browser window opens.
4. Locate the addressables\_content\_state.bin file produced by the previous build. This file is in a subfolder of Assets/AddressableAssestsData named for the target platform.
5. Click **Open**. The **Content Update Preview** window searches for changes and identifies assets that must be moved to a new group for the update. If you have not changed any assets in groups set to "Cannot Change Post Release," then no changes will be listed in the preview. (When you change an asset in a group set to "Can Change Post Release," then Addressables rebuilds all the AssetBundles for the group; Addressables does not move the changed assets to a new group in this case.)
6. Click **Apply Changes** to accept any changes.
7. From the **Build** menu, run the\_\_ Update a Previous Build\_\_ command.
8. Open the addressables\_content\_state.bin file produced by the previous build.

The build process starts.

After the build is complete, you can upload the files from your **RemoteBuildPath** to your hosting server.

##### IMPORTANT

Addressables uses the addressables\_content\_state.bin file to identify which assets you changed. You must preserve a copy of this file for each published build. Without the file, you can only create a full content build, not an update.  
可寻址对象使用addressables\_content\_state.bin文件来标识您更改了哪些资产。必须为每个已发布的版本保留此文件的副本。如果没有该文件，您只能创建完整的内容版本，而不能创建更新。

# Overview of the Addressables system 可寻址设备系统概述

Addressables provides a system that can grow with your project. You can start with a simple setup and then reorganize as your project grows in complexity and your team grows in size, and you can do this all with minimal code changes.  
可寻址对象提供了一个可以随项目一起增长的系统。您可以从简单的设置开始，然后随着项目复杂性的增加和团队规模的扩大而重新组织，并且只需最少的代码更改即可完成所有这些操作。

For example, you could start with a single group of Addressable assets, which Unity loads as a set. Then, as you add more content, you could split your assets into multiple groups so that you can load only the ones you need at a given time. As your team grows in size, you could make separate Unity Projects for developing different types of assets. These auxiliary Projects can produce their own Addressables content builds that you load from the main Project (again with minimal code changes).  
例如，您可以从一组可寻址资源开始，Unity 将其作为一个集合加载。然后，随着您添加更多内容，您可以将资产拆分为多个组，以便您可以在给定时间仅加载所需的资产。随着团队规模的扩大，您可以制作单独的 Unity 项目来开发不同类型的资源。这些辅助项目可以生成自己的可寻址内容构建，您可以从主项目加载这些内容（同样只需最少的代码更改）。

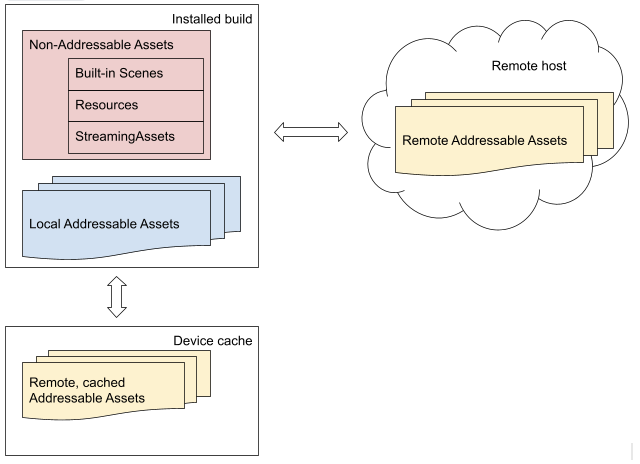
This overview discusses the following concepts to help you understand how to manage and use your assets with the Addressables system:  
本概述讨论以下概念，以帮助您了解如何通过可寻址对象系统管理和使用资产：

* **Asset address**: a string ID that identifies an Addressable asset. You can use an address as a key to load the asset.  
  **Asset address**: 标识可寻址资产的字符串 ID。您可以使用地址作为加载资产的键。
* **AssetReferences**: a type you can use to support the assignment of Addressable assets to fields in an Inspector window. You can use an AssetReference instance as a key to load the asset. The [AssetReference](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AssetReferences.html) class also provides its own loading methods.  
  **AssetReferences**: 一种可用于支持将可寻址资源分配给Inspector窗口中的字段的类型。您可以使用资产引用实例作为加载资产的键。[AssetReference](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AssetReferences.html) 类还提供了自己的加载方法。
* **Label**: a tag that you can assign to multiple assets and use to load related assets together as a group. You can use a label as a key to load the asset.  
  **Label**: 可以分配给多个资产并用于将相关资产作为一个组一起加载的标签。您可以使用标签作为键来加载资产。
* **Asset location**: a runtime object that describes how to load an asset and its dependencies. You can use a location object as a key to load the asset.  
  **Asset location**: 该运行时对象描述如何加载资产及其依赖项。您可以使用位置对象作为键来加载资产。
* **Key**: an object that identifies one ore more Addressables. Keys include addresses, labels, AssetReference instances and location objects.  
  **Key**: 标识一个或多个可寻址对象。键包括地址、标签、资产引用实例和位置对象。
* **Asset loading and unloading**: the Addressables API provides its own functions to load and release Assets at runtime.  
  **Asset loading and unloading**: 可寻址对象 API 提供自己的函数，用于在运行时加载和释放资产。
* **Dependencies**: An asset dependency is one asset used by another, such as a Prefab used in a Scene asset or a Material used in a Prefab asset.  
  **依赖关系：**资源依赖关系是一个资源被另一个资源使用，例如场景资源中使用的预制件或预制件资源中使用的材质。
* **Dependency and resource management**: the Addressables system uses reference counting to track which assets and AssetBundles are in use, including whether the system should load or unload dependencies (other referenced Assets).  
  **依赖关系和资源管理：**可寻址对象系统使用引用计数来跟踪正在使用的资产和资产包，包括系统是否应加载或卸载依赖关系（引用的其他资产）。
* **Group**: you assign assets to groups in the Editor. The group settings determine how Addressables packages the group assets into AssetBundles and how it loads them at runtime.  
  **（资产）组：**在编辑器中将资产分配给组。组设置确定可寻址对象如何将组资源打包到AssetBundles中，以及如何在运行时加载它们。
* **Content catalogs**: Addressables uses catalogs to map your assets to the resources that contain them.  
  **内容目录：**可寻址对象使用目录将资产映射到包含它们的资源。
* **Content builds**: when using Addressables, you make a content build to collate and package your assets as a separate step before you make a player build.  
  **内容构建：**使用可寻址对象时，在进行播放器构建之前，您可以单独制作内容构建以整理和打包资产。
* **Multiple platform support**: the build system separates content built by platform and resolves the correct path at runtime.  
  **多平台支持：**构建系统分离平台构建的内容，并在运行时解析正确的路径。
* **Addressables tools**: the Addressables package contains several windows and tools to organize, build, and optimize your content.  
  **可寻址对象工具：**可寻址对象包包含多个窗口和工具，用于组织、构建和优化内容。

By default, Addressables uses AssetBundles to package your assets. You can also implement your own [IResourceProvider](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceProviders.IResourceProvider.html) class to support other ways to access assets.

## Asset addresses 资产地址

A key feature of the Addressables system is that you assign addresses to your assets and use those addresses to load them at runtime. The Addressables resource manager looks up the address in the content catalog to find out where the asset is stored. (Assets can be built-in to your application, cached locally, or hosted remotely.) The resource manager loads the asset and any dependencies, downloading the content first, if necessary.  
可寻址对象系统的一个关键功能是，您可以为资产分配地址(address)，并在运行时使用这些地址加载它们。可寻址对象资源管理器在内容目录(content catalog)中查找地址(address)，以找出资产的存储位置。（资产可以内置到应用程序中、在本地缓存或远程托管。资源管理器加载资产和任何依赖项，如有必要，请先[下载内容](#_Preloading_dependencies_预加载依赖项)）。

  
Addressables loads Assets by address no matter where they're located

Because an address isn't tied to the physical location of the Asset, you have much more flexibility when managing and optimizing your Assets, both in the Unity Editor and at runtime. [Content catalogs](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsOverview.html#content-catalogs) map Addresses to physical locations.  
由于地址与资源的物理位置无关，因此在 Unity 编辑器和运行时管理和优化资源时，您都有更大的灵活性。[Content catalogs](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsOverview.html#content-catalogs)将地址映射到物理位置。

Although, you should typically assign unique addresses to your assets, an asset address is not required to be unique. You can assign the same address string to more than one asset when useful. For example, if you have variants of an asset, you could assign the same address to all the variants and use labels to distinguish between the variants:  
尽管您通常应为资产分配唯一地址，但资产地址不需要是唯一的。当有用时，您可以将相同的地址字符串分配给多个资产。例如，如果您有资产的多属性，则可以为所有多属性分配相同的地址，并使用标签来区分多属性：

* Asset 1: address: "plate\_armor\_rusty", label: "hd"
* Asset 2: address: "plate\_armor\_rusty", label: "sd"

Addressables API functions that only load a single asset, such as [LoadAssetAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetAsync.html), load the first instance found if you call them with an address assigned to multiple assets. Other functions, like [LoadAssetsAsync](xref:UnityEngine.AddressableAssets.Addressables.LoadAssetsAsync%60%601(System.Collections.Generic.IList%7bSystem.Object%7d,System.Action%7b%60%600%7d,UnityEngine.AddressableAssets.Addressables.MergeMode)), load multiple assets in one operation and load all the assets with the specified address.  
仅加载单个资产的可寻址 API 函数（如  [LoadAssetAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetAsync.html)）加载使用分配给多个资产的地址调用它们时找到的第一个实例。其他函数（如 [LoadAssetsAsync](xref:UnityEngine.AddressableAssets.Addressables.LoadAssetsAsync%60%601(System.Collections.Generic.IList%7bSystem.Object%7d,System.Action%7b%60%600%7d,UnityEngine.AddressableAssets.Addressables.MergeMode))）在一次操作中加载多个资产，并使用指定地址加载所有资产。

##### TIP

You can use the [MergeMode](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.MergeMode.html) parameter of [LoadAssetsAsync](xref:UnityEngine.AddressableAssets.Addressables.LoadAssetsAsync%60%601(System.Collections.Generic.IList%7bSystem.Object%7d,System.Action%7b%60%600%7d,UnityEngine.AddressableAssets.Addressables.MergeMode)) to load the intersection of two keys.

In the example above, you could specify the address, "plate\_armor\_rusty", and the label, "hd", as keys and intersection as the merge mode to load "Asset 1". You could change the label value to "sd" to load "Asset 2".

See [Making an asset Addressable](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsGettingStarted.html#making-an-asset-addressable) for how to assign addresses to assets.

See [Loading assets](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadingAddressableAssets.html) for how to load assets by keys, including addresses.

## AssetReference 资产引用

An AssetReference is a type that you can set to any kind of Addressable asset. Unity does not automatically load the asset assigned to the reference, so you have more control over when to load and unload it.  
资产引用是一种可以设置为任何类型的可寻址资产的类型。Unity 不会自动加载分配给引用的资源，因此您可以更好地控制何时加载和卸载它。

Use fields of type AssetReference in your MonoBehaviours and ScriptableObjects to help you specify which Addressable asset to use for that field (instead of using the string that specifies the address). AssetReferences support drag-and-drop and object picker assignment, which can make them more convenient to use in an Editor Inspector.  
在 MonoBehaviors 和 ScriptableObject 中使用 AssetReference 类型的字段来帮助您指定要用于该字段的可寻址资源（而不是使用指定地址的字符串）。资源引用支持拖放，这使得它们在编辑器检查器中使用起来更方便。

In addition to the base AssetReference type, Addressables provides a few more specialized types, such as AssetReferenceGameObject and AssetReferenceTexture. You can use these specialized subclasses to eliminate the possibility of assigning the wrong type of asset to an AssetReference field. In addition, you can use the AssetReferenceUILabelRestriction attribute to limit assignment to Assets with specific labels.  
除了基本的 AssetReference 类型之外，Addressables 还提供了一些更专业的类型，例如 AssetReferenceGameObject 和 AssetReferenceTexture。您可以使用这些专用子类来消除将错误类型的资产分配给资产引用字段的可能性。此外，还可以使用 AssetReferenceUILabelRestriction 属性来限制对具有特定标签的资产的分配。

See [Using AssetReferences](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AssetReferences.html) for more information.

## Loading and releasing assets 加载和释放资产

To load an Addressable asset, you can use its address or other key such as a label or AssetReference. See [Loading Addressable Assets](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadingAddressableAssets.html) for more information. You only need to load the main asset; Addressables loads any dependent assets automatically.  
要加载可寻址资源，您可以使用其地址或其他键，例如标签或资产引用。有关详细信息，请参阅[Loading Addressable Assets](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadingAddressableAssets.html)。您只需要加载主资产;可寻址对象会自动加载任何依赖资产。

When your application no longer needs access to an Addressable asset at runtime, you **must** release it so that Addressables can free the associated memory. The Addressables system keeps a reference count of loaded assets. It doesn't unload an asset until the reference count returns to zero. Thus, you don't need to keep track of whether an asset or its dependencies are still in use; you only need to make sure that anytime you explicitly load an asset, you release it when your application no longer needs that instance. See [Releasing Addressable assets](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadingAddressableAssets.html#releasing-addressable-assets) for more information.  
当应用程序在运行时不再需要访问可寻址资产时，**必须**释放它，以便可寻址资产可以释放关联的内存。可寻址对象系统保留已加载资产的引用计数。在引用计数恢复为零之前，它不会卸载资产。因此，您无需跟踪资产或其依赖项是否仍在使用中;您只需确保在显式加载资产时，在应用程序不再需要该实例时释放该资产。有关详细信息，请参阅释放可寻址资产。

## Dependency and resource management 依赖关系和资源管理

One asset in Unity can depend on another. A Scene might reference one or more Prefabs; a Prefab might use one or more Materials. The same Material can be used by more than one Prefab and those Prefabs can exist in different AssetBundles. When you load an Addressable asset, the system automatically finds and loads any dependent assets that it references. When the system unloads an asset, it also unloads its dependencies -- unless they're still being used by a different asset.  
Unity 中的一个资源可以依赖于另一个资源。场景可能引用一个或多个预制件;预制件可能使用一种或多种材质。同一材质可由多个预制件使用，这些预制件可以存在于不同的资源包中。加载可寻址资产时，系统会自动查找并加载它引用的任何从属资产。当系统卸载资产时，它也会卸载其依赖项 - 除非它们仍由其他资产使用。

As you load and release assets, the Addressables system keeps a reference count for each item. When an asset is no longer referenced, Addressables unloads it. If the asset was in a bundle that no longer contains any assets that are in use, Addressables also unloads the bundle.  
加载和释放资产时，可寻址对象系统会为每个项目保留引用计数。当资产不再被引用时，可寻址对象会卸载它。如果资产捆绑包的所有资源都被卸载，则可寻址对象也会卸载捆绑包。

See [Memory management](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/MemoryManagement.html) for more information.

## Addressables groups and labels 可寻址对象的groups和labels

Use Addressables groups to organize your content. All Addressable Assets belong to a group. If you don't explicitly assign an asset to a group, Addressables adds it to the default group.  
使用可寻址对象group来组织内容。所有可寻址资产都属于一个组。如果未将资产显式分配给组，则可寻址对象会将其添加到默认组。

You can set the group settings to specify how the Addressables build system should package the assets in a group into bundles. For example, you can choose whether or not all the assets in a group should be packed together in a single AssetBundle file.  
您可以通过组设置，指定可寻址对象构建系统应如何将组中的资产打包到捆绑包中。例如，您可以选择是否组内的所有资产都打包进同一个AssetBundle文件

Use labels to tag content that you want to treat together in some way. For example, if you had labels defined for "red", "hat", and "feather", you could load all red hats with feathers in a single operation, whether or not they are part of the same AssetBundle. In addition, you can use labels to determine how assets in a group are packed into bundles.  
使用标签标记要以某种方式一起处理的内容。例如，如果您为red、hat和feather定义了标签，则可以在单个操作中加载带有羽毛的所有红帽，无论它们是否属于同一 AssetBundle。此外，您还可以使用标签来确定如何将组中的资产打包到捆绑包中。

Add an asset to a group and move assets between groups using the [**Addressables Groups**](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html#groups-window) window. You can also assign labels to your assets in the Groups window.

### Group schemas 组架构

The schemas assigned to a group define the settings used to build the assets in a group. Different schemas can define different groups of settings. For example, one standard schema defines the settings for how to pack and compress your assets into AssetBundles (among other options). Another standard schema defines which of the categories, "Can Change Post Release" and "Cannot Change Post Release" the assets in the group belong to.  
分配给组的schemas用于设置在组中构建资产的方式。不同的schemas可以定义不同的设置项。例如，一个标准架构定义了如何将资产打包和压缩到资产包中的设置（以及其他选项）。另一个标准架构定义组中的资产属于哪个类别，即“可以更改发布后”和“无法更改发布后”。

You can define your own schemas to use with custom build scripts.  
可以定义自己的架构以用于自定义生成脚本。

See [Schemas](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/GroupSettings.html#schemas) for more information about group schemas.

## Content catalogs 内容目录

The Addressables system produces a content catalog file that maps the addresses of your assets to their physical locations. Addressables produces one content catalog per project, but you can load catalogs created by other Unity Projects to load Addressable assets produced by those Projects. This allows you to use separate Projects to develop and build some of your assets, which can make iteration and team collaboration easier on large productions.  
可寻址对象系统会生成一个内容目录文件，该文件将资产的地址映射到其物理位置。可寻址对象为每个项目生成一个内容目录，但您可以加载其他 Unity 项目创建的目录以加载这些项目生成的可寻址资源。这允许您使用单独的项目来开发和构建您的一些资产，这使得大型作品的迭代和团队协作更加容易。

When Addressables produces the content catalog, it also creates a hash file containing the hash (a mathematical fingerprint) of the catalog. If you are hosting your Addressable assets remotely, the system uses this hash file to determine if the content catalog has changed and needs to be downloaded.  
当可寻址对象生成内容目录时，它还会创建一个包含目录哈希（数学指纹）的哈希文件。如果要远程托管可寻址资产，系统将使用此哈希文件来确定内容目录是否已更改并需要下载。

The Profile selected when you perform a content build determines how the addresses in the content catalog map to resource loading paths. See [Profiles](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html) for more information.

See [Loading additional catalogs](#_Loading_additional_catalogs) for information about loading content catalogs.

See [Distributing content remotely](#_Remote_content_distribution) for information about hosting content remotely.

See[Managing catalogs at runtime](#_Managing_catalogs_at) if you want to load additional catalogs or override the default update behavior.

## Content builds 内容构建

The Addressables system separates the building of Addressable content from the build of your player. A content build produces the content catalog, catalog hash, and the AssetBundles containing your assets.  
可寻址内容的构建与播放器的构建分开。内容构建会生成content catalog, catalog hash和包含资产的AssetBundles。

Because asset formats are platform-specific, you must make a content build for each platform before building a player.  
由于资产格式是特定于平台的，因此在构建播放器之前，必须为每个平台进行内容构建。

See [Building Addressable content](#_Building_content)for more information.

## Play mode scripts Play mode 脚本

When you run your game or application in the Editor Play mode, it can be inconvenient and slow to always perform a content build before pressing the Play button. At the same time, you do want to be able to run your game in a state as close to a built player as possible. For flexibility, Addressables provides three options that determine how the Addressables system locates and loads assets in Play mode:  
在编辑器运行模式下运行游戏或应用程序时，在按下“运行”按钮之前始终执行内容构建可能会很不方便且速度很慢。同时，您确实希望能够在尽可能接近内置玩家的状态下运行游戏。为了提高灵活性，可寻址对象提供了三个选项，用于确定可寻址对象系统在运行模式下如何定位和加载资源：

* **Use the Asset database**: Addressables loads Assets directly from the Asset database. This option typically provides the fastest iteration speed if you are making both code and Asset changes, but also least resembles a production build.  
  **Use the Asset database**: 可寻址对象直接从资产数据库加载资产。如果同时进行代码和资产更改，此选项通常提供最快的迭代速度，但也与生产版本最不相似。
* **Simulate groups**: Addressables loads Assets while simulating groups. This option is helpful if you are working on organizing and optimizing your Addressables groups themselves. It provides Addressables events without requiring a full content rebuild after every change.  
  **Simulate groups**: 可寻址对象在模拟组时加载资源。如果您正在组织和优化可寻址对象组本身，则此选项非常有用。它提供可寻址对象事件，而无需在每次更改后重新生成完整的内容。
* **Use existing build**: Addressables loads content from your last content build. This option most resembles a production build and can provide fast iteration turnaround if you aren't changing Assets.  
  **Use existing build**: 可寻址对象从最后一次构建生成的内容中进行加载。此选项最类似于生产版本，在没有改变资源的情况下，提供快速迭代周转

See [Play mode Scripts](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html#play-mode-scripts) for more information.

## Support for multiple platforms 多平台支持

Addressables supports projects designed for multiple platforms by including the target platform name in the build path and by making sure that it copies the correct platform files to the StreamingAssets folder when you make a player build.  
Addressables 支持为多个平台设计的项目，方法是在生成路径中包含目标平台名称，并确保在生成播放器时将正确的平台文件复制到 StreamingAssets 文件夹。

## Addressables tools 可寻址对象工具

The Addressables system provides a few tools and windows to help you manage your Addressable assets:

* [Addressable Groups window](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html#groups-window): The Groups window is the main interface for managing assets, group settings, and making builds..  
  Groups 主窗口
* [Profiles window](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html): helps set up paths used by your builds.  
  Profiles 用于设置build路径
* [Addressables Event Viewer](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/EventViewer.html): monitor and profile runtime events related to your Addressable assets.  
  EventViewer 用于运行时监控
* [Analyze tool](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AnalyzeTool.html): the Analyze tool runs analysis rules that check whether your Addressables content conforms to the set of rules you have defined. The Addressables system provides some basic rules, such as checking for duplicate assets; you can add your own rules using the [AnalyzeRule](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.AnalyzeRules.AnalyzeRule.html) class.  
  分析工具
* [Hosting tool](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsHostingServices.html): the Hosting tool provides a simple asset hosting service that runs from the Unity Editor to aide development and testing of your project.  
  Hosting tool 用于模拟远程服务
* [Build layout report](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildLayoutReport.html): describes the AssetBundles produced by a content build.  
  构建报告是一个单独的文件。

# Configuring Addressables 配置可寻址对象

## Overview

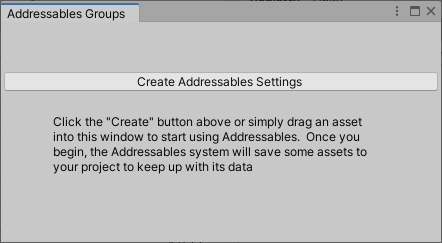
The following topics provide an overview of the configuration options for setting up the Addressables system in a project and links to more detailed information:

* [Initialization](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Configuration.html#initialization)
* [System settings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Configuration.html#system-settings)
* [Group settings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Configuration.html#group-settings)
* [Profiles](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Configuration.html#profiles)
* [Asset hosting service](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Configuration.html#asset-hosting-service)
* [Preferences](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Configuration.html#unity-preferences)
* [Additional topics](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Configuration.html#additional-topics)

### Initialization 初始化

The Addressables system uses a set of [ScriptableObject](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/ScriptableObject.html) assets to store your configuration settings. The system stores these settings assets in the Assets/AddressableAssetsData folder of your Unity project. It creates this folder and default settings objects when you initialize Addressables from the Groups window. Open the [Groups window](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html#groups-window) (menu: **Window > Asset Management > Addressables > Groups**) after installing the Addressables package.

The first time you open the [Groups window](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html#groups-window), click **Create Addressables Settings** to run the initialization command to create the settings folder and assets:

  
*Before initializing the Addressables system in a Project*

Add the AddressableAssetsData folder and its contents to your source control system.

See [Getting started](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsGettingStarted.html) for a quick guide to using the Addressable system and [Managing Addressables in the Editor](#_Managing_Addressables_in) for information on ways to organize your Addressable assets.

### System settings 系统设置

The AddressableAssetsSettings object contains the global, system settings for your Project. You can access these settings from the menu: **Window > Asset Management > Addressables > Settings** or from the **Tools** menu on the [Groups window](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html#groups-window).

See [Addressable system settings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html) for information about each setting.

### Group settings 组设置

The Addressables system uses the [Groups](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html) you define to determine how to package your Addressable assets into local and remote AsssetBundles. Each group has its own settings object that you can use to control that group's options. Addressables creates a new settings object whenever you create a group.

See [Groups](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html) and [Group settings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/GroupSettings.html) for more information.

### Profiles 变量配置

[Profiles](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html) let you configure sets of build variables as appropriate for the purpose of build. For example, you could configure a profile to use for development builds of your project, one for test builds, and another for publishing release builds. You can create as many profiles as you need.

**TIP**

When you host

See [Profiles](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html) for more information.

### Asset hosting service 模拟服务器

The Addressables system provides a asset hosting service that runs within the Unity Editor. You can use this service to test your remote content via an HTTP connection.

See [Asset hosting service](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsHostingServices.html) for more information.

### Unity Preferences Unity 首选项

The Addressables package adds its own section to the Unity Editor [Preferences](https://docs.unity3d.com/2019.4/Documentation/Manual/Preferences.html) window. The Addressables preferences include:

**Debug Build Layout**

When enabled, the build system produces the [Build layout report](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildLayoutReport.html). This option is disabled by default since it increases the time need to create a build. The build report contains a detailed description of each AssetBundle produced by the build.

See [Diagnostic tools](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/DiagnosticTools.html) for a description of this and other analysis tools.

**Build Addressables on Player Build** (Unity 2021.2+)

Determines whether Unity builds Addressables content as part of your Player build.

Building Addressables content together with the Player can be convenient, but does increase build time, especially on large projects, since this rebuilds the Addressables content even when you haven't modified any assets. If you don't change your Addressables content between most builds, consider disabling this option.

The options include:

* **Build Addressables content on Player Build**: Always build Addressables content when building the Player.
* **Do not Build Addressables content on Player Build**: Never build Addressables content when building the Player. (If you modify Addressables content, you must rebuild it manually before building the Player.)
* **Use global Settings (stored in preferences)**: Use the value specified in the Unity Editor Preferences (under **Addressables**). This option allows every project contributor to set the option as they choose.

The first two options override the global Preference for the current Project and affect all contributors who build the Project. Otherwise, the global Preference applies to all Unity projects.

See [Building Addressables content with Player builds](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Builds.html#build-with-player) for more information.

### Additional topics 附加内容

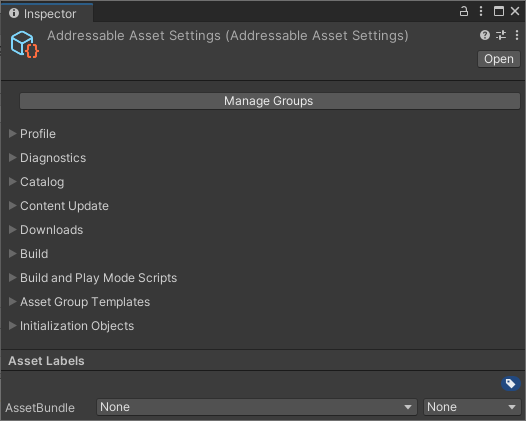
See the following topics on more involved setup options:

* [Continuous integration](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContinuousIntegration.html)
* [Build scripting](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildPlayerContent.html)
* [Customizing Addressables runtime initialization](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/InitializeAsync.html)

## Addressable Asset Settings 可寻址资产设置

You can access the main Addressable system option on the **Addressable Asset Settings** Inspector (menu: **Window > Asset Management > Addressables > Settings**).

The Addressables system stores the settings asset in the AddressableSettingsData folder (under your Project Assets folder). If this folder doesn't exist yet, you must initialize the Addressables system from the **Groups** window (menu: **Window > Asset Management > Addressables > Groups**).



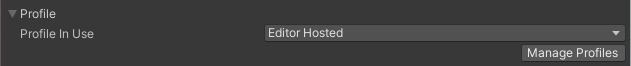
*The Addressable Asset Settings Inspector*

The Inspector contains the following sections:

* [Profile](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html#profile)
* [Diagnostics](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html#diagnostics)
* [Catalog](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html#catalog)
* [Content Update](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html#content-update)
* [Downloads](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html#downloads)
* [Build](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html#build)
* [Build and Play Mode Scripts](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html#build-and-play-mode-scripts)
* [Asset Group Templates](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html#asset-group-templates)
* [Initialization object list](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html#initialization-object-list)

You can click the **Manage Groups** button to open the [Groups window](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html#groups-window).

### Profile 变量配置

  
*Profile settings*

Use the **Profile in Use** list to choose the active profile. The active profile determines the value of variables used by the Addressables build scripts.  
使用 **Profile in Use** 列表选择当前使用的配置。当前的配置文件确定可寻址生成脚本使用的**变量**的值。

Click the **Manage Profiles** button to open the **Profiles** window where you can create new profiles and change profile variables.  
单击**Manage Profiles**按钮以打开**Profiles** 窗口，您可以在其中创建新配置文件和更改配置文件变量。

See [Profiles](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html) for more information about profiles.

### Diagnostics 诊断

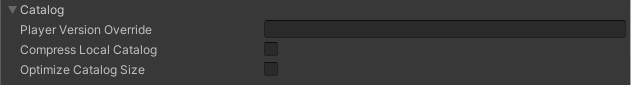
  
*Diagnostics settings*

| **Property** | **Function** |
| --- | --- |
| **Send Profiler Events** | Enables profiler events. You must enable this setting to use the Addressables [Event Viewer](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/EventViewer.html) window. 启用探查器事件。必须启用此设置才能使用[Event Viewer](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/EventViewer.html)窗口。 |
| **Log Runtime Exceptions** | Logs runtime exceptions for asset loading operations (in addition to recording the error to the [AsyncOperationHandle.OperationException](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.OperationException.html#UnityEngine_ResourceManagement_AsyncOperations_AsyncOperationHandle_OperationException) property). 记录资产加载操作的运行时异常（除了将错误记录到 [AsyncOperationHandle.OperationException](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.OperationException.html#UnityEngine_ResourceManagement_AsyncOperations_AsyncOperationHandle_OperationException) 属性之外）。 |

**TIP**

By default, Addressable Assets only logs warnings and errors. You can enable detailed logging by opening the **Player** settings window (menu: **Edit** > **Project Settings...** > **Player**), navigating to the **Other Settings** > **Configuration** section, and adding "ADDRESSABLES\_LOG\_ALL" to the **Scripting Define Symbols** field.  
默认情况下，可寻址资产仅记录警告和错误。您可以通过打开“播放器设置”窗口（菜单：**Edit** > **Project Settings...** > **Player**），导航到**Other Settings** > **Configuration**部分，然后将"ADDRESSABLES\_LOG\_ALL"添加到**Scripting Define Symbols** 字段来启用详细日志记录。

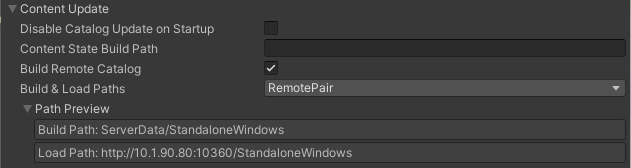
### Catalog 目录

  
*Catalog settings*

Settings related to the Addressables Catalog, which maps the address of an asset to its physical location.

| **Property** | **Function** |
| --- | --- |
| **Player Version Override** | Overrides the timestamp used to formulate the remote catalog name. If set, the remote catalog is named, Catalog\_<Player Version Override>.json. If left blank, then the timestamp is used. Note that when you use a unique remote catalog name for every new build, you can host multiple versions of your content at the same base URL. If you use the same override string for every build, then all players will load the new catalog. Note also that player update builds always use the same remote catalog name as the build they are updating (see [Content update builds](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html)). 覆盖用于制定远程目录名称的时间戳。如果设置，远程目录将命名为 Catalog\_<Player Version Override>.json。如果留空，则使用时间戳。请注意，当您为每个新版本使用唯一的远程目录名称时，您可以在同一基本 URL 上托管内容的多个版本。如果对每个构建使用相同的覆盖字符串，则所有玩家都将加载新目录。另请注意，玩家更新版本始终使用与其要更新的版本相同的远程目录名称（请参阅[Content update builds](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html)）。 |
| **Compress Local Catalog** | Builds the catalog in a compressed AssetBundle file.Reduces the storage size of the catalog, but increases the time to build and to load the catalog. |
| *Optimize Catalog Size* | Reduces the size of the catalog by creating a lookup table for internal IDs. Can increase the time required to load the catalog. |

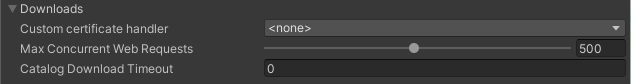
### Content Update 内容更新

  
*Content update settings*

Settings that control remote content builds and updates.

| **Property** | **Function** |
| --- | --- |
| **Disable Catalog Update on Startup** | Disables the automatic check for an updated remote catalog when the Addressables system initializes at runtime. You can manually [check for an updated catalog.](#_Updating_catalogs_更新目录) 禁用在运行时初始化可寻址对象系统时对更新的远程目录的自动检查。您可以手动检查更新的目录。 |
| **Content State Build Path** | Where to build the content state file produced by the default build script. |
| **Build Remote Catalog** | Enable to build a remote catalog. |
| **Build & Load Paths** | Where to build and load the remote catalog. Choose a [Profile](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html) path pair from the list or select <custom> if you want to set the build and load paths separately. Only visible when you enable **Build Remote Catalog**. |
| **Build Path** | Where to build the remote catalog. Typically, you should use the *RemoteBuildPath* [Profile](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html) variable. Only shown if you set **Build & Load Paths** to <custom>. |
| **Load Path** | The URL at which to access the remote catalog. Typically, you should use the *RemoteLoadPath* [Profile](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html) variable. Only shown if you set **Build & Load Paths** to <custom>. |

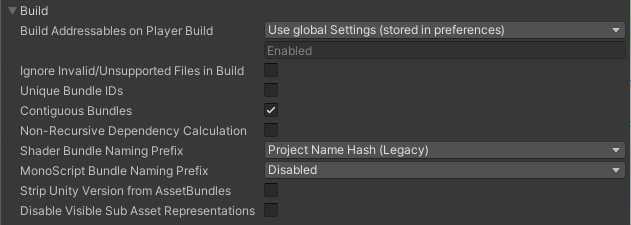
### Downloads 下载

  
*Download settings*

Settings that affect catalog and AssetBundle download handling.

| **Property** | **Function** |
| --- | --- |
| **Custom certificate handler** | The class to use for custom certificate handling. The list contains all classes in the project that extend [UnityEngine.Networking.CertificateHandler](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Networking.CertificateHandler.html). |
| **Max Concurrent Web Requests** | The system queues any requests beyond this limit. |
| **Catalog Download Timeout** | How many seconds to wait for a catalog file to download. |

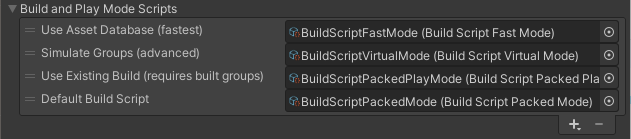
### Build 构建

  
*Build settings*

Settings that affect all builds.

| **Property** | **Function** |
| --- | --- |
| **Build Addressables on Player Build** | Whether Unity builds Addressables content as part of your Player build. • **Build Addressables content on Player Build**: Always build Addressables content when building the Player. • **Do not Build Addressables content on Player Build**: Never build Addressables content when building the Player. (If you modify Addressables content, you must rebuild it manually before building the Player.) • **Use global Settings (stored in preferences)**: Use the value specified in the [Unity Editor Preferences](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Configuration.html#unity-preferences) (under **Addressables**).  The first two options override the global Preference for the current Project and affect all contributors who build the Project. Otherwise, the global, Preferences value applies to all Unity projects. See [Building content](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Builds.html#build-with-player) for more information. |
| **Ignore Invalid/Unsupported Files in Build** | If enabled, the Addressables build script excludes invalid or unsupported files rather than aborting the build. |
| **Unique Bundle IDs** | Whether to produce a unique name for a bundle in every build. See [Unique Bundle IDs](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html#unique-bundle-ids-setting) for more information. |
| **Contiguous Bundles** 连续捆绑 | Produces a more efficient bundle layout. If you have bundles produced by Addressables 1.12.1 or earlier, disable this option to minimize bundle changes. |
| **Non-Recursive Dependency Calculation** 非递归依赖关系计算 | Enable this option to improve build times and reduce runtime memory overhead when assets have circular dependencies. Examples: • A prefab assigned to Bundle A references a material assigned to Bundle B. If this option is disabled, Unity needs to calculate the material's dependencies twice, once for each bundle. If this option is enabled, Unity only needs to calculate the material's dependencies once, for Bundle B. • Many scenes reference the same material. If this option is disabled, Unity opens each scene to calculate shader usage, which is a costly operation. If this option is enabled, Unity only loads the material and doesn't need to open any scenes for dependency calculation.  This option is enabled by default when using Unity version 2021.2 or later. Disabling this option invalidates previously built bundles because the rebuilt bundles will have a different build layout. Therefore this option should remain enabled unless builds have been shipped.  Some circular dependencies can fail to load when the option is enabled because the referenced asset is always assigned to the same bundle location, even when more content is added to the build. This issue often occurs for Monoscripts. Building the MonoScript bundle (see **MonoScript Bundle Naming Prefix**) can help resolve these load failures. |
| **Shader Bundle Naming Prefix** | How to name the bundle produced for Unity shaders. |
| **MonoScript Bundle Naming Prefix** | How to name the bundle containing all MonoScripts. The bundle ensures that Unity loads all Monoscripts before any MonoBehaviors can reference them. It also decreases the number of duplicated or complex Monoscript dependencies and so, reduces runtime memory overhead. |
| **Strip Unity Version From AssetBundles** | Whether to remove the Unity version from the bundle header. |
| **Disable Visible Sub Asset Representations** | Enable this option to improve build times if you do not use subobjects directly (Sprites, sub-meshes, etc.). |

### Build and Play Mode Scripts 构建和播放模式脚本

  
*Configured build and Play mode scripts*

Configures the [IDataBuilder](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.IDataBuilder.html) scripts available in the project. If you create a custom Build or Play Mode script, you must add it to this list before you can use it.

The Addressables packages contains a few build scripts that handle the default build processes and provide different ways to access your data in Play mode. You can find these scripts in the AddressableAssetsData/DataBuilders folder.

To add a custom script, click the **+** button and locate your script in the project. The custom script must extend [BuildScriptBase](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.DataBuilders.BuildScriptBase.html) or implement [IDataBuilder](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.IDataBuilder.html).

**NOTE**

Build scripts and Play Mode scripts both implement [**IDataBuilder**](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.IDataBuilder.html). The system distinguishes between them by the data type of the result they produce. A build script produces an [**AddressablesPlayerBuildResult**](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.AddressablesPlayerBuildResult.html), while a Play Mode script produces an [**AddressablesPlayModeBuildResult**](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.AddressablesPlayModeBuildResult.html). In a custom script, implement the [**CanBuildData**](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.IDataBuilder.CanBuildData.html) method.

### Asset Group Templates 资产组模板

  
*Configured group templates*

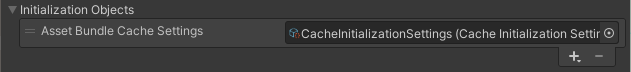
Defines the list of templates that you can use to create new groups. When you create a new template, you must add it to this list before you can use it.

The Addressables package contains one template that includes the schemas used by the default build scripts. You can find the template in the AddressableAssetsData/AssetGroupTemplates folder.

To add a custom template, click the **+** button and locate your custom template asset in your project.

See [Group templates](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/GroupSettings.html#group-templates) for information on creating custom templates.

### Initialization object list 初始化对象列表

  
*Configured InitializationObjects*

Configures the initialization objects for the project. Initialization objects are ScriptableObject classes that implement the [IObjectInitializationDataProvider](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.Util.IObjectInitializationDataProvider.html) interface. You can create these objects to pass data to the Addressables initialization process at runtime.

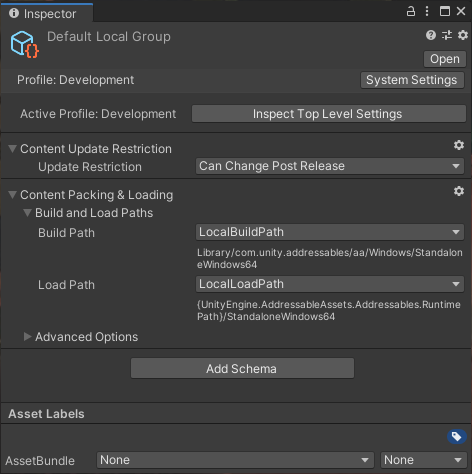
To add an initialization object, click the **+** button and locate your initialization object asset in the project.

See [Customizing initialization](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/InitializeAsync.html) for more information.

## Group settings 组设置

Group settings determine how the assets in a group are treated in content builds. For example, you can specify where AssetBundles are built, bundle compression settings, and so on.  
组设置确定如何在内容构建中处理组中的资产。例如，您可以指定资产包的构建位置、捆绑包压缩设置等。

A group's settings are declared in [Schema](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/GroupSettings.html#schemas) objects attached to the group. When you create a group with the **Packed Assets** [template](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/GroupSettings.html#group-templates), the **Content Packing & Loading** and **Content Update Restriction** schemas define the settings for the group. These settings are expected by the default [Build scripts](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Builds.html#build-commands).  
组的设置在附加到组的[Schema](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/GroupSettings.html#schemas)对象中声明。使用**Packed Assets** [template](#_Group_templates)创建组时， **内容打包及加载**和**内容更新限制**这两个Schema将定义组的设置。默认[Build scripts](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Builds.html#build-commands) 需要这些设置。

  
The Inspector window for the Default Local Group

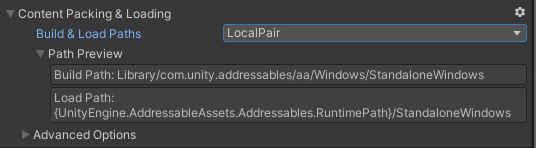
##### NOTE

If you create a group with the **Blank** template, then no schemas are attached to the group. Assets in such a group cannot be processed by the default build scripts.

### Content Packing & Loading settings 内容打包和加载设置

#### Build and Load Paths 构建和加载路径

The Build and Load Paths settings of the Content Packing & Loading schema determine where the artifacts for your content builds are created and where the Addressables system should look for them at runtime.

  
Building and loading paths

| **Setting** | **Purpose** |
| --- | --- |
| **Build & Load Paths** | The Profile path pair that defines where the Addressables build system creates artifacts for this group and where the Addressables system loads those artifacts at runtime. Choose a path pair from the list or select <custom> if you want to set the build and load paths separately. |
| **Build Path** | A Profile variable that defines where the Addressables build system creates artifacts for this group. You can also set a custom string. Use one of the following for the build path: - **LocalBuildPath**: use for assets that you plan to distribute as part of your application installation. - **RemoteBuildPath**: use for assets that you plan to distribute using a remote hosting service such Unity Cloud Content Delivery or other Content Delivery Network. - **<custom>**: specify a string as the build path for this group. Only shown if you set **Build & Load Paths** to <custom>. |
| **Load Path** | A Profile variable that defines where the Addressables system loads the build artifacts for this group at runtime. You can also set a custom string. Use one of the following for the load path: - **LocalLoadPath**: use for assets that you plan to distribute as part of your application installation. - **RemoteLoadPath**: use for assets that you plan to distribute using a remote hosting service such Unity Cloud Content Delivery or other Content Delivery Network. - **<custom>**: specify a string as the load path for this group. Only shown if you set **Build & Load Paths** to <custom>. |

The build and load path options are defined by variables in your [Profiles](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html). Note that only variables intended for a given purpose should be used for a setting. For example, choosing a load path variable for a build path setting wouldn't give you a useful result.  
构建和加载路径选项由[Profiles](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html)中的变量定义。请注意，只有用于给定目的的变量才应用于设置。例如，为生成路径设置选择加载路径变量不会提供有用的结果。

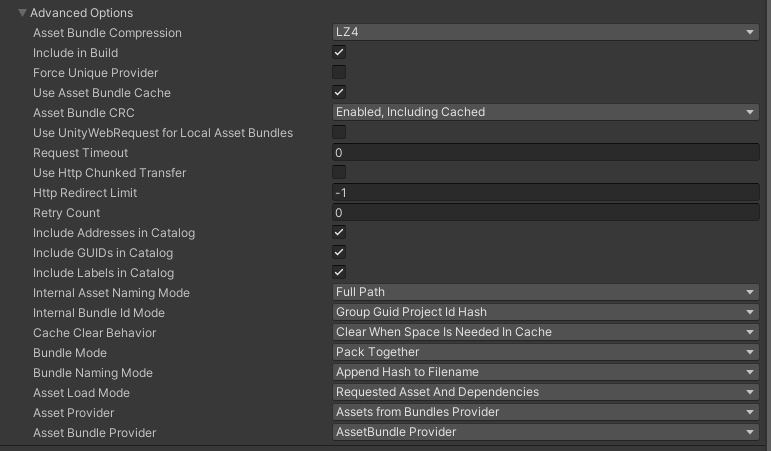
When you choose a Profile variable, the current evaluation of the path is shown in the **Path Preview**. Components of the path in braces, such as {UnityEngine.AddressableAssets.Addressable.RuntimePath}, indicate that static variable is used to construct the final path at runtime. That portion of the path is replaced by the current value of the static variable when the Addressables system initializes at runtime.  
选择配置文件变量时，路径的当前评估将显示在路径预览中。大括号中路径的组件（例如 {UnityEngine.AddressableAssets.Addressable.RuntimePath}）表示静态变量用于在运行时构造最终路径。当可寻址系统在运行时初始化时，路径的该部分将替换为静态变量的当前值。

##### WARNING

In most cases, you should not change the local build or load paths from their default values. If you do, you must copy the local build artifacts from your custom build location to the project's [StreamingAssets] folder before making a Player build. Altering these paths also precludes building your Addressables as part of the Player build.  
在大多数情况下，不应更改本地生成或加载路径的默认值。如果这样做，则必须在进行播放器构建之前将本地构建工件从自定义构建位置复制到项目的 [StreamingAssets] 文件夹。更改这些路径还排除了将可寻址对象构建为播放器构建的一部分。

See [Profiles](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html) for more information.

#### Advanced Options 高级选项

  
The Advanced Options section

| **Setting** | **Purpose** |
| --- | --- |
| **Asset Bundle Compression** | The compression type for all bundles produced from the group. LZ4 is usually the most efficient option, but other options can be better in specific circumstances. See [AssetBundle Compression](https://docs.unity3d.com/2019.4/Documentation/Manual/AssetBundles-Cache.html) for more information. |
| **Include In Build** | Whether to include assets in this group in a content build. |
| **Force Unique Provider** | Whether Addressables uses unique instances of Resource Provider classes for this group. Enable this option if you have custom Provider implementations for the asset types in this group and instances of those Providers must not be shared between groups. |
| **Use Asset Bundle Cache** | Whether to cache remotely distributed bundles. |
| **Asset Bundle CRC** | Whether to verify a bundle's integrity before loading it. • **Disabled**: Never check bundle integrity. • **Enabled, Including Cached**: Always check bundle integrity. • **Enabled, Excluding Cached**: Check integrity of bundles when downloading. |
| **Use UnityWebRequest for Local Asset Bundles** | Load local AssetBundle archives from this group using [UnityWebRequestAssetBundle.GetAssetBundle](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Networking.UnityWebRequest.GetAssetBundle.html) instead of [AssetBundle.LoadFromFileAsync](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/AssetBundle.LoadFromFileAsync.html). |
| **Request Timeout** | The timeout interval for downloading remote bundles. |
| **Use Http Chunked Transfer** | Whether to use the HTTP/1.1 chunked-transfer encoding method when downloading bundles.  Deprecated and ignored in Unity 2019.3+. |
| **Http Redirect Limit** | The number of redirects allowed when downloading bundles. Set to -1 for no limit. |
| **Retry Count** | The number of times to retry failed downloads. |
| **Include Addresses in Catalog** | Whether to include the address strings in the catalog. If you don't load assets in the group using their address strings, you can decrease the size of the catalog by not including them. |
| **Include GUIDs in Catalog** | Whether to include GUID strings in the catalog. You must include GUID strings to access an asset with an [AssetReference](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AssetReferences.html). If you don't load assets in the group using AssetReferences or GUID strings, you can decrease the size of the catalog by not including them. 是否在目录中包含 GUID 字符串。必须包含 GUID 字符串才能使用 [AssetReference](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AssetReferences.html)访问资产。如果不使用 [AssetReference](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AssetReferences.html)或 GUID 字符串加载组中的资产，则可以通过不包含它们来减小目录的大小。 |
| **Include Labels in Catalog** | Whether to include label strings in the catalog. If you don't load assets in the group using labels, you can decrease the size of the catalog by not including them. |
| **Internal Asset Naming Mode** | How to name assets in the catalog internally: - **Full Path**: the assets full path in the Project - **Filename**: the asset's filename - **GUID**: the assets GUID string - **Dynamic**: the Addressables system chooses the smallest internal named based on the assets in the group |
| **Internal Bundle Id Mode** | Determines how to construct the internal Id of an AssetBundle. For example, when you set the **Group Guid** option, Addressables creates the bundle Id by combining the group name with a bundle GUID string. |
| **Cache Clear Behavior** | Determines when an installed application clears AssetBundles from the cache. |
| [Bundle Mode](#_Packing_groups_into) | How to pack the assets in this group into bundles: - **Pack Together**: create a single bundle containing all assets. - **Pack Separately**: create a bundle for each primary asset in the group. Subassets, such as Sprites in a Sprite sheet are packed together. Assets within a folder added to the group are also packed together. - **Pack Together by Label**: create a bundle for assets sharing the same combination of labels. |
| **Bundle Naming Mode** | How to construct the file names of AssetBundles. |
| **Asset Load Mode** | Whether to load assets individually as you request them (the default) or always load all assets in the group together. |
| **Asset Provider** | Defines which Provider class Addressables uses to load assets from the AssetBundles generated from this group. Set this option to **Assets from Bundles Provider** unless you have a custom Provider implementation to provide assets from an AssetBundle. |
| **Asset Bundle Provider** | Defines which Provider class Addressables uses to load AssetBundles generated from this group. Set this option to **AssetBundle Provider** unless you have a custom Provider implementation to provide AssetBundles. |

### Content Update Restriction 内容更新限制

The Content Update Restriction options determine how the [Check for Content Update Restrictions] tool treats assets in the group. Run this tool to prepare your groups for a differential content update build (rather than a full content build). The tool moves modified assets in any groups set to **Cannot Change Post Release** to a new group.  
Content Update Restriction选项确定 [Check for Content Update Restrictions] 工具如何处理组中的资产。运行此工具以准备组进行差异内容更新生成（而不是完整内容生成）。该工具会将设置为 **Cannot Change Post Release**的任何组中已修改的资源移动到新组。

The **Update Restriction** options include:

* **Can Change Post Release**: No assets are moved by the tool. If any assets in the bundle have changed, then the entire bundle is rebuilt.  
  **发布后可以更改：**该工具不会移动任何资产。如果捆绑包中的任何资产已更改，则会重建整个捆绑包。
* **Cannot Change Post Release**: If any assets in the bundle have changed, then the [Check for Content Update Restrictions] tool moves them to a new group created for the update. When you make the update build, the assets in the AssetBundles created from this new group override the versions found in the existing bundles.  
  **发布后无法更改：**如果捆绑包中的任何资源已更改，则[Check for Content Update Restrictions]工具会将其移动到为更新创建的新组中。进行更新版本时，从此新组创建的 AssetBundle 中的资产将覆盖现有捆绑包中的版本。

See [Content update builds](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html) for more information.

### Group templates 组模板

A Group template defines which types of schema objects are created for a new group. The Addressables system includes the **Packed Assets** template, which includes all the settings needed to build and load Addressables using the default build scripts.  
Group模板定义创建新Group时使用哪些Schema。可寻址对象系统包括 **Packed Assets** 模板，其中包括使用默认构建脚本构建和加载可寻址对象所需的所有设置。

If you create your own build scripts or utilities that need additional settings you can define these settings in your own schema objects and create your own group templates:

1. Navigate to the desired location in your Assets folder using the Project panel.
2. Create a Blank Group Template (menu: **Assets > Addressables > Group Templates > Blank Group Templates**).
3. Assign a suitable name to the template.
4. In the Inspector window, add a description, if desired.
5. Click the **Add Schema** button and choose from the list of schemas.
6. Continue adding schemas until all required schemas are added to the list.

##### NOTE

If you use the default build script, a group must use the **Content Packing & Loading** schema. If you use content update builds, a group must include the **Content Update Restrictions** schema. See [Builds](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Builds.html) for more information.

### Schemas 数据组

A group schema is a ScriptableObject that defines a collection of settings for an Addressables group. You can assign any number of schemas to a group. The Addressables system defines a number of schemas for its own purposes. You can also create custom schemas to support your own build scripts and utilities.  
组架构是一个ScriptableObject，用于定义可寻址组的设置项。您可以将任意数量的架构分配给一个组。可寻址对象系统为自己的目的定义了许多架构。您还可以创建自定义架构以支持您自己的构建脚本和实用程序。

The built-in schemas include:

* **Content Packing & Loading**: this is the main Addressables schema used by the default build script and defines the settings for building and loading Addressable assets.
* **Content Update Restrictions**: defines settings for making differential updates of a previous build. See [Builds](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Builds.html) for more information about update builds.
* **Resources and Built In Scenes**: a special-purpose schema defining settings for which types of built-in assets to display in the **Built In Data** group.

#### Defining custom schemas

To create your own schema, extend the [AddressableAssetGroupSchema](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetGroupSchema.html) class (which is a kind of ScriptableObject).

using UnityEditor.AddressableAssets.Settings;

public class \_\_CustomSchema \_\_: AddressableAssetGroupSchema

{

public string CustomDescription;

}

Once you have defined your custom schema object, you can add it to existing groups and group templates using the Add Schema buttons found on the Inspector windows of those entities.  
定义自定义架构对象后，可以使用这些实体的“检查器”窗口中的“添加架构”按钮将其添加到现有组和组模板中。

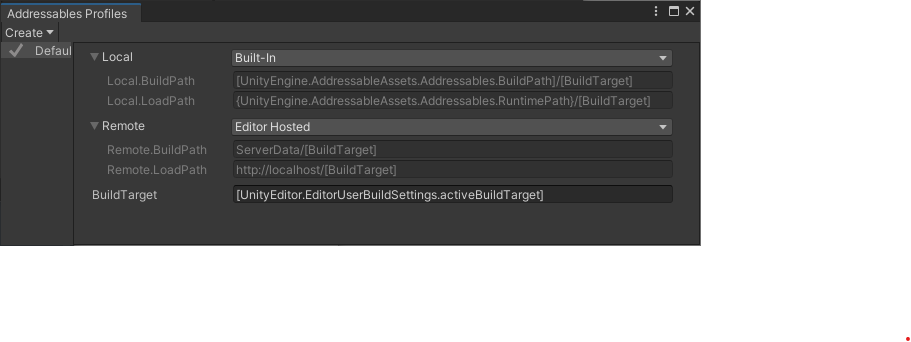
You might also want to create a custom Editor script to help users interact with your custom settings. See [Custom Inspector scripts](https://docs.unity3d.com/2019.4/Documentation/Manual/VariablesAndTheInspector.html).

In a build script, you can access the schema settings for a group using its [AddressableAssetGroup](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetGroup.html) object.

## Profiles 变量配置

A profile contains a set of variables used by the Addressables build scripts. These variables define information such as where to save build artifacts and where to load data at runtime. You can add custom profile variables to use in your own build scripts.  
配置文件包含可寻址构建脚本使用的一组变量。这些变量定义诸如在何处保存生成工件以及在运行时在何处加载数据等信息。您可以添加自定义配置文件变量以在您自己的构建脚本中使用。

Open the Profiles window (menu: **Window > Asset Management > Addressables > Profiles**) to edit profile values and create new profiles.

  
The ***Addressables Profiles*** window showing the default profile.

You can set up different profiles for the different phases or tasks in your development process. For example, you could create one profile to use while developing your project, one to use for testing, and one to use for final publishing or release. Setting up the profiles in advance and swapping between them is much less error prone than editing the values individually when you move to a different phase or perform a different task.  
您可以为开发过程中的不同阶段或任务设置不同的配置文件。例如，您可以创建一个配置文件以在开发项目时使用，一个用于测试，另一个用于最终发布。提前设置配置文件并在它们之间交换比在不同阶段或执行不同任务时单独编辑值更不容易出错。

Right-click a profile name to set it as the active profile, rename the profile, or delete it.

Addressables defines five profile variables by default:

* **Local**: defines two path variables for local content:
  + **Local.BuildPath**: where to build the files containing assets you want to install locally with your application. By default, this path is inside your Project Library folder.
  + **Local.LoadPath**: where to load assets installed locally with your application. By default, this path is in the StreamingAssets folder. Addressables automatically includes local content built to the default location in StreamingAssets when you build a Player (but not from other locations).
* **Remote**: defines two path variables for remote content:
  + **Remote.BuildPath**: where to build the files containing assets you plan to distribute remotely.
  + **Remote.LoadPath**: the URL from which to download remote content and catalogs.
* **BuildTarget**: the name of the build target, such as Android or StandaloneWindows64

You can choose the following pre-defined **Bundle Locations** for the **Local** and **Remote** path variables:

* **Built-In**: the path definitions for local content. The build system automatically includes content built with this setting in your Player builds. You should not change these path values.
* **Editor Hosted**: the path definitions to use with the Editor [Hosting service]. Depending on how you set up the hosting service, you might need to edit the load path to match the service URL.
* **Cloud Content Delivery**: the path definitions for the **Unity Cloud Content Delivery** (CCD) service. Requires creating a Unity Project ID in the **Services** section of your Project settings (or linking to an existing ID). You must also install the **CCD Management SDK** package. See [Addressable Asset system with Cloud Content Delivery](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressablesCCD.html) for information about setting up and using the **Cloud Content Delivey** **Bundle Location** option.
* **Custom**: Allows you to edit the values used for the build and load paths. See [Profile variable syntax](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html#profile-variable-syntax) for information about setting a variable value using placeholders that Addressables evaluates at build-time and runtime.

WARNING

In most cases, you should not change the local build or load paths from their default values. If you do, you must manually copy the local build artifacts from your custom build location to the project's [StreamingAssets](https://docs.unity3d.com/2019.4/Documentation/Manual/SpecialFolders.html) folder before making a Player build. Altering these paths also precludes building your Addressables as part of the Player build.  
在大多数情况下，不应更改本地生成或加载路径的默认值。如果这样做，则必须在进行播放器生成之前，手动将本地生成项目从自定义生成位置复制到项目的[StreamingAssets](https://docs.unity3d.com/2019.4/Documentation/Manual/SpecialFolders.html)文件夹。更改这些路径还排除了将可寻址对象构建为播放器构建的一部分。

See [Builds](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Builds.html) for more information about how Addressables uses profiles during content builds.

##### TIP

Using multiple profiles is most helpful when you distribute content for your application remotely. If you distribute all content as part of your application install, then the single, default profile might be the only profile you need.

### Setting the active profile 设置当前的配置文件

The active profile determines the set of variables used when you run a build script.

To set the active profile, either:

1. Open the Groups window (menu: **Window > Asset Management > Addressables > Groups**).
2. Click the **Profile** menu in the toolbar.
3. Choose the profile to make active.

Or:

1. Open the Profiles window (menu: **Window > Asset Management > Addressables > Profiles**).
2. Right- or cmd-click on a profile to open the context menu.
3. Choose **Set Active**.

##### NOTE

Build scripts include the [Play Mode Scripts](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html#play-mode-scripts) that Addressables invokes when you enter Play mode in the Editor. Some Play Mode scripts use variables from the active profile to locate content. See [Play Mode Scripts](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html#play-mode-scripts) for more information.  
构建脚本包括可寻址对象在编辑器中进入运行模式时调用的[Play Mode Scripts](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html#play-mode-scripts)。某些Play Mode scripts使用当前配置文件中的变量来查找内容。有关详细信息，请参阅[Play Mode Scripts](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html#play-mode-scripts)。

### Adding a new profile 添加新配置文件

To create a new profile, select **Create** > **Profile**. A new profile row appears in the table.

Every profile must define a value for every variable. When you create a new profile, Addressables copies all values from the currently selected profile.  
每个配置文件必须为每个变量定义一个值。创建新配置文件时，可寻址对象会复制当前所选配置文件中的所有值。

### Adding a new variable 添加新变量

You can add two kinds of variables to your profiles:

* A basic variable defines a single value
* A path pair defines a set of two path values; one for the build path and one for the load path

You can use basic variables as components of your path values (**BuildTarget** is an example of this) and you can use them in your own build scripts. Use path pair variables to set the **Build & Local Paths** setting of your Addressables [Groups] and [remote catalog](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html#content-update).

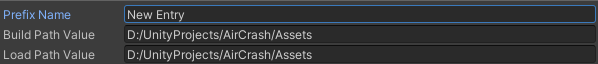
To add a new Profile variable, select either **Variable** or **Build Load Path Variable** from the **Create** menu. At the prompt, give the new variable a name and value, then click **Save**. Addressables adds the new variable to all profiles.  
要添加新的配置文件变量，请从“创建”菜单中选择**Variable** 或**Build Load Path Variable**。出现提示时，为新变量指定名称和值，然后单击保存。可寻址对象将新变量**添加到所有配置文件**。

Right-click the variable name to rename or delete the variable.

#### Path Pairs 路径对

Path pairs define a matched set of BuildPath and LoadPath variables. When you create a path pair you can use the pair name to assign the [path setting](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/GroupSettings.html#build-and-load-paths) of an Addressable [Group](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html) or remote catalog as a unit.

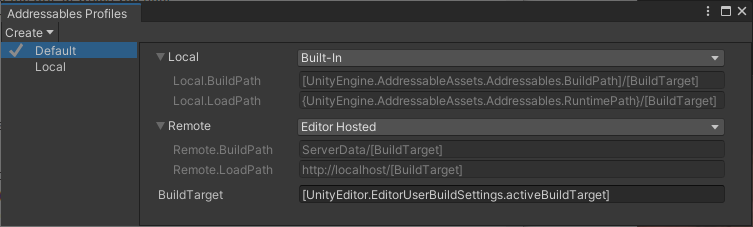
To create a path pair, select **Create** > **Build Load Path Variables**. At the prompt, give the path pair a prefix name and assign path strings to the individual fields.

  
A new path pair

The new path pair uses the **Custom** **Bundle Location** setting with your initial values. You can change to a different **Bundle Location** if needed.

##### TIP

You can "convert" two regular variables for the build and load paths into a path pair by renaming them in the Profile window. Set one to VariableName.BuildPath and the other to VariableName.LoadPath.

  
The ***Addressables Profiles*** window showing two profiles with two path pairs.

#### Default path values 默认路径值

The default values for the build and load paths are:

* Local build path: [UnityEditor.EditorUserBuildSettings.activeBuildTarget]
* Local load path: [UnityEngine.AddressableAssets.Addressables.BuildPath]/[BuildTarget]
* Remote build path: ServerData/[BuildTarget]
* Remote load path: http://localhost/[BuildTarget]

In most cases, you should not change the local path values. The Unity build system expects the AssetBundles and other files to exist in the default location. If you change the local paths, you must copy the files from the build path to the load path before making your player build. The load path must always be within the Unity StreamingAssets folder.  
在大多数情况下，不应更改本地路径值。Unity 构建系统期望资源包和其他文件存在于默认位置。如果更改本地路径，则必须在生成播放器之前将文件从生成路径复制到加载路径。加载路径必须始终位于 Unity StreamingAssets文件夹中。

If you distribute content remotely, you must modify the the remote load path to reflect the URL at which you host your remote content. You can set the remote build path to any convenient location; the build system does not rely on the default value.  
如果远程分发内容，则必须修改远程加载路径以反映承载远程内容的 URL。您可以将远程构建路径设置为任何方便的位置；build系统不依赖于默认值。

### Profile variable syntax 配置文件的变量

All variables are of type "string". You can type in a fixed path or value. You can also use two syntax designations to derive the all or part of a variable's value from static properties or other variables:  
所有变量的类型均为string。您可以键入固定路径或值。还可以使用两个语法指定从静态属性或其他变量派生变量的全部或部分值：

* **Square brackets [ ]**: Addressables evaluates entries surrounded by square brackets at build time. The entries can be other profile variables (such as [BuildTarget]) or code variables (such as [UnityEditor.EditorUserBuildSettings.activeBuildTarget](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/EditorUserBuildSettings-activeBuildTarget.html)). During a build, as it process your groups, Addressables evaluates the strings inside square brackets and writes the result into the catalog.  
  方括号**[ ]**：可寻址对象在生成时评估用方括号括起来的条目。这些条目可以是其他配置文件变量（如 [BuildTarget]）或代码变量（如  [UnityEditor.EditorUserBuildSettings.activeBuildTarget](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/EditorUserBuildSettings-activeBuildTarget.html)）。在生成过程中，当 Addressables 处理groups时，它会计算方括号内的字符串，并将结果写入目录。
* **Curly brackets { }**: Addressables evaluates entries surrounded by curly brackets at runtime. You can use code variables of runtime classes (such as {UnityEngine.AddressableAssets.Addressables.RuntimePath}).  
  大括号**{ }**：可寻址对象在运行时计算用大括号括起来的条目。您可以使用运行时类的代码变量（例如 {UnityEngine.AddressableAssets.Addressables.RuntimePath}）。

You can use static fields and properties inside the square and curly brackets. The names must be fully qualified and the types must be valid in context. For example, classes in the UnityEditor namespace can't be used at runtime.  
您可以在方括号和大括号内使用静态字段和属性。名称必须是完全限定的，类型必须在上下文中有效。例如，UnityEditor 命名空间中的类不能在运行时使用。

The code variables used in the default Profile variable settings include:  
默认配置文件变量设置中使用的代码变量包括：

* [UnityEditor.EditorUserBuildSettings.activeBuildTarget](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/EditorUserBuildSettings-activeBuildTarget.html)
* [UnityEngine.AddressableAssets.Addressables.BuildPath](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.BuildPath.html#UnityEngine_AddressableAssets_Addressables_BuildPath)
* [UnityEngine.AddressableAssets.Addressables.RuntimePath](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.RuntimePath.html#UnityEngine_AddressableAssets_Addressables_RuntimePath)

As an example, suppose you have a load path of: {MyNamespace.MyClass.MyURL}/content/[BuildTarget] set on a group that creates an AssetBundle called "trees.bundle". During the build, the catalog registers the load path for that bundle as {MyNamespace.MyClass.MyURL}/content/Android/trees.bundle, evaluating [BuildTarget] as "Android" and adding the AssetBundle name to the path. At runtime, as the catalog is processed, the Addressables system evaluates {MyNamespace.MyClass.MyURL} to produce the final load path, http://myinternet.com/content/Android/trees.bundle.

##### NOTE

Referencing a runtime variable in a Profile string does not prevent Unity from stripping that variable from your application’s runtime libraries during its build optimization phase (if nothing else in your code references the same variable). See [Managed code stripping](https://docs.unity3d.com/2019.4/Documentation/Manual/ManagedCodeStripping.html) for more information on how to prevent a class or member from being stripped.  
在配置文件字符串中引用运行时变量不会阻止 Unity 在其构建优化阶段从应用程序的运行时库中删除该变量（如果代码中没有其他变量引用同一变量）。有关如何防止类或成员被剥离的更多信息，请参见 [Managed code stripping](https://docs.unity3d.com/2019.4/Documentation/Manual/ManagedCodeStripping.html) 。

### Specifying packing and loading paths 指定打包和加载路径

Once you set up the necessary variables in your profile, you can select the build and load paths for an asset group based on those specified variables.  
在配置文件中设置必要的变量后，您可以根据这些指定的变量为资产组选择构建和加载路径。

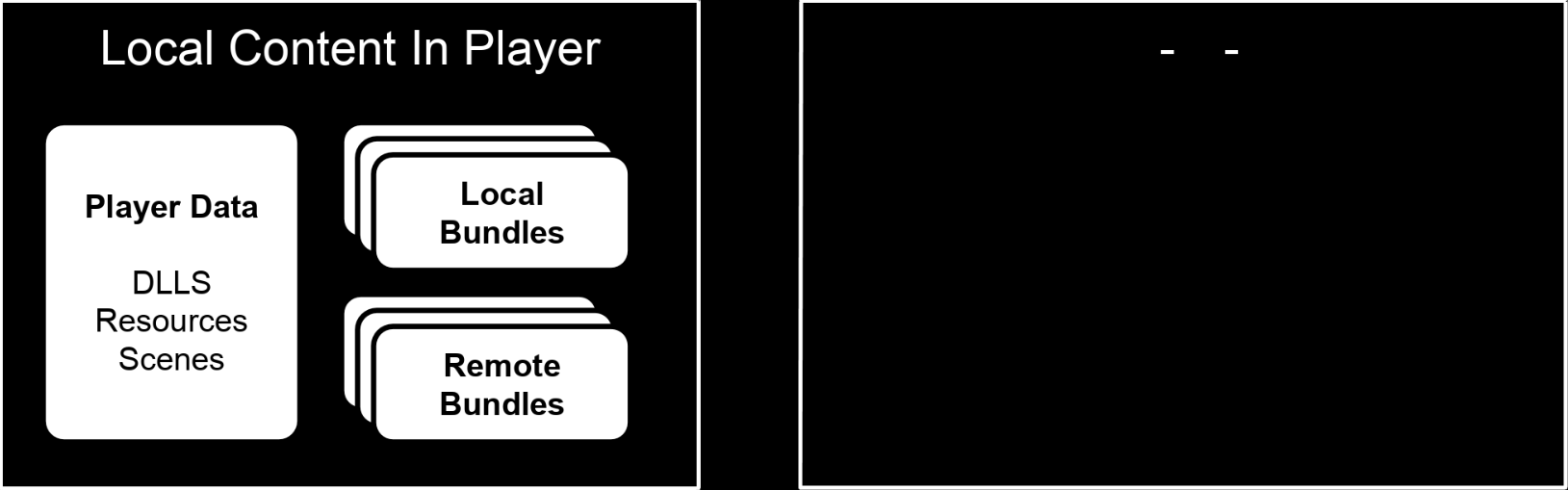
To set your build and load paths:

1. Select an Addressable Assets group from the **Project** window.
2. In the group’s **Inspector**, under **Content Packing & Loading** > **Build and Load Paths**, select the desired path pair. If you choose the <custom> option, you can specify the build and load paths separately.

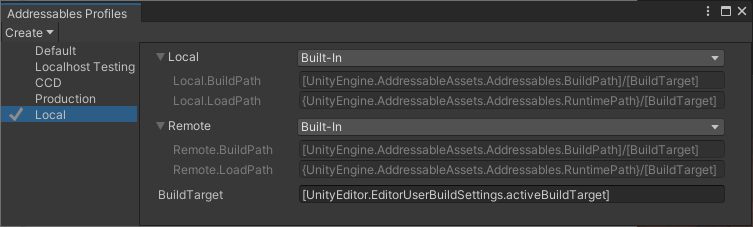
Notice that you do not enter the path directly, but rather select a variable representing the path defined in the **Profiles** window earlier. The **Path Preview** shows the current path, based on the active Profile. To edit a path directly in the Group settings Inspector, set **Build & Load Paths** to <custom> and also set the individual **Build** or **Load Path** to <custom>. The edited path applies to that group only.  
请注意，您没有直接输入路径，而是选择一个代表先前在**Profiles** 窗口中定义的路径的变量。**Path Preview** 显示基于活动配置文件的当前路径。要直接在组设置检查器中编辑路径，请将**Build & Load Paths**设置为<custom>，并将单独的构建或加载路径设置为 <custom>。编辑后的路径仅适用于该组。

### Profile examples 配置示例

Consider the following example, demonstrating the local development phase of your content.  
请考虑以下示例，演示内容的本地开发阶段。

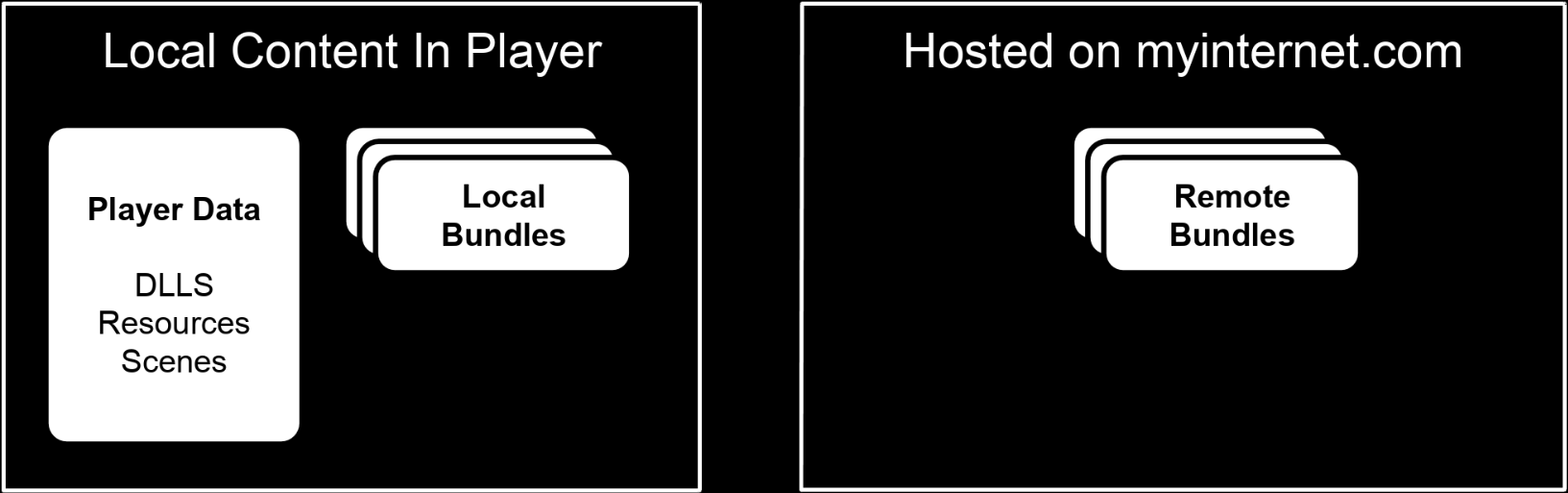
  
Content with local and remote bundles stored locally for development.

While in development, you would have both your local and remote bundles using local paths, as seen below.

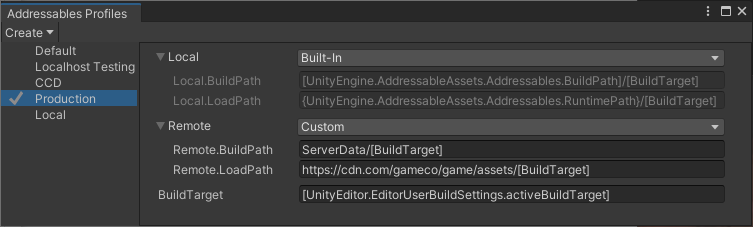
  
Paths set for local development.

In this instance, you can see that the local and remote paths are in fact local, which makes it unnecessary to set up a remote server just for local development.

Once the content is ready for production, you would move the remote bundles to a server, as the diagram below shows.

  
Content with the remote bundles moved to a server for production.

In this case, using profiles, you could change the remote load path for "Production" to that server. Without having to change your asset groups, you can change all of your remote bundles to actually become remote.

  
Paths set for hosting remote content

##### IMPORTANT

* The Addressables system only copies data from [Addressables.BuildPath](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.BuildPath.html#UnityEngine_AddressableAssets_Addressables_BuildPath) to the StreamingAssets folder during a Player build -- **it does not handle arbitrary paths specified through the LocalBuildPath or LocalLoadPath variables**. If you build data to a different location or load data from a different location than the default, you must copy the data manually.  
  Addressables 系统仅在播放器构建期间将数据从[Addressables.BuildPath](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.BuildPath.html" \l "UnityEngine_AddressableAssets_Addressables_BuildPath) 复制到StreamingAssets 文件夹——**它不处理通过LocalBuildPath 或LocalLoadPath 变量指定的任意路径**。如果将数据构建到不同位置或从不同于默认位置的位置加载数据，则必须手动复制数据。
* Similarly, you must manually upload remote AssetBundles and associated catalog and hash files to your server so that they can be accessed at the URL defined by **RemoteLoadPath**.  
  同样，您必须手动将远程 AssetBundle 和关联的目录和哈希文件上传到您的服务器，以便可以通过 **RemoteLoadPath** 定义的 URL 访问它们。

## Asset Hosting Services 资产托管服务

### Overview 总览

Hosting Services provide an integrated facility for using Addressable Assets configuration data to serve packed content to local or network-connected application builds from within the Unity Editor. Hosting Services can improve iteration velocity when testing packed content and can also serve content to connected clients on local and remote networks.  
托管服务提供了一个集成设施，用于使用可寻址资产配置数据从 Unity 编辑器中为本地或网络连接的应用程序构建提供打包内容。托管服务可以在测试打包内容时提高迭代速度，还可以为本地和远程网络上连接的客户端提供内容。

#### Packed mode testing and iteration 打包模式测试和迭代

Moving from Editor Play mode testing to platform application build testing introduces complexities and time costs to the development process. Hosting Services provide extensible Editor-embedded content delivery services that map directly to your Addressables group configuration. Using a custom Addressables profile, you can configure your application to load all content from the Unity Editor itself. This includes builds deployed to mobile devices, or any other platform, that have network access to your development system.  
从编辑器播放模式测试转移到平台应用程序构建测试会给开发过程带来复杂性和时间成本。托管服务提供可扩展的编辑器嵌入式内容交付服务，可直接映射到您的 Addressables 组配置。使用自定义 Addressables 配置文件，您可以将应用程序配置为从 Unity 编辑器本身加载所有内容。这包括部署到移动设备或任何其他平台的构建，这些平台可以通过网络访问您的开发系统。

### Setup 设置

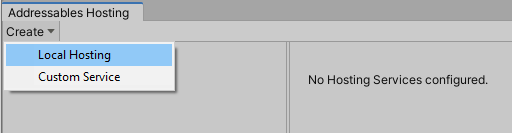
This article details the initial setup of Asset Hosting Services for your project. While the setup guide focuses on Editor workflows, you can use the API to configure Hosting Services by setting the [HostingServicesManager](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.HostingServices.HostingServicesManager.html) property of the [AddressableAssetSettings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.html) class.  
本文详细介绍了您项目的资产托管服务的初始设置。虽然设置指南侧重于编辑器工作流，但您可以使用 API 通过设置 [AddressableAssetSettings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.html) 类的[HostingServicesManager](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.HostingServices.HostingServicesManager.html) 属性来配置托管服务。

#### Configuring a new Hosting Service 配置一个新的托管服务

Use the **Hosting window** to add, configure, and enable new Hosting Services. In the Editor, select **Window** > **Asset Management** > **Addressables** > **Hosting**, or click the **Tools** > **Window** > **Hosting Services** button from the **Addressables Groups** window menu to access the **Addressables Hosting** window.

  
The ***Addressables Hosting*** window.

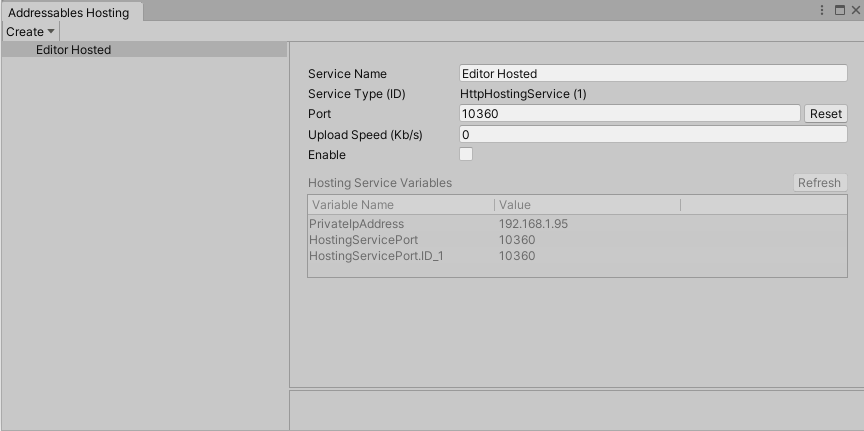
To add a new Local Hosting Service, click the **Create** > **Local Hosting** button.

  
Adding a new Hosting Service.

**Note**: For more information on implementing custom hosting service types, see the section on [custom services](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsHostingServices.html#custom-services).

The newly added service appears in the **Hosting Services** section of the **Addressables Hosting** window. Use the **Service Name** field enter a name for the service.

The new service defaults to the disabled state. To start the service, select the **Enable** check box.

  
The updated ***Addressables Hosting*** window after adding a service.

The HTTP Hosting Service automatically assigns a port number when it starts. The service saves the port number and reuses it between Unity sessions. To choose a different port, either assign a specific port number in the **Port** field, or click the **Reset** button to assign a different, random port number.  
HTTP 托管服务在启动时会自动分配一个端口号。该服务保存端口号并在 Unity 会话之间重复使用它。要选择不同的端口，请在“端口”字段中分配一个特定的端口号，或单击**Reset** 按钮分配一个不同的随机端口号。

In Editor versions 2022.1 and above, HTTP downloads are disallowed by default. In order for the default HTTPHostingService to work correctly, you need to set the **Allow downloads over HTTP** setting in **Edit** > **Project Settings...** > **Player** > **Other Settings** > **Allow downloads over HTTP** to something other than **Not allowed**.  
在 Editor 版本 2022.1 及更高版本中，默认情况下不允许使用 HTTP 下载。为了让默认的 HTTPHostingService 正常工作，您需要将**Edit** > **Project Settings...** > **Player** > **Other Settings** > **Allow downloads over HTTP**设置为 **Not allowed**以外的设置。

##### WARNING

If you reset the port number, you must execute a full application build to generate and embed the correct URL.  
如果重置端口号，则必须执行完整的应用程序构建以生成并嵌入正确的 URL。

##### NOTE

You may need to disable your local firewall or add exceptions to allow connections from mobile or other devices.  
您可能需要禁用本地防火墙或添加例外以允许来自移动设备或其他设备的连接。

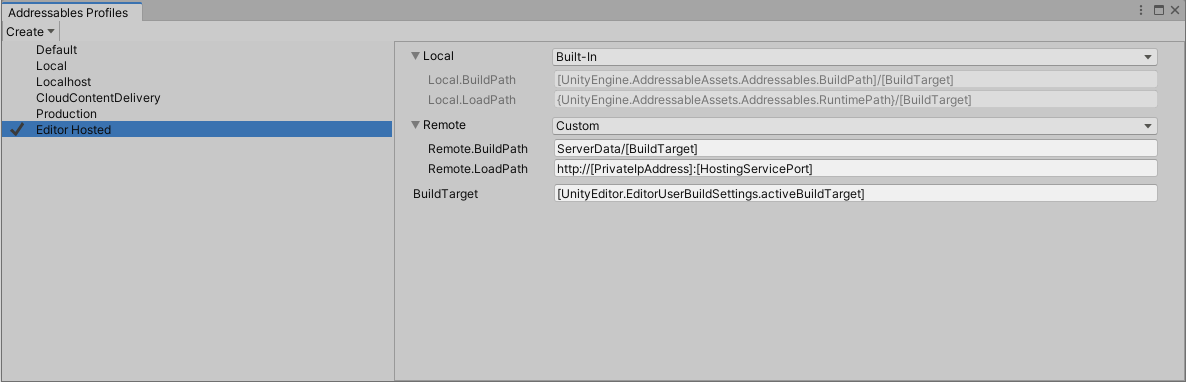
The HTTP Hosting Service is now enabled and ready to serve content from the directory specified in the remote [BuildPath](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.GroupSchemas.BundledAssetGroupSchema.BuildPath.html#UnityEditor_AddressableAssets_Settings_GroupSchemas_BundledAssetGroupSchema_BuildPath) of each asset group.

#### Hosting Service profile setup 托管服务配置

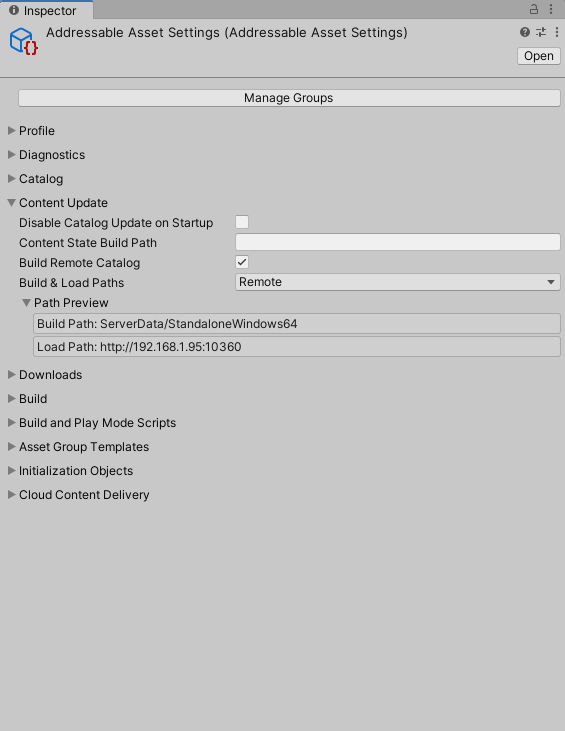
When working with Hosting Services during development, consider creating a profile that configures your asset groups to load from the Hosting Service. For more about profiles, see [Addressable Assets Profiles](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html).

Once in the **Addressables Profiles** window, create a new profile via **Create** > **Profile**. In the following example, the new profile is called "Editor Hosted".

Modify the remote loading URL to load from the Hosting Service. From the **Addressables Hosting** window, you can use the fields named [PrivateIpAddress] and [HostingServicePort] in the remote **LoadPath** variable to construct the path URL (for example, http://[PrivateIpAddress]:[HostingServicePort]).

  
Configuring the service's profile.

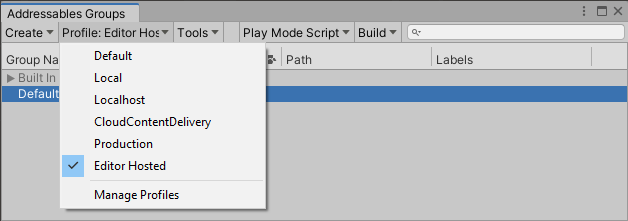
Verify that each group is configured correctly. Ensure that you set the [BuildPath](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.GroupSchemas.BundledAssetGroupSchema.BuildPath.html#UnityEditor_AddressableAssets_Settings_GroupSchemas_BundledAssetGroupSchema_BuildPath) and [LoadPath](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.GroupSchemas.BundledAssetGroupSchema.LoadPath.html#UnityEditor_AddressableAssets_Settings_GroupSchemas_BundledAssetGroupSchema_LoadPath) paths to their respective profile keys that you modified for use with Hosting Services. In this example, you can see how the profile variables in the [LoadPath](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.GroupSchemas.BundledAssetGroupSchema.LoadPath.html#UnityEditor_AddressableAssets_Settings_GroupSchemas_BundledAssetGroupSchema_LoadPath) should be expanded to build a correct base URL for loading assets from Hosted Services.

  
Inspecting the service's load paths.

##### TIP

Use the **Path Preview** to verify that the profile variables resolve to the correct variables. The load path URL IP address and port must match those shown for the service on the **Addressables Hosting** window.

Finally, select the new profile from the **Addressables Groups** window, create a build, and deploy to the target device. The Unity Editor now serves all load requests from the application through the [HttpHostingService](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.HostingServices.HttpHostingService.html) service. You can now make additions and changes to content without redeployment. Rebuild the Addressable content, and relaunch the already deployed application to refresh the content.

  
Selecting a Hosting Service profile.

#### Batch mode

You can also use Hosting Services to serve content from the Unity Editor running in batch mode. To do so, launch Unity from the command line with the following options:

-batchMode -executeMethod UnityEditor.AddressableAssets.HostingServicesManager.BatchMode

This loads the Hosting Services configuration from the default [AddressableAssetSettings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.html) object, and starts all configured services.

To use an alternative [AddressableAssetSettings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.html) configuration, create your own static method entry point, to call through the [HostingServicesManager.BatchMode(AddressableAssetSettings settings)](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.HostingServices.HostingServicesManager.BatchMode.html#UnityEditor_AddressableAssets_HostingServices_HostingServicesManager_BatchMode_UnityEditor_AddressableAssets_Settings_AddressableAssetSettings_) overload.

### Custom services 自定义服务

You can create a custom service to implement your own logic for serving content-loading requests from the Addressable assets system. For example:

* Support a custom [IResourceProvider](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceProviders.IResourceProvider.html) that uses a non-HTTP protocol for downloading content.
* Manage an external process for serving content that matches your production CDN solution (such as an Apache HTTP server).

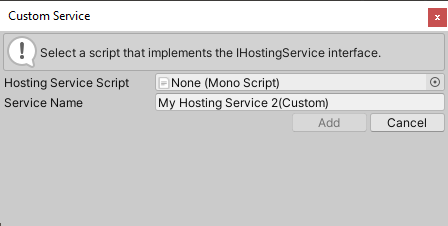
#### Implementing a custom service

The [HostingServicesManager](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.HostingServices.HostingServicesManager.html) can manage any class that implements an [IHostingService](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.HostingServices.IHostingService.html) interface (for more details on method parameters and return values, see the [API documentation](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.HostingServices.IHostingService.html).

To create a new custom service:

1. Follow the steps outlined in the [configuring a new Hosting Service](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsHostingServices.html#configuring-a-new-hosting-service) section above, but instead of selecting **Create** > **Local Hosting** button, select **Create** > **Custom Service** button instead.
2. Drag the applicable script into its field, or select it from the object picker. The dialog validates that the selected script implements the [IHostingService](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.HostingServices.IHostingService.html) interface.
3. To finish adding the service, click the **Add** button.

Moving forward, your custom service will appear in the **Service Type** drop-down options.

  
Adding a custom Asset Hosting Service.

## Customizing Addressables initialization 自定义可寻址初始化

The Addressables system initializes itself at runtime the first time you load an Addressable or make another call to an Addressable API. Call [Addressables.InitializeAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.InitializeAsync.html) to initialize Addressables earlier (this function does nothing if initialization has already occurred).  
Addressables 系统会在您第一次加载 Addressable 或再次调用 Addressable API 时在运行时自行初始化。调用  [Addressables.InitializeAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.InitializeAsync.html) 以更早地初始化 Addressables（如果初始化已经发生，则此函数不执行任何操作）。

The initialization operation performs the following tasks:

* Sets up the [ResourceManager](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceManager.html) and the [ResourceLocators](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.ResourceLocators.html).
* Loads configuration data created by Addressables from StreamingAssets.
* Executes any [initialization object](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html#initialization-object-list) operations.
* Loads the content catalog. By default, Addressables first checks for updates to the content catalog and downloads a new catalog if available.

The following Addressables settings can change initialization behavior:

* [Disable Catalog Update on Startup](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html#catalog): Addressables won't automatically check for an updated catalog. See [Updating catalogs](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadContentCatalogAsync.html#updating-catalogs) for information about manually updating your catalogs.
* [Build Remote Catalog](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html#catalog): Addressables won't attempt to load remote content without a remote catalog.
* [Custom certificate handler](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html#downloads): identify a custom certificate handler if you need one to access your remote asset hosting service.
* [Initialization object list](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html#initialization-object-list): add [IObjectInitializationDataProvider](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.Util.IObjectInitializationDataProvider.html) ScriptableObjects to your application that are invoked during the initialization operation.

The following runtime properties should be set before the initialization operation starts:

* [Custom URL transform function](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/TransformInternalId.html)
* [ResourceManager exception handler](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceManager.ExceptionHandler.html#UnityEngine_ResourceManagement_ResourceManager_ExceptionHandler)
* Static properties used for any custom runtime placeholders in your [Profile variables](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html#profile-variable-syntax)

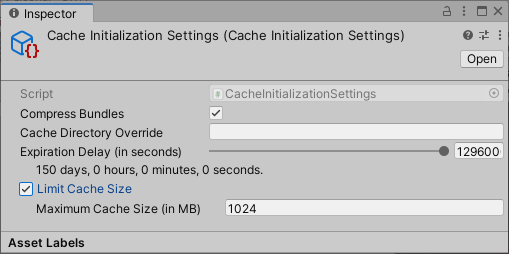
### Initialization objects

You can attach objects to the Addressable Assets settings and pass them to the initialization process at runtime. For example, you can create a [CacheInitializationSettings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.CacheInitializationSettings.html) object to initialize Unity's [Cache](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Cache.html) settings at runtime. To create your own types of initialization object, create a ScriptableObject that implements the [IObjectInitializationDataProvider](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.Util.IObjectInitializationDataProvider.html) interface. Use this object to create the [ObjectInitializationData](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.Util.ObjectInitializationData.html) asset that Addressables includes with your the runtime data.

### Cache initialization objects

Use a [CacheInitializationSettings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.CacheInitializationSettings.html) object to initialize Unity's [Cache](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Cache.html) settings at runtime.

To specify the cache initialization settings that the Addressables system should use:

1. Create the CacheInitializationSettings asset (menu: **Assets > Creat > Addressables > Initialization > Cache Initialization Settings**).
2. Select the new asset file in the Project panel to view the settings in the Inspector 
3. Adjust the settings as desired.
4. Open the Addressables Settings Inspector (menu: **Window > Asset Management > Addressables > Settings**).
5. In the **Initialization Objects** section of the Inspector, click the **+** button to add a new object to the list.
6. Select your CacheInitializationSettings asset in the File dialog and click **Open**.
7. The cache settings object is added to the list. 

When Addressables initializes at runtime, it applies these settings to the default Unity [Cache](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Cache.html). The settings apply to all AssetBundles in the default cache, not just those downloaded by the Addressables system. See [Caching](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Caching.html) for more information about the Unity cache system.

##### NOTE

Android applications built with Unity 202.1 or earlier or running on Android 9 or earlier can only play videos from uncompressed AssetBundles. You can use a CacheInitializationSettings object to disable recompression of the cache by disabling the **Compress Bundles** option.

# Managing Addressables in the Editor 在编辑器中管理可寻址

## Overview

While it's impossible to comprehensively catalog all the different ways you can organize the assets in your project, [Organizing Addressable assets](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsDevelopmentCycle.html#organizing-addressable-assets) outlines several considerations to take into account when you plan your organizational strategy.  
虽然不可能对项目中组织资产的所有不同方式进行全面分类，但组织可寻址资产概述了在[Organizing Addressable assets](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsDevelopmentCycle.html#organizing-addressable-assets) 时需要考虑的几个注意事项。

You should also understand [How Addressables interact with your Project assets](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ManagingAssets.html) while you consider how to manage your assets.  
在考虑如何管理您的资产时，您还应该了解 [How Addressables interact with your Project assets](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ManagingAssets.html)。

Addressable [Groups](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html) are the primary unit of organization with which you manage Addressable assets. An important consideration when using Addressables are your options for [Packing groups into AssetBundles](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/PackingGroupsAsBundles.html).  
可寻址 [Groups](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html)是您管理可寻址资产的主要组织单位。使用 Addressables 时的一个重要考虑因素是[Packing groups into AssetBundles](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/PackingGroupsAsBundles.html).。

In addition to your group settings, you can use the following to control how Addressables work in a project:  
除了组设置之外，您还可以使用以下内容来控制 Addressables 在项目中的工作方式：

* [Addressable Asset Settings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html): the Project-level settings
* [Profiles](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html): defines collections of build path settings that you can switch between depending on the purpose of a build. (Primarily of interest if you plan to distribute content remotely.)
* [Labels](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Labels.html): edit the Addressable asset labels used in your project.
* [Play Mode Scripts](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html#play-mode-scripts): choose how the Addressables system loads assets when you enter Play mode in the Editor.

[AssetReferences](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AssetReferences.html) provide a UI-friendly way to use Addressable assets. You can include AssetReference fields in your MonoBehaviour and ScriptableObject classes and then assign assets to them in the Editor using drag-and-drop or the object picker dialog.

The Addressables system provides the following additional tools to aid development:

* [Analyze tool](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AnalyzeTool.html): provides various analysis rules that you can run to verify that you have organized your assets the way you want, including a report on how Addressables will package your assets into bundles.
* [Event viewer](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/EventViewer.html): provides a profile view that shows when your assets are loaded and released. Use the Event viewer to verify that you are releasing assets and to monitor peak memory use.
* [Hosting Service](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsHostingServices.html): provides a simple asset server that you can use to host remote assets for local development.
* [Build layout report](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildLayoutReport.html): provides a description of the AssetBundles produced by a build.
* [Build profile log](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildProfileLog.html): provides a log profiling the build process itself so that you can see which parts take the longest.

### Organizing Addressable Assets 组织可寻址资产

There’s no single best way to organize your assets; it depends on the specific requirements of each project. Aspects to consider when planning how to manage your assets in a project include:  
没有通用的最佳方式来组织您的资产；这取决于每个项目的具体要求。计划如何管理项目中的资产时要考虑的方面包括：

* Logical organization: keeping assets in logical categories can make it easier to understand your organization and spot items that are out of place.  
  逻辑组织：将资产保持在逻辑类别中可以更容易地了解您的组织并发现不合适的项目。
* Runtime performance: performance bottlenecks can occur if your bundles become very large, or alternatively if you have a very large number of bundles.  
  运行时性能：如果您的包变得非常大，或者如果您有非常多的包，则可能会出现性能瓶颈。
* Runtime memory management: keeping assets together that you use together can help lower peak memory requirements.  
  运行时内存管理：将一起使用的资产放在一起有助于降低峰值内存需求。
* Scale: some ways of organizing assets might work well in small games, but not large ones, and vice versa.  
  规模：某些组织资产的方法可能适用于小型游戏，但不适用于大型游戏，反之亦然。
* Platform characteristics: the characteristics and requirements of a platform can be a large consideration in how to organize your assets. Some examples:  
  平台特征：平台的特征和要求可能是如何组织资产的重要考虑因素。一些例子：
  + Platforms that provide abundant virtual memory can handle large bundle sizes better than those with limited virtual memory.  
    提供丰富虚拟内存的平台可以比虚拟内存有限的平台更好地处理大包。
  + Some platforms don't support downloading content, ruling out remote distribution of assets entirely.  
    一些平台不支持下载内容，完全排除了资产的远程分发。
  + Some platforms don't support AssetBundle caching, so putting assets in local bundles, when possible, is more efficient.  
    一些平台不支持 AssetBundle 缓存，因此尽可能将资产放在本地包中会更有效率。
* Distribution: whether you distribute your content remotely or not means, at the very least, that you must separate your remote content from your local content.  
  分发：无论您是否远程分发您的内容，至少您必须将您的远程内容与您的本地内容分开。
* How often assets are updated: keep assets that you expect to update frequently separate from those that you plan to rarely update.  
  资产的更新频率：将您希望经常更新的资产与您计划很少更新的资产分开。
* Version control: the more people who work on the same assets and asset groups, the greater the chance for version control conflicts to occur in a project.  
  版本控制：处理相同资产和资产组的人越多，项目中发生版本控制冲突的可能性就越大。

### Common strategies 基本策略

Typical strategies include:

* Concurrent usage: group assets that you load at the same time together, such as all the assets for a given level. This strategy is often the most effective in the long term and can help reduce peak memory use in a project.  
  并发使用：将您同时加载的资产分组在一起，例如某一关的所有资产。从长远来看，这种策略通常是最有效的，并且可以帮助减少项目中的内存使用峰值。
* Logical entity: group assets belonging to the same logical entity together. For example, UI layout assets, textures, sound effects. Or character models and animations.  
  逻辑实体：将属于同一逻辑实体的资产组合在一起。例如，UI 布局资产、纹理、音效。或角色模型和动画。
* Type: group assets of the same type together. For example, music files, textures.  
  类型：将相同类型的资产组合在一起。例如，音乐文件、纹理。

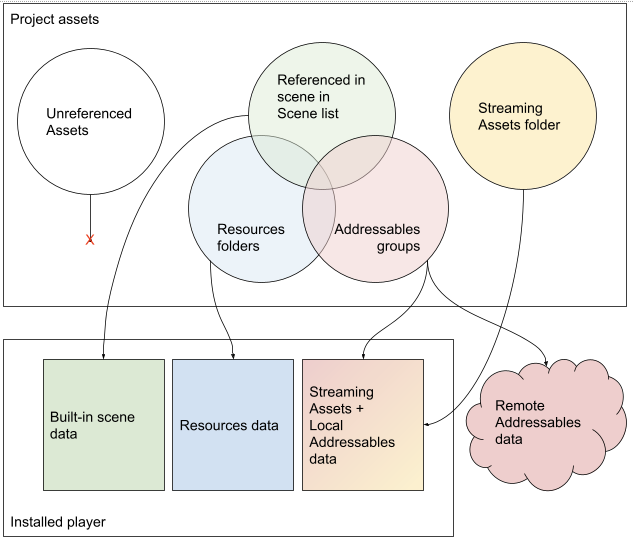
Depending on the needs of your project, one of these strategies might make more sense than the others. For example, in a game with many levels, organizing according to concurrent usage might be the most efficient both from a project management and from a runtime memory performance standpoint. At the same time, you might use different strategies for different types of assets. For example, your UI assets for menu screens might all be grouped together in a level-based game that otherwise groups its level data separately. You might also pack a group that contains the assets for a level into bundles that contain a particular type of asset.  
根据项目的需要，其中一种策略可能比其他策略更有意义。例如，在具有多个关卡的游戏中，从项目管理和运行时内存性能的角度来看，根据并发使用进行组织可能是最有效的。同时，您可能会针对不同类型的资产使用不同的策略。例如，您的菜单屏幕 UI 资产可能在基于关卡的Group里面，而其他关卡数据则单独分组。您还可以将包含某个关卡的资产的组按类型分为不同的Bundle。

See [Preparing Assets for AssetBundles](https://docs.unity3d.com/2019.4/Documentation/Manual/AssetBundles-Preparing.html) for additional information.

## How Addressables interact with other project assets Addressables 如何与其他项目资产交互

When you include a scene in your Project Build Settings and build a player, Unity includes that scene and any assets used in the scene in your game or application's built-in data. Similarly, Unity includes any assets found in your project's Resources folders in a separate, built-in collection of assets. (The difference is that assets in a scene are only loaded as part of a scene, whereas assets in Resources can be loaded independently.)  
当您在项目构建设置中包含场景并构建播放器时，Unity 会将该场景和场景中使用的任何资产包含在您的游戏或应用程序的内置数据中。类似地，Unity 将项目Resources文件夹中的任何资产包含在单独的内置资产集合中。 （不同之处在于场景中的资产仅作为场景的一部分加载，而Resources中的资产可以独立加载。）

Addressable assets can either be built into your game or application as an additional set of "local" assets, or kept external to the game build as "remote" assets hosted on a server and downloaded when they are needed. You can update remote assets independently from the application itself (although remote assets cannot include code, so you can only change assets and serialized data).  
可寻址资产可以作为一组额外的“本地”资产内置到您的游戏或应用程序中，也可以作为托管在服务器上并在需要时下载的“远程”资产保存在游戏构建外部。您可以独立于应用程序本身更新远程资产（尽管远程资产不能包含代码，因此您只能更改资产和序列化数据）。



How project assets are exported to a player build

However, if you use the same asset in more than one of these categories, then Unity makes copies of the asset when building rather than sharing a single instance. For example, if you used a Material in a built-in scene and also used it in a Prefab located in a Resources folder, you would end up with two copies of that Material in your build -- even if the Material asset itself is not located in Resources. If you then marked that same Material as Addressable, you would end up with three copies. (Files in the project StreamingAssets folder can never be referenced by assets outside that folder.)  
但是，如果您在多个类别中使用相同的资产，那么 Unity 会在构建时复制资产而不是共享单个实例。例如，如果您在内置场景中使用了一种材质，并且还在位于Resources文件夹中的预制件中使用了它，那么您最终会在构建中得到该材质的两个副本——即使材质资产本身不是位于Resources文件夹中。如果您随后将相同的材料标记为可寻址，您最终会得到三个副本。 （项目 StreamingAssets 文件夹中的文件永远不能被该文件夹外的资产引用。）

##### NOTE

Before building a player, you must make a content build of your Addressable assets. During the player build, Unity copies your local Addressables to the StreamingAssets folder so that they are included in the build along with any assets you placed in StreamingAssets. (These assets are removed at the end of the build process.) It is your responsibility to upload the remote Addressables files produced by the content build to your hosting service. See [Builds](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Builds.html) for more information.  
在构建播放器之前，您必须对可寻址资产进行内容构建。在播放器构建期间，Unity 将您的本地 Addressables 复制到 StreamingAssets 文件夹，以便它们与您放置在 StreamingAssets 中的任何资产一起包含在构建中。 （这些资产在构建过程结束时被删除。）您有责任将内容构建生成的远程 Addressables 文件上传到您的托管服务。有关详细信息，请参阅 [Builds](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Builds.html) 。

When you use Addressables in a project, Unity recommends that you move your scenes and any data in Resources folders into Addressable groups and manage them as Addressables.  
当您在项目中使用 Addressables 时，Unity 建议您将Resources文件夹中的场景和任何数据移动到 Addressable 组中，并将它们作为 Addressables 进行管理。

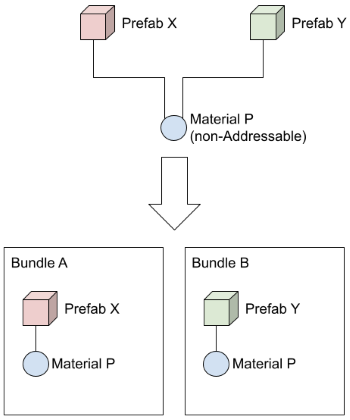
The Build Settings scene list must contain at least one scene. You can create a minimal scene that initializes your application or game.  
Build Settings 场景列表必须包含至少一个场景。您可以创建一个最小场景来初始化您的应用程序或游戏。

A small amount of data in Resources folders typically doesn't cause performance issues. If you use 3rd party packages that place assets there, you don't need to move them unless they cause problems. (Addressable assets cannot be stored in Resources folders.)  
Resources 文件夹中的少量数据通常不会导致性能问题。如果您使用将资产放在那里的第三方包，则无需移动它们，除非它们会导致问题。 （可寻址资产不能存储在 Resources 文件夹中。）

### Shared assets in groups 在groups中共享资产

When you add an asset to an Addressables group, that asset is packed into an AssetBundle when you make a content build. If an asset references other assets, known as dependencies, how those dependencies are treated depends on whether or not they are also Addressable. Dependencies that are Addressable are packed into AssetBundles according to the settings of the group they are in -- this could be the same bundle as the referencing asset or a different bundle. A dependency that is not Addressable is included in the bundle of its referencing asset.  
当您将资产添加到 Addressables 组时，该资产会在您进行内容构建时打包到 AssetBundle 中。如果资产引用其他资产，称为依赖项，那么如何处理这些依赖项取决于它们是否也是可寻址的。可寻址的依赖项根据它们所在的组的设置打包到 AssetBundle 中——这可以是与引用资产相同的包，也可以是不同的包。不可寻址的依赖项包含在其引用资产的捆绑包中。

If more than one Addressable references the same non-Addressable asset, then copies of the non-Addressable asset are included in each bundle containing a referencing Addressable.  
如果多个 Addressable 引用相同的 non-Addressable 资产，则非 Addressable 资产的副本将包含在每个包含引用 Addressable 的包中。



Non-Addressable assets are copied to each bundle with a referencing Addressable

A subtle consequence that can occur when an asset is implicitly included in more than one bundle, is that multiple instances of that asset can be instantiated at runtime rather than the single instance your game logic expects. If you change the instance state at runtime, only the object from the same bundle can see the change since all the other assets now have their own individual instance rather than sharing the common one.  
当一项资产隐式包含在多个捆绑包中时，可能会出现一个微妙的后果，即该资产的多个实例可以在运行时实例化，而不是您的游戏逻辑期望的单个实例。如果您在运行时更改实例状态，则只有来自同一包的对象才能看到更改，因为所有其他资产现在都有自己的实例，而不是共享公共实例。

To eliminate this duplication, you can make the dependency an Addressable asset and include it in one of the existing bundles or add it to a different bundle. Once you make the dependency an Addressable, the bundle it is a part of is loaded whenever you load one of the Addressables that reference it.  
要消除这种重复，您可以使依赖项成为可寻址资产并将其包含在现有捆绑包之一中或将其添加到不同的捆绑包中。一旦你使依赖成为一个 Addressable，当你加载一个引用它的 Addressable 时，它​​所属的 bundle 就会被加载。

Be aware that when you reference an asset in another bundle, then that bundle must be loaded when you load ANY asset in the current bundle, not just the asset containing the reference. Although none of the assets in this other AssetBundle are loaded, loading a bundle has its own runtime cost. See [Asset bundle dependencies](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/MemoryManagement.html#assetbundle-dependencies) for more information.  
请注意，当您在另一个包中引用资产时，那么您在当前包中加载**任何**资产时，必须加载另外的那个包，而不仅仅是包含引用资产的包。虽然没有加载其他 AssetBundle 中的任何资产，但加载一个 bundle 有其自己的运行时成本。有关更多信息，请参阅[Asset bundle dependencies](#_AssetBundle_dependencies_资产包依赖项) 。

##### TIP

Use the [Analyze tool](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AnalyzeTool.html) to identify unwanted asset duplication resulting from your project content organization.

#### SpriteAtlas 精灵图集

Some SpriteAtlas options can change how Sprites are loaded. This is important to consider when using the **Use Asset Database** [Play Mode Script](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html#play-mode-scripts).  
一些 SpriteAtlas 选项可以改变 Sprites 的加载方式。在使用**Use Asset Database** [Play Mode Script](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html#play-mode-scripts)时要考虑这一点。

* [Sprite Packer Mode](https://docs.unity3d.com/Manual/SpritePackerModes.html)
* SpriteAtlas [Include In Build](https://docs.unity3d.com/Manual/SpriteAtlasDistribution.html#Dontinclbuild)

Additionally Addressables handles SpriteAtlases a bit differently than other assets, as illustrated by the following examples:  
此外，Addressables 处理 SpriteAtlases 的方式与其他资产略有不同，如以下示例所示：

##### Addressable Sprites 可寻址精灵

**Example 1:**

Three textures exist and are marked as Addressable in three separate groups. Each texture builds to about 500KB. During the build, they are built into three separate AssetBundles, each AssetBundle only containing the given sprite metadata and texture. Each AssetBundle is about 500KB and none of these AssetBundles have dependencies.  
存在三个纹理，并在三个单独的组中标记为可寻址。每个纹理构建到大约 500KB。在构建期间，它们被构建到三个独立的 AssetBundle 中，每个 AssetBundle 只包含给定的精灵元数据和纹理。每个 AssetBundle 大约 500KB，并且这些 AssetBundle 都没有依赖关系。

**Example 2:**

The three textures in Example 1 are put into a SpriteAtlas. That atlas is not Addressable. One of the AssetBundles generated contains that atlas texture and is about 1500KB. The other two AssetBundles only contain Sprite metadata (a few KB), and list the atlas AssetBundle as a dependency. Which AssetBundle contains the texture is deterministic in that it is the same through rebuilds, but is not something that can be set by the user. This is the key portion that goes against the standard duplication of dependencies. The sprites are dependent on the SpriteAtlas texture to load, and yet that texture is not built into all three AssetBundles, but is instead built only into one.  
示例 1 中的三个纹理被放入一个 SpriteAtlas 中。该地图集不可寻址。生成的 AssetBundle 之一包含该图集纹理，大小约为 1500KB。另外两个 AssetBundle 仅包含 Sprite 元数据（几 KB），并将图集 AssetBundle 列为依赖项。哪个 AssetBundle 包含纹理是确定性的，因为它reuild过程是相同的，但不是用户可以设置的。这是违反标准重复依赖关系的关键部分。精灵依赖于 SpriteAtlas 纹理来加载，但该纹理并未构建到所有三个 AssetBundle 中，而是仅构建到一个中。

**Example 3:**

The SpriteAtlas from Example 2 is marked as Addressable in its own AssetBundle. At this point there are four AssetBundles created. If you are using a 2020.x or newer version of Unity, this builds as you would expect. The three AssetBundles with the sprites are each only a few KB and have a dependency on this fourth SpriteAtlas AssetBundle, which is about 1500KB. If you are using 2019.x or older, the texture itself may end up elsewhere. The three Sprite AssetBundles still depend on the SpriteAtlas AssetBundle. However, the SpriteAtlas AssetBundle may only contain metadata, and the texture may be in one of the other Sprite AssetBundles.  
示例 2 中的 SpriteAtlas 在其自己的 AssetBundle 中被标记为可寻址。此时创建了四个 AssetBundle。如果您使用的是 2020.x 或更新版本的 Unity，则可以按预期构建。带有精灵的三个 AssetBundle 每个只有几 KB，并且依赖于第四个 SpriteAtlas AssetBundle，大约 1500KB。如果您使用的是 2019.x 或更早版本，纹理本身可能会打包到其他地方，三个 Sprite AssetBundle 仍然依赖于 SpriteAtlas AssetBundle。但是，SpriteAtlas AssetBundle 可能仅包含元数据，并且纹理可能位于其他 Sprite AssetBundle 之一中。

##### Addressable Prefabs With Sprite dependencies 具有 Sprite 依赖性的可寻址预制件

**Example 1:**

Instead of three Addressable textures, there are three Addressable sprite prefabs. Each prefab depends on its own sprite (about 500KB). Building the three prefabs separately results, as expected, in three AssetBundles of about 500KB each.  
不是三个可寻址纹理，而是三个可寻址精灵预制件。每个预制件都依赖于它自己的精灵（大约 500KB）。正如预期的那样，分别构建三个预制件会产生三个 AssetBundle，每个约 500KB。

**Example 2:**

Taking the prefabs and textures from the previous example, all three textures are added to a SpriteAtlas, and that atlas is not marked as Addressable. In this scenario, the SpriteAtlas texture is duplicated. All three AssetBundles are approximately 1500KB. This is expected based on the general rules about duplication of dependencies, but goes against the behavior seen in "Addressable Sprite Example 2".  
以前面示例中的预制件和纹理为例，所有三个纹理都添加到 SpriteAtlas 中，并且该图集未标记为可寻址。在这种情况下，SpriteAtlas 纹理被复制。所有三个 AssetBundle 都大约为 1500KB。根据有关重复依赖项的一般规则，这是预期的，但与“可寻址精灵示例 2”中看到的行为背道而驰。

**Example 3:**

Taking the prefabs, textures, and SpriteAtlas from the above example, the SpriteAtlas is also marked as Addressable. Conforming to the rules of explicit inclusion, the SpriteAtlas texture is included only in the AssetBundle containing the SpriteAtlas. The AssetBundles with prefabs reference this fourth AssetBundle as a dependency.  
以上面例子中的 prefabs、textures 和 SpriteAtlas 为例，SpriteAtlas 也被标记为 Addressable。遵循显式包含规则，SpriteAtlas 纹理仅包含在包含 SpriteAtlas 的 AssetBundle 中。带有预制件的 AssetBundle 将这第四个 AssetBundle 引用为依赖项。

## Packing groups into AssetBundles 将组打包到 AssetBundle 中

You have a few options when choosing how the assets in a group are packed into AssetBundles:

* You can pack all Addressables in a group together in a single bundle.
* You can pack each Addressable in a group separately in its own bundle.
* You can pack all Addressables sharing the same set of labels into their own bundles.

Scene assets are always packed separately from other Addressable assets in the group. Thus a group containing a mix of Scene and non-Scene assets always produces at least two bundles when built, one for scenes and one for everything else.  
场景资产始终与组中的其他可寻址资产分开打包。因此，一个包含场景和非场景资产混合的组在构建时总是会产生至少两个包，一个用于场景，一个用于其他所有内容。

Assets in folders that are marked as Addressable and compound assets like Sprite Sheets are treated specially when you choose to pack each Addressable separately:  
当您选择单独打包每个 Addressable 时，标记为 Addressable 的文件夹中的资产和复合资产（如 Sprite Sheets）将得到特殊处理：

* All the assets in a folder that is marked as Addressable are packed together in the same folder (except for assets in the folder that are individually marked as Addressable themselves).  
  标记为可寻址的文件夹中的所有资产都打包在同一个文件夹中（文件夹中单独标记为可寻址的资产除外）。
* Sprites in an Addressable Sprite Atlas are included in the same bundle.  
  Addressable Sprite Atlas 中的Sprite 包含在同一个包中。

See [Content Packing & Loading settings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/GroupSettings.html#content-packing-loading-settings) for more information.

**NOTE**

Keeping many assets in the same group can increase the chance of version control conflicts when many people work on the same project.  
当许多人在同一个项目上工作时，将许多资产放在同一个组中会增加版本控制冲突的可能性。

The choice whether to pack your content into a few large bundles or into many smaller bundles, can have consequences at either extreme:  
选择将您的内容打包成几个大包还是许多小包，可能会产生两种极端的后果：

Dangers of too many bundles:

* Each bundle has [memory overhead](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/MemoryManagement.html#assetbundle-memory-overhead). This is tied to a number of factors, outlined on that page, but the short version is that this overhead can be significant. If you anticipate 100's or even 1000's of bundles loaded in memory at once, this could mean a noticeable amount of memory eaten up.  
  每个包都有内存开销。这与该页面上概述的许多因素有关，但简而言之，这种开销可能很大。如果您预计内存中会同时加载 100 个甚至 1000 个包，这可能意味着内存消耗量非常可观。
* There are concurrency limits for downloading bundles. If you have 1000's of bundles you need all at once, they cannot not all be downloaded at the same time. Some number will be downloaded, and as they finish, more will trigger. In practice this is a fairly minor concern, so minor that you'll often be gated by the total size of your download, rather than how many bundles it's broken into.  
  下载包有并发限制。如果您同时需要 1000 个捆绑包，则无法同时下载它们。将下载一些数字，当它们完成时，将触发更多数字。在实践中，这是一个相当小的问题，小到你经常会被下载的总大小所限制，而不是它被分成多少包。
* Bundle information can bloat the catalog. To be able to download or load catalogs, we store string-based information about your bundles. 1000's of bundles worth of data can greatly increase the size of the catalog.  
  捆绑信息会使目录膨胀。为了能够下载或加载目录，我们会存储有关您的捆绑包的基于字符串的信息。1000 束的数据会大大增加目录的大小。
* Greater likelihood of duplicated assets. Say two materials are marked as Addressable and each depend on the same texture. If they are in the same bundle, then the texture is pulled in once, and referenced by both. If they are in separate bundles, and the texture is not itself Addressable, then it will be duplicated. You then either need to mark the texture as Addressable, accept the duplication, or put the materials in the same bundle.
* 重复资产的可能性更大。假设两种材质被标记为可寻址，并且每种材质都依赖于相同的纹理。如果它们在同一个包中，那么纹理将被拉入一次，并被两者引用。如果它们在单独的包中，并且纹理本身不是可寻址的，那么它将被复制。然后您需要将纹理标记为可寻址，接受复制，或者将材料放在同一个包中。

Dangers of too few bundles:

* The UnityWebRequest (which we use to download) does not resume failed downloads. So if a large bundle is downloading and your user loses connection, the download is started over once they regain connection.  
  UnityWebRequest（我们用来下载）不会恢复失败的下载。因此，如果正在下载一个大包并且您的用户失去连接，则一旦他们重新获得连接，下载就会重新开始。
* Items can be loaded individually from bundles, but cannot be unloaded individually. For example, if you have 10 materials in a bundle, load all 10, then tell Addressables to release 9 of them, all 10 will likely be in memory. See [Memory management](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/MemoryManagement.html) for more information.  
  物品可以从捆绑包中单独装载，但不能单独卸载。例如，如果一个包中有 10 个材料，加载所有 10 个，然后告诉 Addressables 释放其中的 9 个，所有 10 个都可能在内存中。有关详细信息，请参阅内存管理。

### Scale implications as your project grows larger 随着项目规模的扩大影响规模

As your project grows larger, keep an eye on the following aspects of your assets and bundles:  
随着您的项目变得越来越大，请关注资产和捆绑包的以下方面：

* **Total bundle size**: Historically Unity has not supported files larger than 4GB. This has been fixed in some recent editor versions, but there can still be issues. It is recommended to keep the content of a given bundle under this limit for best compatibility across all platforms.  
  总包大小：从历史上看，Unity 不支持大于4GB 的文件。这已在最近的一些编辑器版本中得到修复，但仍然存在问题。建议将给定捆绑包的内容保持在此限制以下，以实现跨所有平台的最佳兼容性。
* **Bundle layout at scale**: The memory and performance trade-offs between the number of AssetBundles produced by your content build and the size of those bundles can change as your project grows larger.  
  Bundle 大规模布局：内容构建生成的 AssetBundle 数量与这些 Bundle 的大小之间的内存和性能权衡会随着项目变大而改变。
* **Sub assets affecting UI performance**: There is no hard limit here, but if you have many assets, and those assets have many subassets, it may be best to turn off sub-asset display. This option only affects how the data is displayed in the Groups window, and does not affect what you can and cannot load at runtime. The option is available in the groups window under **Tools** > **Show Sprite and Subobject Addresses**. Disabling this will make the UI more responsive.  
  影响UI 性能的子资产：这里没有硬性限制，但是如果您有很多资产，并且这些资产有很多子资产，最好关闭子资产显示。此选项仅影响数据在Addressables Groups窗口中的显示方式，不影响您在运行时可以加载和不能加载的内容。该选项位于**Tools** > **Show Sprite and Subobject Addresses**窗口中。禁用此功能将使 UI 更具响应性。
* **Group hierarchy display**: Another UI-only option to help with scale is **Group Hierarchy with Dashes**. This is available within the inspector of the top level settings. With this enabled, groups that contain dashes '-' in their names will display as if the dashes represented folder hierarchy. This does not affect the actual group name, or the way things are built. For example, two groups called "x-y-z" and "x-y-w" would display as if inside a folder called "x", there was a folder called "y". Inside that folder were two groups, called "x-y-z" and "x-y-w". This will not really affect UI responsiveness, but simply makes it easier to browse a large collection of groups.  
  组层次结构显示：另一个用于帮助缩放的UI-only 选项是带破折号的组层次结构。这在顶级设置的检查器中可用。启用此功能后，名称中包含破折号“-”的组将显示为好像破折号代表文件夹层次结构一样。这不会影响实际的组名，也不会影响事物的构建方式。例如，名为“x-y-z”和“x-y-w”的两个组将显示为好像在名为“x”的文件夹中有一个名为“y”的文件夹。该文件夹内有两组，称为“x-y-z”和“x-y-w”。这不会真正影响 UI 的响应速度，而只会使浏览大量组的集合变得更加容易。

## Groups 组

A group is the main organizational unit of the Addressables system. Use the [Addressables Groups window](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html#groups-window) to create and manage your groups and the assets they contain.  
组是 Addressables 系统的主要组织单位。使用[Addressables Groups window](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html#groups-window)创建和管理您的组及其包含的资产。

You can organize your Addressables into groups and assign settings to each group to control how the assets are handled when you create a content build. See [Organizing Addressable Assets](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsDevelopmentCycle.html#organizing-addressable-assets) for information about ways to organize your assets.  
您可以将 Addressables 组织成组，并为每个组分配设置，以控制在创建内容构建时如何处理资产。有关组织资产的方法的信息，请参阅[Organizing Addressable Assets](#_Organizing_Addressable_Assets)。

When you perform a content build, the build scripts create AssetBundles containing the assets in a group. The build determines the number of bundles to create and where to create them from both the [settings of the group](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/GroupSettings.html#group-settings) and your overall [Addressables system settings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html). See [Builds](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Builds.html) for more information.  
当您执行内容构建时，构建脚本会创建包含组中资产的 AssetBundle。构建决定了要创建的包的数量以及从 [settings of the group](#_Group_settings_组设置) 和您的整体[Addressables system settings](#_Addressable_Asset_Settings)中创建它们的位置。有关详细信息，请参阅构建。

##### NOTE

Addressable Groups only exist in the Unity Editor. The Addressables runtime code does not use a group concept. However, you can assign a label to the assets in a group if you want to find and load all the assets that were part of that group. See [**Loading Addressable assets**](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadingAddressableAssets.html) for more information about selecting the assets to load using labels.  
可寻址组仅存在于 Unity 编辑器中。 Addressables 运行时代码不使用组概念。但是，如果您想要查找并加载属于该组的所有资产，您可以为该组中的资产分配标签。有关使用标签选择要加载的资产的更多信息，请参阅 [**Loading Addressable assets**](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadingAddressableAssets.html)。

### Managing groups 管理组

Open the Addressables Groups window (menu: **Window > Asset Management > Addressables > Groups**) to manage your groups and Addressable assets. See [Addressables Groups window](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html#groups-window) for details about the information and features of this window.  
打开 Addressables Groups 窗口（菜单：Window > Asset Management > Addressables > Groups）以管理您的组和 Addressable 资产。有关此窗口的信息和功能的详细信息，请参阅 [Addressables Groups](#_Groups_window) 窗口。

#### Create a group

To create a group:

1. Open the Addressables Group window (menu: **Window > Asset Management > Addressables > Groups**).
2. Click **Create** in the toolbar at the top of the window to open the Create menu.
3. Select **Group > Packed Asset** to create a new group. (If you have created your own [Group Templates](#_Group_templates_组模板) they are also displayed in the menu.)
4. Select the new group and open its context menu (right- or cmd click on the name).
5. Choose **Rename** and assign the desired name.
6. Open the context menu again and choose **Inspect Group Settings**.
7. Adjust the group settings as desired.

For groups containing assets that you plan to distribute with your main application, the default settings are a reasonable starting point. For groups containing assets that you plan to distribute remotely, you must change the build and load paths to use the remote versions of the [Profile](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html) path variables. (To build AssetBundles for remote distribution, you must also enable the **Build Remote Catalog** option in your [Addressable System Settings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html).)  
对于包含您计划与主应用程序一起分发的资产的组，默认设置是一个合理的起点。对于包含您计划远程分发的资产的组，您必须更改构建和加载路径以使用配置文件路径变量的远程版本。 （要为远程分发构建 AssetBundle，您还必须在[Addressable System Settings](#_Addressable_Asset_Settings)中启用**Build Remote Catalog**选项。）

See [Group settings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/GroupSettings.html#group-settings) for more information about individual settings.

#### Manage assets in a group 管理组中的资产

##### To add assets to a group 添加资产

Do one of the following:

* Drag assets from the Project window to the Group window, dropping them into the desired group.
* Drag assets from one group to another.
* Check the **Addressables** option in the asset's Inspector window to add the asset to the default group. (Use the group context menu to change which group is designated as the default.)
* Add the folder containing the assets to a group (all assets added to the folder are included in the group).

###### NOTE

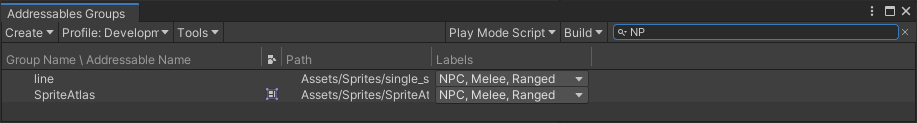
If you add assets in a Resources folder to a group, the Addressables system first moves the assets to a non-Resource location. You can move the assets elsewhere, but Addressable assets cannot be stored in a Resources folder in your Project.  
如果将 Resources 文件夹中的资产添加到组中，Addressables 系统首先将资产移动到非Resource位置。您可以将资产移动到其他地方，但可寻址资产不能存储在项目的Resources文件夹中。

##### To remove assets 移除资产

Select one or more assets in the Groups window and choose **Remove Addressables** from the context menu (or press the Delete key).

##### To find an asset 查找资产

To locate an Addressable Asset in the Groups window type all or part of its address, path, or a label into the filter control on the Groups window toolbar.

  
Filtering the group list by the string "NP" to find all assets labeled NPC

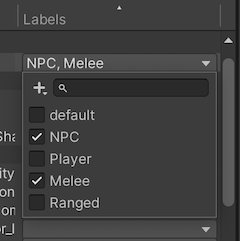
To locate the asset in your project, click on it in the Groups window. Unity selects the asset in the Project window and displays the asset's Inspector.

###### TIP

* To view the groups of the assets found, enable **Hierarchical Search**; disable this option to only show groups if they match the search string. Click the magnifying glass icon in the search box to enable or disable **Hierarchical Search**.  
  要查看找到的资产组，启用分层搜索；禁用此选项以仅显示与搜索字符串匹配的组。单击搜索框中的放大镜图标以启用或禁用分层搜索。
* To view subobject addresses, such as the Sprites in a Sprite Atlas, enable the **Show Sprite and Subobject Addresses** option using the **Tools** menu on the Groups window toolbar.  
  要查看子对象地址，例如Sprite Atlas 中的Sprite，请使用Groups 窗口工具栏上的Tools 菜单启用Show Sprite and Subobject Addresses 选项。

##### To add or remove labels 添加或移除标签

Select one or more assets in the Groups window, then click on the label field for one of the selected assets.

  
Assigning labels

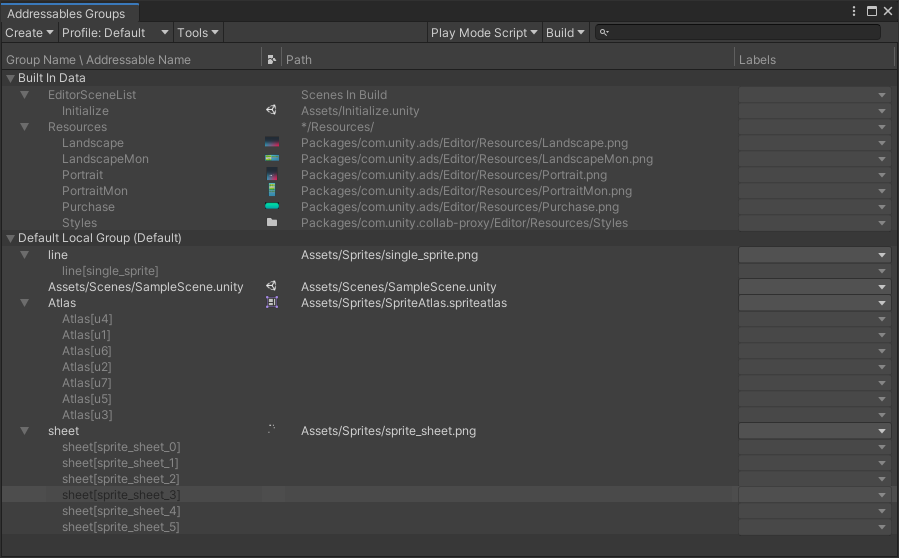
To assign labels, check or uncheck the boxes for the desired labels.

Click the top left **Plus** button, then **Manage Labels** to add, remove, or rename your labels. Or the top left **Plus** button, then **New Label** to add a new label. See [Labels](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Labels.html) for more information on how labels can be used.

### Groups window 组窗口

Use the Groups window to manage your groups and Addressable assets.

The Groups window also serves as a central location for starting content builds and accessing the tools and settings of the Addressables system.  
Groups 窗口还用作启动内容构建和访问 Addressables 系统的工具和设置的中心位置。



The Addressables Groups window showing the toolbar and list of groups and assets.

#### Group list 组列表

The Group list displays the Addressable groups in your Project. Expand a group in the list to show the assets it contains. You can also expand composite assets, such as Sprite sheets, to show the subobjects they contain.  
组列表显示项目中的可寻址组。展开列表中的组以显示它包含的资产。您还可以展开复合资源（例如 Sprite sheets）以显示它们包含的子对象。

When you first install the Addressables package, the Groups window displays two groups of assets:  
当您第一次安装 Addressables 包时，Groups 窗口会显示两组资产：

* **Built In Data**: contains assets in any Project Resource folders and any Scenes included in the Build Settings list. (None of these assets can be Addressable unless removed from Resources or the Scene list.)  
  内置数据：包含所有Resource文件夹中的资产和Build Settings列表中包含的任何场景。 （除非从Resources或Scene list中删除，否则这些资产都不能是可寻址的。）
* **Default Local Group (Default)**: Initially empty, any assets you make Addressable are added to this group. The group is set up so that its assets are built to your local build path and included in your Project builds. You can change the name, settings, and make another group the default group, if desired.  
  默认本地组（默认）：最初为空，任何您设置为可寻址的资产都会添加到该组中。该组已设置，以便将其资产构建到您的本地构建路径并包含在您的项目构建中。如果需要，您可以更改名称、设置并使另一个组成为默认组。

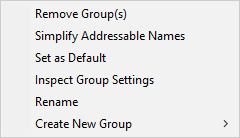
The list columns contain the following information:

| **Column** | **Purpose** |
| --- | --- |
| **Group Name** \ **Addressable Name** | The name of the item. For groups, this is an arbitrary name that you can assign. For assets, this is the Addressable address. You can edit the name or address using the context menu. |
| **Icon** | The Unity asset icon based on asset type. |
| **Path** | The path to the source asset in your Project. |
| **Labels** | Shows any labels assigned to the asset. Click on a Label entry to change the assigned labels or to manage your label definitions. |

You can sort the assets shown in the Group list by clicking one of the column headers. This sorts the assets within each group, but does not reorder the groups themselves. You can change the order in which your groups are displayed by dragging them into the desired position.  
您可以通过单击其中一个列标题对组列表中显示的资产进行排序。这会对每个组内的资产进行排序，但不会对组本身重新排序。您可以通过将组拖动到所需位置来更改组的显示顺序。

#### Group context menu 组右键菜单

Open the Group context menu (Right- or Cmd-click a group name) to access group-related commands:

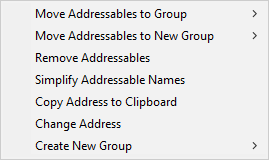


The Group content menu

| **Command** | **Action** |
| --- | --- |
| **Remove Group(s)** | Removes the Group (deleting its ScriptableObject asset). Any assets in the group revert to non-Addressable. |
| **Simplify Addressable Names** | Shortens the name of assets in the group by removing path-like components and extensions. |
| **Set as Default** | Designates the group as the default group. When you mark an asset as Addressable without explicitly assigning a group, it is added to the default group. |
| **Inspect Group Settings** | Selects the group asset in the Unity Project window and Inspector so that you can view the settings. |
| **Rename** | Allows you to edit the name. |
| **Create New Group** | Create a new group based on a group template. |

#### Asset context menu 资产右键菜单

Open the Addressable Asset context menu (Right- or Cmd-click an asset address) to access asset-related commands:



Addressable Asset context menu

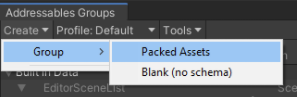
| **Command** | **Action** |
| --- | --- |
| **Move Addressables to Group** | Move the selected assets to a different, existing group. |
| **Move Addressables to New Group** | Create a new group with the same settings as the current group and move the selected assets to it. |
| **Remove Addressables** | Remove the selected assets from the Group and make it non-Addressable. |
| **Simplify Addressable Names** | Shortens the names of the selected assets by removing path-like components and extensions. |
| **Copy Address to CLipboard** | Copies the assigned address string to your system Clipboard so that you can paste it in another field. |
| **Change Address** | Allows you to edit the name. |
| **Create New Group** | Create a new group based on a group template. (Does not move the selected assets.) |

### Groups window toolbar 组工具栏

The toolbar at the top of the Addressables Group window provides access to the following commands and tools:

#### Create

Create a group.

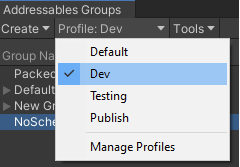


Choose a template for the group or Blank for no schema.

See [Group templates](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/GroupSettings.html#group-templates) for information about creating your own templates.

#### Profile

Set the active Profile to determine the paths used for building and loading Addressables.

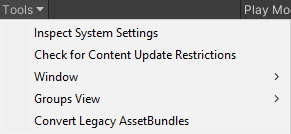


Select an existing profile or choose **Manage Profiles** to open the Profiles window.

See [Profiles](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html) for more information.

#### Tools

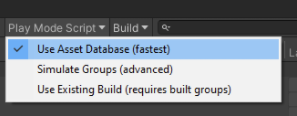
Choose from a menu of settings windows and tools.



* **Inspect System Settings**: open the [Addressables Settings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html) Inspector.
* **Check for Content Update Restrictions**: run a pre-update content check. See [Content Workflow: Update Restrictions](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html#settings) for more information.
* **Window**: open other Addressables system windows:
  + **Profiles**: open the [Profiles](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html) window.
  + **Labels**: open the [Labels](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Labels.html) window.
  + **Analyze**: open the [Analyze](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AnalyzeTool.html) tool
  + **Hosting Services**: open the [Hosting](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsHostingServices.html) window.
  + **Event Viewer**: open the [Event Viewer](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/EventViewer.html) window.
* **Groups View**: set Group window display options:
  + **Show Sprite and Subobject Addresses**: whether to show Sprite and subobjects in the Group list or just the parent object.
  + **Group Hierarchy with Dashes**: when enabled, the Groups window displays groups that contain dashes '-' in their names as if the dashes represented a group hierarchy. For example, if you name two groups "x-y-z" and "x-y-w", the the window shows an entry called "x" with a child called "y", which contains two groups, called "x-y-z" and "x-y-w". Enabling this option affects the group display only.
* **Convert Legacy AssetBundles**: Assigns non-Addressable assets to Addressable groups based on their current AssetBundle settings.

#### Play Mode Script

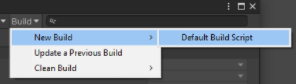
Set the active Play Mode Script.



The active Play Mode Script determines how Addressables are loaded in the Editor Play mode. See [Play Mode Scripts](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html#play-mode-scripts) for more information.

#### Build Script

Select a content build command.



* **New Build**: choose a build script to run a full content build.
* **Update a Previous Build**: run a differential update based on an earlier build.
* **Clean Build**: choose a command to clean existing build artifacts.

See [Builds](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Builds.html) for more information.

#### Filter list 搜索栏

Find items in the group list matching the specified string.



An item is shown if the specified string matches any part of the text in any column in the list.

##### TIP

Click the magnifying glass icon to enable or disable **Hierarchical Search**, which shows results within their assigned group rather than as a flat list.  
单击放大镜图标以启用或禁用**Hierarchical Search**，启用**Hierarchical Search**在其分配的组内显示结果而不是平面列表。

### Play Mode Scripts

The active Play Mode Script determines how the Addressable system accesses Addressable assets when you run your game in the Editor Play mode. When you select a Play Mode Script, it remains the active script until you choose a different one. The Play Mode Script has no effect on asset loading when you build and run your application outside the Editor.  
当前的Play Mode Script决定了当您在编辑器播放模式下运行游戏时，可寻址系统如何访问可寻址资产。当您选择一个播放模式脚本时，它会保持为当前脚本，直到您选择一个不同的脚本。当您在编辑器外构建和运行应用程序时，播放模式脚本对资产加载没有影响。

The Play Mode Scripts include:

* **Use Asset Database**: loads assets directly from the Editor asset database (which is also used for all non-Addressable assets). You do not have to build your Addressable content when using this option.
* **Simulate Groups**: analyzes content for layout and dependencies without creating AssetBundles. Loads assets from the asset database through the ResourceManager as if they were loaded through bundles. Simulates download speeds for remote AssetBundles and file loading speeds for local bundles by introducing a time delay. You can use the [Event Viewer](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/EventViewer.html) with this Play Mode script. See [ProjectConfigData](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.ProjectConfigData.html) for configuration options.  
  模拟组：在不创建AssetBundle 的情况下分析布局和依赖项的内容。通过 ResourceManager 从资产数据库加载资产，就好像它们是通过捆绑包加载的一样。通过引入时间延迟模拟远程 AssetBundle 的下载速度和本地包的文件加载速度。您可以将[Event Viewer](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/EventViewer.html)与此播放模式脚本一起使用。有关配置选项，请参阅  [ProjectConfigData](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.ProjectConfigData.html)。
* **Use Existing Build**: loads Assets from bundles created by an earlier content build. You must run a full build using a Build Script such as [Default Build Script](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Builds.html) before using this option. Remote content must be hosted at the **RemoteLoadPath** of the [Profile](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html) used to build the content.

## Labels 标签

You can tag your Addressable assets with one or more labels in the [Groups](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html) window. Labels have a few uses in the Addressables system, including:

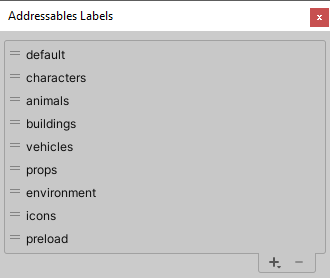
* You can use one or more labels as keys to identify which assets to load at runtime.  
  您可以使用一个或多个标签作为键来标识在运行时加载哪些资产。
* You can pack assets in a group into AssetBundles based on their assigned labels.  
  您可以根据分配的标签将一组中的资产打包到AssetBundle 中。
* You can use labels in the filter box of the Groups window to help find labeled assets

When you load assets using a list of labels, you can specify whether you want to load all assets having *any* label in the list or only assets that have *every* label in the list. For example, if you used the labels, "characters" and "animals" to load assets, you could choose to load the *union* of those two sets of assets, which includes all characters and all animals, or the *intersection* of those two sets, which includes only characters that are animals. See [Loading multiple assets](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadingAddressableAssets.html#loading-multiple-assets) for more information.  
当您使用标签列表加载资产时，您可以指定是要加载列表中具有任何标签的所有资产，还是只加载列表中具有每个标签的资产。例如，如果您使用标签“角色”和“动物”来加载资产，您可以选择加载这两组资产的并集，其中包括所有角色和所有动物，或者这两组资产的交集，其中仅包含动物角色。有关详细信息，请参阅[加载多个资产](#_Loading_multiple_assets)。

When you choose to pack assets in a group based on their assigned labels (using the group [Bundle Mode](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/GroupSettings.html#advanced-options) setting), the Addressables build script creates a bundle for each unique combination of labels in the group. For example, if you have assets in a group that you have labeled as either "cat" or "dog" and either "small" or "large", the build produces four bundles: one for small cats, one for small dogs, one for large cats, and another for large dogs.  
当您选择根据分配的标签（使用组捆绑模式设置）将资产打包到组中时，Addressables 构建脚本会为组中每个唯一的标签组合创建一个包。例如，如果您在一个组中拥有标记为“猫”或“狗”以及“小”或“大”的资产，则构建会生成四个包：一个用于小猫，一个用于小狗，一个用于大猫，另一种用于大狗。

### Managing labels 管理标签

Create and delete labels on the Labels window. Open the **Labels** window from the **Tools** menu on the **Groups** window toolbar.



*The Labels window*

To create a new label, click the **+** button at the bottom of the list. Enter the new name and click **Save**.

To delete a label, select it in the list and then click the **-** button. Deleting a label also removes it from all assets.

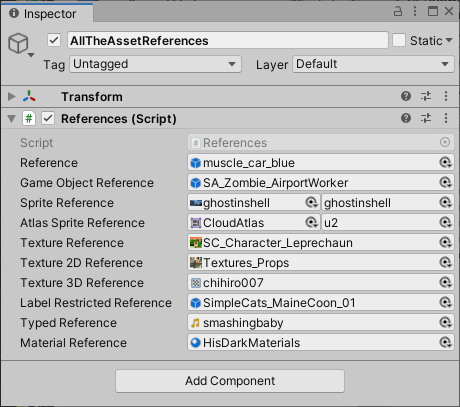
**TIP**

Until you run an Addressables build, you can undo the deletion of a label by adding it back to the Labels dialog (using the exact same string). This also adds the label back to its original assets. After you run an Addressables build, however, re-adding a deleted label no longer reapplies it to any assets.  
在运行 Addressables 构建之前，您可以通过将标签添加回“标签”对话框（使用完全相同的字符串）来撤消标签的删除。这还会将标签添加回其原始资产。但是，在运行 Addressables 构建后，重新添加已删除的标签不再将其重新应用于任何资产。

## Asset References

An [AssetReference](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.AssetReference.html) is a type that can reference an Addressable asset.

Use the AssetReference class in a MonoBehaviour or ScriptableObject. When you add a serializable AssetReference field to one of these classes, you can assign a value to the field in an Inspector window. You can restrict the assets that can be assigned to a field by type and by label.

  
*An Inspector window showing several AssetReference fields*

To assign a value, drag an asset to the field or click on the object picker icon to open a dialog that lets you choose an Addressable asset.

If you drag a non-Addressable asset to an AssetReference field, the system automatically makes the asset Addressable and adds it to your default Addressables group. Sprite and SpriteAtlas assets can have subobjects. AssetReferences assigned these types of asset display an additional object picker that allows you to specify which subobject to reference.  
如果将不可寻址的资产拖到 AssetReference 字段，系统会自动将资产设为可寻址并将其添加到默认的可寻址组中。 Sprite 和 SpriteAtlas 资源可以有子对象。分配这些类型资产的 AssetReferences 显示一个额外的对象选择器，允许您指定要引用的子对象。

See the [Basic AssetReference](https://github.com/Unity-Technologies/Addressables-Sample/tree/master/Basic/Basic%20AssetReference), [Component Reference](https://github.com/Unity-Technologies/Addressables-Sample/tree/master/Basic/ComponentReference), and [Sprite Land](https://github.com/Unity-Technologies/Addressables-Sample/tree/master/Basic/Sprite%20Land) projects in the [Addressables-Sample](https://github.com/Unity-Technologies/Addressables-Sample) repository for examples of using AssetReferences in a project.

**IMPORTANT**

To be able to assign assets from a group to an AssetReference field, the **Include GUID in Catalog** option must be enabled in the group’s Advanced Settings. The **Include GUID in Catalog** option is enabled by default.  
为了能够将组中的资产分配给 AssetReference 字段，必须在组的高级设置中启用在目录中包含 GUID 选项。默认情况下启用在目录中包含 GUID 选项。

### AssetReference types

The Addressables API provides [AssetReference](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.AssetReference.html) subclasses for common types of assets. You can use the generic subclass, [AssetReferenceT<TObject>](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.AssetReferenceT-1.html), to restrict an AssetReference field to other asset types.

The types of AssetReference include:

* [AssetReference](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.AssetReference.html): can reference any asset type
* [AssetReferenceT<TObject>](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.AssetReferenceT-1.html): can reference assets that are the same type as TObject
* [AssetReferenceTexture](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.AssetReferenceTexture.html): can reference a [Texture](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Texture.html) asset
* [AssetReferenceTexture2D](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.AssetReferenceTexture2D.html): can reference a [Texture2D](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Texture2D.html) asset
* [AssetReferenceTexture3D](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.AssetReferenceTexture3D.html): can reference a [Texture3D](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Texture3D.html) asset
* [AssetReferenceGameObject](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.AssetReferenceGameObject.html): can reference a [Prefab](https://docs.unity3d.com/2019.4/Documentation/Manual/Prefabs.html) asset
* [AssetReferenceAtlasedSprite](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.AssetReferenceAtlasedSprite.html): can reference a [SpriteAtlas](https://docs.unity3d.com/2019.4/Documentation/Manual/class-SpriteAtlas.html) asset
* [AssetReferenceSprite](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.AssetReferenceSprite.html): can reference a single [Sprite](https://docs.unity3d.com/2019.4/Documentation/Manual/Sprites.html) asset

**NOTE**

If you want to use a [CustomPropertyDrawer](https://docs.unity3d.com/2019.4/Documentation/Manual/editor-PropertyDrawers.html) with a generic AssetReferenceT (or are using a version of Unity earlier than 2020.1), you must make a concrete subclass to support custom AssetReference types.

### Adding AssetReference fields to a class

Add an [AssetReference](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.AssetReference.html), or one of its subclasses, to a MonoBehaviour or ScriptableObject by declaring it as a serializable field to the class:

using System;

using UnityEngine;

using UnityEngine.AddressableAssets;

internal class DeclaringReferences : MonoBehaviour

{

// Any asset type

public AssetReference reference;

// Prefab assets

public AssetReferenceGameObject gameObjectReference;

// Sprite asset types

public AssetReferenceSprite spriteReference;

public AssetReferenceAtlasedSprite atlasSpriteReference;

// Texture asset types

public AssetReferenceTexture textureReference;

public AssetReferenceTexture2D texture2DReference;

public AssetReferenceTexture3D texture3DReference;

// Any asset type with the specified labels

[AssetReferenceUILabelRestriction("animals", "characters")]

public AssetReference labelRestrictedReference;

// Generic asset type (Unity 2020.3+)

public AssetReferenceT<AudioClip> typedReference;

// Custom asset reference class

public AssetReferenceMaterial materialReference;

[Serializable]

public class AssetReferenceMaterial : AssetReferenceT<Material>

{

public AssetReferenceMaterial(string guid) : base(guid) { }

}

private void Start() {

// Load assets...

}

private void OnDestroy() {

// Release assets...

}

}

**NOTE**

Before Unity 2020.1, the Inspector window couldn't display generic fields by default. In earlier versions of Unity, you must make your own non-generic subclass of AssetReferenceT instead. See [Creating a concrete subclass](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AssetReferences.html#creating-a-concrete-subclass).

### Loading and releasing AssetReferences

The [AssetReference](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.AssetReference.html) class provides its own methods to load, instantiate, and release a referenced asset. You can also use an AssetReference instance as a key to any [Addressables](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.html) class method that loads assets.

The following example instantiates an AssetReference as a child of the current GameObject and releases it when the parent is destroyed:

using UnityEngine;

using UnityEngine.AddressableAssets;

internal class InstantiateReference : MonoBehaviour

{

[SerializeField]

private AssetReferenceGameObject reference;

void Start() {

if (reference != null)

reference.InstantiateAsync(this.transform);

}

private void OnDestroy() {

if (reference != null && reference.IsValid())

reference.ReleaseAsset();

}

}

See [Loading an AssetReference](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadingAddressableAssets.html#loading-an-assetreference) for more information and examples about loading assets using AssetReferences.

### Restricting AssetReference assignment to assets with specific labels 将 AssetReference 分配限制为具有特定标签的资产

Use the [AssetReferenceUILabelRestriction](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AssetReferenceUILabelRestriction.html) attribute to restrict the assets you can assign to an [AssetReference](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.AssetReference.html) field to those with specific labels. You can use this attribute reference in addition to AssetReference subclasses to restrict assignment by both type and label.

The following example prevents someone from assigning an Addressable asset to a reference that does not have either the label, "animals", or the label, "characters":

[AssetReferenceUILabelRestriction("animals", "characters")]

public AssetReference labelRestrictedReference;

**NOTE**

* The attribute only prevents assigning assets without the specified label using an Inspector in the Unity Editor. Someone could still assign an asset without the label to the field using a script.
* You cannot drag non-Addressable assets to a field with the AssetReferenceUILabelRestriction attribute.

### Creating a concrete subclass

For those cases in which you cannot use the generic form of the [AssetReference](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.AssetReference.html) class directly (in versions of Unity prior to Unity 202.1 or when using the [CustomPropertyDrawer](https://docs.unity3d.com/2019.4/Documentation/Manual/editor-PropertyDrawers.html) attribute), you can create a concrete subclass.

To create a concrete subclass, inherit from the [AssetReferenceT](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.AssetReferenceT-1.html) class and specify the asset type. You must also pass the GUID string to the base class constructor:

[Serializable]

internal class AssetReferenceMaterial : AssetReferenceT<Material>

{

public AssetReferenceMaterial(string guid) : base(guid) { }

}

You can use your custom AssetReference subclass in another script the same way as other AssetReference types:

// Custom asset reference class

public AssetReferenceMaterial materialReference;

# Building content 构建内容

## Overview

A content build processes your Addressables groups to produce the content catalog and the AssetBundles that contain your assets.  
内容构建处理您的 Addressables 组以生成内容目录和包含您的资产的 AssetBundle。

You can configure the Addressables system to build your Addressables content as part of every Player build or you can build your content separately before making a Player build. See [Building Addressables content with Player builds](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Builds.html#build-with-player) for more information about configuring these options.  
您可以配置 Addressables 系统以构建您的 Addressables 内容作为每个 Player 构建的一部分，或者您可以在构建 Player 之前单独构建您的内容。有关配置这些选项的更多信息，请参阅使用播放器构建构建可寻址内容。

If you configure Unity to build your content as part of the Player build, use the normal **Build** or **Build and Run** buttons on the Editor [Build Settings](https://docs.unity3d.com/2019.4/Documentation/Manual/PublishingBuilds.html) window to start a build. You can also invoke the Editor on the command line, passing in one of the -buildPlatformPlayer options or use an API such as [BuildPipeline.BuildPlayer](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/BuildPipeline.BuildPlayer.html) to start the build. In all cases, Unity builds your Addressables content as a pre-build step before building the Player.  
如果您将 Unity 配置为构建您的内容作为播放器构建的一部分，请使用编辑器构建设置窗口上的正常构建或构建和运行按钮来启动构建。您还可以在命令行上调用编辑器，传入 -buildPlatformPlayer 选项之一或使用 BuildPipeline.BuildPlayer 等 API 来启动构建。在上述所有情况下，Unity 都会在构建 Player 之前构建您的 Addressables 内容作为预构建步骤。

If you configure Unity to build your content separately, you must start the Addressables build using the **Build** menu on the **Addressables Groups** window as described in [Making builds](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildingContent.html). The next time you build the Player for your project, it uses the artifacts produced by the last Addressables content build run for the current platform. See [Build scripting](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildPlayerContent.html) for information about automating your Addressables build process.  
如果您将 Unity 配置为单独构建您的内容，则必须使用 Addressables Groups 窗口上的 Build 菜单启动 Addressables 构建，如制作构建中所述。下次您为您的项目构建 Player 时，它会使用上次针对当前平台运行的 Addressables 内容构建所生成的工件。有关自动化 Addressables 构建过程的信息，请参阅构建脚本。

Your content build can produce two general categories of content:

* **Local content**: content that's included directly in your player build. The Addressables system manages local content automatically as long as you use the default build path for your local content. If you change the local build path, you must copy the artifacts from the local build path to the project's Assets/StreamingAssets folder before making a Player build.  
  本地内容：直接包含在您的播放器构建中的内容。只要您使用本地内容的默认构建路径，Addressables 系统就会自动管理本地内容。如果更改本地构建路径，则必须在构建播放器之前将工件从本地构建路径复制到项目的 Assets/StreamingAssets 文件夹。
* **Remote content**: content that's downloaded from a URL after your application is installed. It is your responsibility to upload remote content to a hosting server so your application can access it the designated URL (specified by your [RemoteLoadPath](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html)).  
  远程内容：安装应用程序后从URL 下载的内容。您有责任将远程内容上传到托管服务器，以便您的应用程序可以访问指定的 URL（由您的 RemoteLoadPath 指定）。

Your [Group settings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/GroupSettings.html) determine which category a group belongs to; the active [Profile](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html) determines the specific paths and URLs that the Addressables system uses to build and load the content. (Your [Addressable Asset settings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html) also contain options that affect your content builds, such as whether to build remote content at all.)  
您的组设置决定了组属于哪个类别；活动配置文件确定 Addressables 系统用于构建和加载内容的特定路径和 URL。 （您的可寻址资产设置还包含影响内容构建的选项，例如是否构建远程内容。）

You can start builds from a script as well as from the **Groups** window. See [Build scripting](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildPlayerContent.html) for more information.  
您可以从脚本以及“组”窗口开始构建。有关详细信息，请参阅构建脚本。

The Addressables system includes the following build scripts:  
Addressables 系统包括以下构建脚本：

* **Default Build Script**: performs a full content build based on Group, Profile, and Addressables system settings.  
  默认构建脚本：根据组、配置文件和可寻址系统设置执行完整内容构建。
* **Update a Previous Build**: performs a differential content build to update a previously created build.  
  更新以前的构建：执行差异内容构建以更新以前创建的构建。
* **Play Mode scripts**: the Play Mode scripts are technically build scripts and control how the Editor accesses your content in Play Mode. See [Play Mode Scripts](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html#play-mode-scripts) for more information.  
  播放模式脚本：播放模式脚本在技术上是构建脚本并控制编辑器如何在播放模式下访问您的内容。有关详细信息，请参阅播放模式脚本。

The build scripts also provide a function to clear the cached files they create. You can run these functions from the **Build > Clean Build** menu of the [Groups window](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html#groups-window).  
构建脚本还提供了清除它们创建的缓存文件的功能。您可以从 Groups 窗口的 Build > Clean Build 菜单运行这些函数。

### Building Addressables content with Player builds 使用 Player builds 构建 Addressables 内容

When you modify Addressable assets during development, you must rebuild your Addressables content before you build the Player. You can run the Addressables content build as a separate step before building a Player or you can run both the Addressables content build and the Player build together.  
当您在开发期间修改 Addressable 资产时，您必须在构建 Player 之前重建您的 Addressables 内容。您可以在构建播放器之前将 Addressables 内容构建作为单独的步骤运行，或者您可以同时运行 Addressables 内容构建和播放器构建。

Building Addressables content together with the Player can be convenient, but does increase build time, especially on large projects, since this rebuilds the Addressables content even when you haven't modified any assets. If you don't change your Addressables content between most builds, consider disabling this option.  
与 Player 一起构建 Addressables 内容可能很方便，但确实会增加构建时间，尤其是在大型项目中，因为即使您没有修改任何资产，这也会重建 Addressables 内容。如果您在大多数构建之间不更改 Addressables 内容，请考虑禁用此选项。

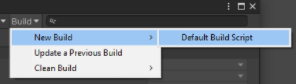
The **Build Addressables on Player Build** setting in the Project [Addressable Asset Settings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html#build) specifies which option to use for building Addressables content. You can choose the appropriate option for each Project or defer to the global Preferences setting (which you can find in the **Addressables** section of your Unity Editor Preferences). When you set a Project-level setting, it applies to all contributors who build the Project. The Preferences setting applies to all Unity Projects that don't set a specific value.  
Addressable Asset Settings 中的 Build Addressables on Player Build 设置指定用于构建 Addressables 内容的选项。您可以为每个项目选择适当的选项或遵循全局首选项设置（您可以在 Unity 编辑器Preferences的**Addressables** 部分中找到）。当您设置项目级别设置时，它适用于构建该项目的所有贡献者。 Preferences 设置适用于所有未设置特定值的 Unity 项目。

**NOTE**

Building Addressables on Player Build requires Unity 2021.2+. In earlier versions of Unity, you must build your Addressables content as a separate step.  
在 Player Build 上构建 Addressables 需要 Unity 2021.2+。在早期版本的 Unity 中，您必须将 Addressables 内容构建为一个单独的步骤。

### Build commands 构建命令

Access build commands from the **Build** menu on the toolbar at the top of the [Groups window](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html#groups-window).



The menu provides the following items:

* **New Build**: choose a build script to run a full content build. The Addressables package includes one build script, **Default Build Script**. If you create custom build scripts, you can access them here (see [Build scripting](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildPlayerContent.html)).  
  新构建：选择一个构建脚本来运行完整的内容构建。 Addressables package包括一个构建脚本：Default Build Script。如果您创建自定义构建脚本，您可以在此处访问它们（请参阅构建脚本）。
* **Update a Previous Build**: run a differential update based on an earlier build. An update build can produce smaller downloads when you support remote content distribution and publish updated content. See [Content update builds](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html).  
  更新以前的版本：运行基于早期版本的差异更新。当您支持远程内容分发并发布更新内容时，更新版本可以产生较小的下载量。请参阅内容更新构建。
* **Clean Build**: choose a command to clean existing build cache files. Each build script can provide a clean up function, which you can invoke from this menu. (See [Build scripting](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildPlayerContent.html).)  
  清除构建：选择一个命令来清除现有的构建缓存文件。每个构建脚本都可以提供清理功能，您可以从此菜单中调用该功能。 （请参阅构建脚本。）

### Build artifacts 构建工件

The build creates the following files that become part of the player build:  
该构建创建以下文件，**这些文件成为播放器构建的一部分**：

* Local AssetBundles (.bundle): according to your group, profile, and platform settings
* settings.json: contains Addressables configuration data used at runtime.
* catalog.json: the catalog used to locate and load assets at runtime (if no newer remote catalog is available).
* link.xml: prevents the Unity linker from stripping types used by your assets. See [Code Stripping](https://docs.unity3d.com/2019.4/Documentation/Manual/ManagedCodeStripping.html).

The Addressables system copies these files into your [StreamingAssets](https://docs.unity3d.com/2019.4/Documentation/Manual/StreamingAssets.html) folder when you make a player build so that they're included in your application. It removes the files when the player build is finished.

The build also creates the following files that aren't copied to streaming assets:

* Remote AssetBundles (.bundle): according to your group, profile, and platform settings. You must upload remote bundles to your hosting server.
* catalog\_timestamp.json: the catalog to upload to your hosting server (overrides the local catalog). An Addressables build only creates a remote catalog file if you enable the **Build Remote Catalog** option in the project [Addressable Asset settings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html).
* catalog\_timestamp.hash: the hash file to upload to your hosting server. Used to check whether the remote catalog has changed since the last time a client app downloaded it.
* addressables\_content\_state.bin: used to make a content update build. If you are supporting dynamic content updates, you must save this file after each full content build that you release. Otherwise, you can ignore this file. See [Content update builds](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html).
* AddressablesBuildTEP.json: build performance data. See [Build profiling](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildProfileLog.html).

## Making builds

## 执行构建

When you use the Addressables package, you can build your content (AssetBundles) as a separate step from your application player. The Addressables package provides its own build scripts for this purpose, accessible from the toolbar of the [Groups window](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html#groups-window).  
当您使用 Addressables 包时，您可以构建您的内容 (AssetBundles) 作为应用程序播放器的一个单独步骤。 Addressables 包为此提供了自己的构建脚本，可从组窗口的工具栏访问。

You have two options when building o project. You can build your Addressables content as part of the Player build or you can build them as separate steps.  
构建项目时，您有两种选择。您可以将 Addressables 内容构建为 Player 构建的一部分，也可以将它们构建为单独的步骤。

### Making a full build 完整构建

To build your content artifacts:

1. Configure your Group settings.
2. If you are distributing content remotely, configure your Profile and Addressables system settings to enable remote content distribution.  
   如果您正在远程分发内容，请配置您的Profile和Addressables system settings以启用远程内容分发。
3. Select the correct Profile.
4. Launch the build from the [Groups window](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html#groups-window).

##### TIP

If you encounter build or runtime loading issues during development, consider running the **Clean > All** command from the **Build** menu before you rebuild your content.

#### Setting up build and load paths 设置构建和加载路径

A [Profile](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html) defines separate variables for the build and load paths of local versus remote content. You can create multiple profiles in order to use different paths for different kinds of builds. For example, you could have a profile to use while you develop your Project in the Editor and another for when you publish your final content builds.  
配置文件为本地与远程内容的构建和加载路径定义了单独的变量。您可以创建多个配置文件，以便为不同类型的构建使用不同的路径。例如，您可以在编辑器中开发项目时使用一个配置文件，在发布最终内容构建时使用另一个配置文件。

For most Projects, you only need multiple profiles when you support remote content distribution. You don't typically need to change the local paths at different stages of your development process. Most projects should build local content to the default local build path and load it from the default local load path (which resolves to the StreamingAssets folder).  
对于大多数项目，只有在支持远程内容分发时才需要多个配置文件。您通常不需要在开发过程的不同阶段更改本地路径。大多数项目应该将本地内容构建到默认的本地构建路径，并从默认的本地加载路径（解析为 StreamingAssets 文件夹）加载它。

##### WARNING

Windows has a file path limit of 260 characters. If the build path of your content ends up creating a path that meets or exceeds the limit on Windows, the build fails. It is also possible to run into this issue if your project is located in a directory that is close to the character limit. The Scriptable Build Pipeline creates AssetBundles in a temporary directory during the build. This temporary path is a sub-directory of your project and can end up generating a string that goes over the Windows limit. If the Addressables content build fails with a Could not find a part of the path error, and you're on Windows, this is a likely culprit.  
Windows 的文件路径限制为 260 个字符。如果内容的构建路径最终创建的路径满足或超过 Windows 上的限制，则构建失败。如果您的项目位于接近字符限制的目录中，也可能会遇到此问题。 Scriptable Build Pipeline 在构建期间在临时目录中创建 AssetBundle。此临时路径是项目的子目录，最终可能会生成超过 Windows 限制的字符串。如果 Addressables 内容构建失败并显示“无法找到路径的一部分”错误，并且您使用的是 Windows，这可能是罪魁祸首。

##### Default local paths

##### 默认本地路径

The local build path defaults to the path provided by [Addressables.BuildPath](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.BuildPath.html#UnityEngine_AddressableAssets_Addressables_BuildPath), which is within the Library folder of your Unity project. Addressables appends a folder to the local build path based on your current platform build target setting. When you build for multiple platforms, the build places the artifacts for each platform in a different subfolder.  
本地构建路径默认为 Addressables.BuildPath 提供的路径，它位于 Unity 项目的 Library 文件夹中。 Addressables 会根据您当前的平台构建目标设置将一个文件夹附加到本地构建路径。当您为多个平台构建时，构建会将每个平台的工件放在不同的子文件夹中。

Likewise, the local load path defaults to that provided by [Addressables.RuntimePath](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.RuntimePath.html#UnityEngine_AddressableAssets_Addressables_RuntimePath), which resolves to the StreamingAssets folder. Again Addressables adds the platform build target to the path.  
同样，本地加载路径默认为 Addressables.RuntimePath 提供的路径，它解析为 StreamingAssets 文件夹。 Addressables 再次将平台构建目标添加到路径中。

When you build your local bundles to the default build path, then the build code temporarily copies the artifacts from the build path to the StreamingAssets folder when you build your player (and removes them after the build).  
当您将本地捆绑包构建到默认构建路径时，构建代码会在您构建播放器时临时将工件从构建路径复制到 StreamingAssets 文件夹（并在构建后删除它们）。

###### WARNING

If you build to or load from custom local paths, then it becomes your responsibility to copy your build artifacts to the correct place in your project before making a player build and to make sure your application can access those artifacts at runtime.  
如果您构建到自定义本地路径或从自定义本地路径加载，那么您有责任在进行播放器构建之前将构建工件复制到项目中的正确位置，并确保您的应用程序可以在运行时访问这些工件。

##### Default remote paths 默认远程路径

Addressables sets the default remote build path to an arbitrarily chosen folder name, "ServerData", which is created under your Project folder. The build adds the current platform target to the path as a subfolder to separate the unique artifacts for different platforms.  
Addressables 将默认的远程构建路径设置为“ServerData”，该文件夹是在您的项目文件夹下创建的。该构建将当前平台目标作为子文件夹添加到路径中，以分隔不同平台的独特工件。

The default remote load path is "<http://localhost/>" appended with the current profile BuildTarget variable. You MUST change this path to the base URL at which you plan to load your Addressable assets.  
默认的远程加载路径是附加了当前配置文件 BuildTarget 变量的“http://localhost/”。您必须将此路径更改为您计划加载可寻址资产的基本 URL。

Use different profiles to set up the remote load path as appropriate for the type of development, testing, or publishing you are doing. For example, you could have a profile that loads assets from a localhost server for general development builds, a profile that loads assets from a staging environment for QA builds, and one that loads assets from your Content Delivery Network (CDN) for release builds. See [Hosting](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsHostingServices.html) for more information about configuring hosting.  
根据您正在进行的开发、暂存或发布类型，使用不同的配置文件来设置远程加载路径。例如，您可能有一个配置文件从本地主机服务器加载资产以进行一般开发构建，一个配置文件从测试环境加载资产以进行 QA 构建，一个配置文件从您的内容交付网络 (CDN) 加载资产以进行发布构建。有关配置托管的更多信息，请参阅托管。

###### NOTE

When running your game in the Editor, you can use the **Use Asset Database** Play Mode Script to bypass loading assets through the remote or local load paths. This can be convenient, especially when you don't have a localhost server set up. However, it can hide group configuration and asset assignment mistakes.  
在编辑器中运行游戏时，您可以使用使用资产数据库播放模式脚本来绕过通过远程或本地加载路径加载资产。这可能很方便，尤其是当您没有设置本地主机服务器时。但是，它可以隐藏组配置和资产分配错误。

#### Setting up remote content builds 设置远程内容构建

To set up a remote content build:  
要设置远程内容构建：

1. Navigate to your AdressablesSystemSetting asset (menu: **Window > Asset Management > Addressables > Settings**).
2. Under **Catalog**, enable the **Build Remote Catalog** option. The **BuildPath** and **LoadPath** settings for the catalog must be the same as those you use for your remote groups. In most cases, use the RemoteBuildPath and RemoteLoadPath profile variables.
3. For each group that you want to build as remote content, set the **BuildPath** and **LoadPath** to the RemoteBuildPath and RemoteLoadPath profile variables (or a custom value if desired).
4. Open the [Profiles window](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html) (menu: **Window > Asset Management > Addressables > Profiles**).
5. Set the RemoteLoadPath variable to the URL where you plan to host your remote content. If you require different URLs for different types of builds, create a new Profile for each build type. See [Profiles](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html) and [Hosting](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsHostingServices.html) for more information.

See [Remote content distribution](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/RemoteContentDistribution.html) for additional information.

#### Performing the build 执行构建

After you have your group and Addressables system settings configured, you can run a content build:

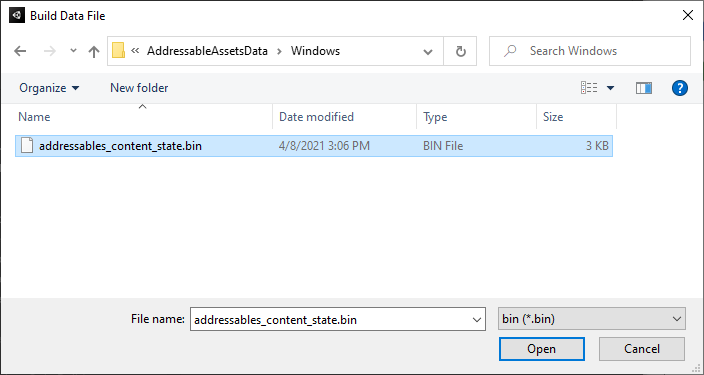
1. Open the [Groups window](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html#groups-window) (menu: **Windows > Asset Management > Addressables > Groups**).
2. Select the desired profile from the **Profile** menu on the toolbar.
3. Select the **Default Build Script** from the **Build > New Build** menu. (If you have created your own build scripts they will also be available from this menu.)

The Default Build Script creates one or more AssetBundles for each group and saves them to either the local or the remote build path.

### Making an update build 构建更新内容

When you distribute content remotely, you can perform a differential update of the previously published build to minimize the amount of data your users must download (compared to a full build).  
当您远程分发内容时，您可以对先前发布的版本执行差异更新，以最大限度地减少用户必须下载的数据量（与完整版本相比）。

Once you have configured your remote groups properly and have a previous build which contains remote content, you can perform a content update build by:  
正确配置远程组并拥有包含远程内容的先前构建后，您可以通过以下方式执行内容更新构建：

1. Open the [Groups window](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html#groups-window) (menu: **Windows > Asset Management > Addressables > Groups**).
2. Select the desired profile from the **Profile** menu on the toolbar.
3. Select the **Update a Previous Build** from the **Build** menu.  The file picker dialog opens.
4. Locate the addressables\_content\_state.bin file produced by the build you are updating. (The default location is in your Assets/AddressableAssetsData/TargetPlatform folder.)
5. Click **Open** to start the update build.

To update existing clients, copy the updated remote content to your hosting service (after appropriate testing). (An update build does include all of your local and remote content -- any player builds you create after a content update build will contain a complete set of Addressable assets.)  
要更新现有客户端，请将更新后的远程内容复制到您的托管服务（经过适当测试后）。 （更新构建确实包括您所有的本地和远程内容——您在内容更新构建之后创建的任何播放器构建都将包含一整套可寻址资产。）

Note that updating a previous build does not change the addressables\_content\_state.bin file. Use the same version of the file for future update builds (until you publish another full build created from the **New Build** menu).  
请注意，更新以前的构建不会更改 addressables\_content\_state.bin 文件。将相同版本的文件用于未来的更新构建（直到您发布从 **New Build** 菜单创建的另一个完整构建）。

See [Content Update Builds](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html) for information on how and when to use content update builds.

#### Shared AssetBundles 所谓“共享包”

When building Addressable's content, there are two possible options for what we call "shared bundles". These shared bundles are the unitybuiltinshaders AssetBundle, and the MonoScript AssetBundle.  
在构建 Addressable 的内容时，我们称之为“共享包”的内容有两种可能的选择。这些共享包是 unitybuiltinshaders AssetBundle 和 MonoScript AssetBundle。

The former is generated if any built-in shaders are used by assets included in the build. All Addressable assets that reference a shader that is built-in with the Unity Editor, such as the Standard Shader, do so by referencing this specialized shader AssetBundle..  
如果构建中包含的资产使用任何内置着色器，则会生成前者。所有引用 Unity 编辑器内置着色器的可寻址资源（例如标准着色器）都通过引用此专用着色器 AssetBundle 来实现。

The latter can be toggled on or off by changing the **AddressableAssetSettings > MonoScript Bundle Naming Prefix** option. The MonoScript bundle has naming options listed here, which are typically used in multi-project situations. It is used to build MonoScript behaviors into AssetBundles that can be referenced as a dependency.  
后者可以通过更改 AddressableAssetSettings > MonoScript Bundle Naming Prefix 选项来打开或关闭。 MonoScript 包具有此处列出的命名选项，通常用于多项目情况。它用于将 MonoScript 行为构建到可以作为依赖项引用的 AssetBundle 中。

For both of these specialized AssetBundles, some build options are derived from the default AddressableAssetGroup. If you plan on making content changes in the future, your default group, and by association the location of the shared bundles, should be remote.

If the shared bundles are built locally, they cannot be updated as part of a Content Update. In the Content Update workflow, these bundles remain unchanged since they're generated at build time. Essentially, once the MonoScript or build-in shaders bundle are included locally, they can never be changed without a new player build.  
如果共享包是在本地构建的，则它们不能作为内容更新的一部分进行更新。在内容更新工作流程中，这些包保持不变，因为它们是在构建时生成的。本质上，一旦本地包含了 MonoScript 或内置着色器包，如果没有新的播放器构建，它们将永远无法更改。

## Continuous integration 持续集成

You can use a Continuous Integration (CI) system to perform your Addressables content builds and your application player builds. This section provides general guidelines for building Addressables with CI systems, but note that every project has its own requirements and constraints, so some guidelines might not apply in all cases.  
您可以使用持续集成 (CI) 系统来执行 Addressables 内容构建和应用程序播放器构建。本节提供了使用 CI 系统构建 Addressables 的一般指南，但请注意每个项目都有自己的要求和约束，因此某些指南可能并不适用于所有情况。

### Selecting a content builder 选择内容构建器

One of the main choices when building Addressables content is selecting a content builder. By default, if you call [AddressableAssetSettings.BuildPlayerContent()](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.BuildPlayerContent.html#UnityEditor_AddressableAssets_Settings_AddressableAssetSettings_BuildPlayerContent) it uses the BuildScriptPackedMode script as the [IDataBuilder](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.IDataBuilder.html) instance. The BuildPlayerContent() function checks the [ActivePlayerDataBuilder](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.ActivePlayerDataBuilder.html#UnityEditor_AddressableAssets_Settings_AddressableAssetSettings_ActivePlayerDataBuilder) setting and calls into that script's BuildDataImplementation(..)

If you've implemented your own custom [IDataBuilder](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.IDataBuilder.html) and want to use it for your CI builds, set the [ActivePlayerDataBuilderIndex](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.ActivePlayerDataBuilderIndex.html#UnityEditor_AddressableAssets_Settings_AddressableAssetSettings_ActivePlayerDataBuilderIndex) property of [AddressableAssetSettings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.html). By default, you can access the correct settings instance through [AddressableAssetSettingsDefaultObject.Settings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.AddressableAssetSettingsDefaultObject.Settings.html#UnityEditor_AddressableAssets_AddressableAssetSettingsDefaultObject_Settings). This index refers to the position of the IDataBuilder in the list of [AddressableAssetSettings.DataBuilders](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.DataBuilders.html#UnityEditor_AddressableAssets_Settings_AddressableAssetSettings_DataBuilders). The following code sample demonstrates how to set a custom IDataBuilder:

#if UNITY\_EDITOR

using UnityEditor.AddressableAssets;

using UnityEditor.AddressableAssets.Build;

using UnityEditor.AddressableAssets.Settings;

using UnityEngine;

internal class CustomDataBuilder

{

public static void SetCustomDataBuilder(IDataBuilder builder) {

AddressableAssetSettings settings

= AddressableAssetSettingsDefaultObject.Settings;

int index = settings.DataBuilders.IndexOf((ScriptableObject)builder);

if (index > 0)

settings.ActivePlayerDataBuilderIndex = index;

else if (AddressableAssetSettingsDefaultObject.Settings.AddDataBuilder(builder))

settings.ActivePlayerDataBuilderIndex

= AddressableAssetSettingsDefaultObject.Settings.DataBuilders.Count - 1;

else

Debug.LogWarning($"{builder} could not be found " +

$"or added to the list of DataBuilders");

}

}

#endif

### Cleaning the Addressables content builder cache 清理 Addressables 内容生成器缓存

[IDataBuilder](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.IDataBuilder.html) implementations define a [ClearCachedData()](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.IDataBuilder.ClearCachedData.html#UnityEditor_AddressableAssets_Build_IDataBuilder_ClearCachedData) method, which cleans up any files created by that data builder. For example, the default BuildScriptPackedMode script deletes the following:  
IDataBuilder 实现定义了一个 ClearCachedData() 方法，该方法清除该数据生成器创建的所有文件。例如，默认的 BuildScriptPackedMode 脚本会删除以下内容：

* The content catalog
* The serialized settings file
* The built AssetBundles
* Any link.xml files created

You can call IDataBuilder.ClearCachedData() as part of your CI process to make sure the build does not use files generated by previous builds.

### Cleaning the Scriptable Build Pipeline cache 清理 Scriptable Build Pipeline 缓存

Cleaning the Scriptable Build Pipeline (SBP) cache cleans the BuildCache folder from the Library directory along with all the hash maps generated by the build and the Type Database. The Library/BuildCache folder contains .info files created by SBP during the build which speeds up subsequent builds by reading data from these .info files instead of re-generating data that hasn't changed.  
清除脚本化构建管道 (SBP) 缓存会清除库目录中的 BuildCache 文件夹以及构建和类型数据库生成的所有哈希映射。 Library/BuildCache 文件夹包含 SBP 在构建期间创建的 .info 文件，它通过从这些 .info 文件中读取数据而不是重新生成未更改的数据来加速后续构建。

To clear the SBP cache in a script without opening a prompt dialog, call [BuildCache.PurgeCache(false)](https://docs.unity3d.com/Packages/com.unity.scriptablebuildpipeline@1.19/api/UnityEditor.Build.Pipeline.Utilities.BuildCache.html#UnityEditor_Build_Pipeline_Utilities_BuildCache_PurgeCache_).

## Build scripting 构建脚本

There are a few ways in which you can use the Addressables API to customize your project build:

* Start a build from a script
* Override an existing script
* Extend [BuildScriptBase](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.DataBuilders.BuildScriptBase.html) or implement [IDataBuilder](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.IDataBuilder.html)

When you customize a build script to handle different asset types or handle assets in a different way, you might also need to customize the [Play Mode Scripts](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html#play-mode-scripts) so that the Editor can handle those assets in the same way during Play mode.  
当您自定义构建脚本以处理不同的资产类型或以不同的方式处理资产时，您可能还需要自定义播放模式脚本，以便编辑器可以在播放模式期间以相同的方式处理这些资产。

### Starting an Addressables build from a script 从脚本启动 Addressables 构建

To start an Addressables build from another script, call the [AddressableAssetSettings.BuildPlayerContent](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.BuildPlayerContent.html) method.  
要从另一个脚本启动 Addressables 构建，请调用 AddressableAssetSettings.BuildPlayerContent 方法。

Before starting the build, you should set the active [Profile](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html) and the active build script. You can also set a different [AddressableAssetSettings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.html) object than the default, if desired.

在开始构建之前，您应该设置活动配置文件和活动构建脚本。如果需要，您还可以设置不同于默认值的 AddressableAssetSettings 对象。

There are a few pieces of information that BuildPlayerContent takes into consideration when performing the build: the [AddressableAssetSettingsDefaultObject](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.AddressableAssetSettingsDefaultObject.html), [ActivePlayerDataBuilder](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.ActivePlayerDataBuilder.html#UnityEditor_AddressableAssets_Settings_AddressableAssetSettings_ActivePlayerDataBuilder), and the addressables\_content\_state.bin file.  
在执行构建时，BuildPlayerContent 会考虑一些信息：AddressableAssetSettingsDefaultObject、ActivePlayerDataBuilder 和 addressables\_content\_state.bin 文件。

#### Set the AddressableAssetSettings

The settings defined by [AddressableAssetSettings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.html) include the list of groups and the profile to use.

To access the settings that you see in the Editor (menu: **Window > Asset Management > Addressables > Settings**), use the static [AddressableAssetSettingsDefaultObject.Settings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.AddressableAssetSettingsDefaultObject.Settings.html#UnityEditor_AddressableAssets_AddressableAssetSettingsDefaultObject_Settings) property. However, if desired, you can use a different settings object for a build.

To load a custom settings object in a build:

static void getSettingsObject(string settingsAsset) {

// This step is optional, you can also use the default settings:

//settings = AddressableAssetSettingsDefaultObject.Settings;

settings

= AssetDatabase.LoadAssetAtPath<ScriptableObject>(settingsAsset)

as AddressableAssetSettings;

if (settings == null)

Debug.LogError($"{settingsAsset} couldn't be found or isn't " +

$"a settings object.");

}

#### Set the active Profile

A build started with BuildContent uses the variable settings of the active Profile. To set the active Profile as part of your customized build script, assign the ID of the desired profile to the [activeProfileId](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.activeProfileId.html#UnityEditor_AddressableAssets_Settings_AddressableAssetSettings_activeProfileId) field of the [AddressableAssetSettingsDefaultObject.Settings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.AddressableAssetSettingsDefaultObject.Settings.html#UnityEditor_AddressableAssets_AddressableAssetSettingsDefaultObject_Settings) object.

The [AddressableAssetSettings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.html) object contains the list of profiles. Use the name of the desired profile to look up its ID value and then assign the ID to the [activeProfileId](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.activeProfileId.html#UnityEditor_AddressableAssets_Settings_AddressableAssetSettings_activeProfileId) variable:

static void setProfile(string profile) {

string profileId = settings.profileSettings.GetProfileId(profile);

if (String.IsNullOrEmpty(profileId))

Debug.LogWarning($"Couldn't find a profile named, {profile}, " +

$"using current profile instead.");

else

settings.activeProfileId = profileId;

}

#### Set the active build script

The BuildContent method launches the build based on the current [ActivePlayerDataBuilder](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.ActivePlayerDataBuilder.html#UnityEditor_AddressableAssets_Settings_AddressableAssetSettings_ActivePlayerDataBuilder) setting. To use a specific build script, assign the index of the IDataBuilder object in the [AddressableAssetSetting.DataBuilders](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.DataBuilders.html#UnityEditor_AddressableAssets_Settings_AddressableAssetSettings_DataBuilders) list to the [ActivePlayerDataBuilderIndex](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.ActivePlayerDataBuilderIndex.html#UnityEditor_AddressableAssets_Settings_AddressableAssetSettings_ActivePlayerDataBuilderIndex) property.

The build script must be a ScriptableObject that implements [IDataBuilder](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.IDataBuilder.html) and you must add it to the [DataBuilders](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.DataBuilders.html#UnityEditor_AddressableAssets_Settings_AddressableAssetSettings_DataBuilders) list in the [AddressableAssetSettings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.html) instance. Once added to the list, use the standard [List.IndexOf](https://docs.microsoft.com/dotnet/api/system.collections.generic.list-1.indexof) method to get the index of the object.

static void setBuilder(IDataBuilder builder) {

int index = settings.DataBuilders.IndexOf((ScriptableObject)builder);

if (index > 0)

settings.ActivePlayerDataBuilderIndex = index;

else

Debug.LogWarning($"{builder} must be added to the " +

$"DataBuilders list before it can be made " +

$"active. Using last run builder instead.");

}

#### Launch a build

After setting the profile and builder to use (if desired), you can launch the build:

static bool buildAddressableContent() {

AddressableAssetSettings

.BuildPlayerContent(out AddressablesPlayerBuildResult result);

bool success = string.IsNullOrEmpty(result.Error);

if (!success) {

Debug.LogError("Addressables build error encountered: " + result.Error);

}

return success;

}

To check for success, use BuildPlayerContent(out AddressablesPlayerBuildResult result). result.Error contains any error message returned if the Addressables build failed. If string.IsNullOrEmpty(result.Error) is true, the build was successful.

#### Example script to launch build

The following example adds a couple of menu commands to the Asset Management > Addressables menu in the Editor. The first command builds the Addressable content using the preset profile and build script. The second command builds the Addressable content, and, if it succeeds, builds the Player, too.

Note that if your build script makes setting changes that require a domain reload, you should run the build script using Unity command line options, instead of running it interactively in the Editor. See [Domain reloads and Addressable builds](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildPlayerContent.html#domain-reloads-and-addressables-builds) for more information.

#if UNITY\_EDITOR

using UnityEditor;

using UnityEditor.AddressableAssets.Build;

using UnityEditor.AddressableAssets.Settings;

using System;

using UnityEngine;

internal class BuildLauncher

{

public static string build\_script

= "Assets/AddressableAssetsData/DataBuilders/BuildScriptPackedMode.asset";

public static string settings\_asset

= "Assets/AddressableAssetsData/AddressableAssetSettings.asset";

public static string profile\_name = "Default";

private static AddressableAssetSettings settings;

static void getSettingsObject(string settingsAsset) {

// This step is optional, you can also use the default settings:

//settings = AddressableAssetSettingsDefaultObject.Settings;

settings

= AssetDatabase.LoadAssetAtPath<ScriptableObject>(settingsAsset)

as AddressableAssetSettings;

if (settings == null)

Debug.LogError($"{settingsAsset} couldn't be found or isn't " +

$"a settings object.");

}

static void setProfile(string profile) {

string profileId = settings.profileSettings.GetProfileId(profile);

if (String.IsNullOrEmpty(profileId))

Debug.LogWarning($"Couldn't find a profile named, {profile}, " +

$"using current profile instead.");

else

settings.activeProfileId = profileId;

}

static void setBuilder(IDataBuilder builder) {

int index = settings.DataBuilders.IndexOf((ScriptableObject)builder);

if (index > 0)

settings.ActivePlayerDataBuilderIndex = index;

else

Debug.LogWarning($"{builder} must be added to the " +

$"DataBuilders list before it can be made " +

$"active. Using last run builder instead.");

}

static bool buildAddressableContent() {

AddressableAssetSettings

.BuildPlayerContent(out AddressablesPlayerBuildResult result);

bool success = string.IsNullOrEmpty(result.Error);

if (!success) {

Debug.LogError("Addressables build error encountered: " + result.Error);

}

return success;

}

[MenuItem("Window/Asset Management/Addressables/Build Addressables only")]

public static bool BuildAddressables() {

getSettingsObject(settings\_asset);

setProfile(profile\_name);

IDataBuilder builderScript

= AssetDatabase.LoadAssetAtPath<ScriptableObject>(build\_script) as IDataBuilder;

if (builderScript == null) {

Debug.LogError(build\_script + " couldn't be found or isn't a build script.");

return false;

}

setBuilder(builderScript);

return buildAddressableContent();

}

[MenuItem("Window/Asset Management/Addressables/Build Addressables and Player")]

public static void BuildAddressablesAndPlayer() {

bool contentBuildSucceeded = BuildAddressables();

if (contentBuildSucceeded) {

var options = new BuildPlayerOptions();

BuildPlayerOptions playerSettings

= BuildPlayerWindow.DefaultBuildMethods.GetBuildPlayerOptions(options);

BuildPipeline.BuildPlayer(playerSettings);

}

}

}

#endif

#### Domain reloads and Addressables builds

If your scripted build process involves changing settings that trigger a domain reload before it makes an Addressables build, then you should script such builds to use the Unity Editor [command line interface](https://docs.unity3d.com/2019.4/Documentation/Manual/CommandLineArguments.html) rather than interactively running a script in the Editor. These types of settings include:

* Changing the defined compiler symbols
* Changing platform target or target group

When you run a script that triggers a domain reload interactively in the Editor (using a menu command, for example), your Editor script finishes executing before the domain reload occurs. Thus, if you immediately start an Addressables build, both your code and imported assets are still in their original state. You must wait for the domain reload to complete before you start the content build.

Waiting for the domain reload to finish is relatively straightforward when you run the build from the command line, but can be difficult or impossible to accomplish reliably in an interactive script (for a variety of reasons).

The following example script defines two functions that can be invoked when running Unity on the command line. The ChangeSettings example sets the specified define symbols. The BuildContentAndPlayer function runs the Addressables build and the Player build.

#if UNITY\_EDITOR

using System;

using UnityEditor;

using UnityEditor.AddressableAssets;

using UnityEditor.AddressableAssets.Build;

using UnityEditor.AddressableAssets.Settings;

using UnityEditor.Build.Reporting;

using UnityEngine;

internal class BatchBuild

{

public static string build\_script

= "Assets/AddressableAssetsData/DataBuilders/BuildScriptPackedMode.asset";

public static string profile\_name = "Default";

public static void ChangeSettings() {

string defines = "";

string[] args = Environment.GetCommandLineArgs();

foreach (var arg in args)

if (arg.StartsWith("-defines=", System.StringComparison.CurrentCulture))

defines = arg.Substring(("-defines=".Length));

var buildSettings = EditorUserBuildSettings.selectedBuildTargetGroup;

PlayerSettings.SetScriptingDefineSymbolsForGroup(buildSettings, defines);

}

public static void BuildContentAndPlayer() {

AddressableAssetSettings settings

= AddressableAssetSettingsDefaultObject.Settings;

settings.activeProfileId

= settings.profileSettings.GetProfileId(profile\_name);

IDataBuilder builder

= AssetDatabase.LoadAssetAtPath<ScriptableObject>(build\_script) as IDataBuilder;

settings.ActivePlayerDataBuilderIndex

= settings.DataBuilders.IndexOf((ScriptableObject)builder);

AddressableAssetSettings.BuildPlayerContent(out AddressablesPlayerBuildResult result);

if (!string.IsNullOrEmpty(result.Error))

throw new Exception(result.Error);

BuildReport buildReport

= BuildPipeline.BuildPlayer(EditorBuildSettings.scenes,

"d:/build/winApp.exe", EditorUserBuildSettings.activeBuildTarget,

BuildOptions.None);

if (buildReport.summary.result != BuildResult.Succeeded)

throw new Exception(buildReport.summary.ToString());

}

}

#endif

To call these functions, use [Unity's command line arguments](https://docs.unity3d.com/2019.4/Documentation/Manual/CommandLineArguments.html) in a terminal or command prompt or in a shell script:

D:\Unity\2020.3.0f1\Editor\Unity.exe -quit -batchMode -projectPath . -executeMethod BatchBuild.ChangeSettings -defines=FOO;BAR -buildTarget Android

D:\Unity\2020.3.0f1\Editor\Unity.exe -quit -batchMode -projectPath . -executeMethod BatchBuild.BuildContentAndPlayer -buildTarget Android

##### NOTE

If you specify the platform target as a command line parameter, you can perform an Addressables build in the same command. However, if you wanted to change the platform in a script, you should do it in a separate command, such as the ChangeSettings function in this example.

### Overriding an existing script

If you want to use the same basic build as the default, but want to treat specific groups or types of assets differently, you can extend the default build script and override the functions within it. If the group or asset the build script is processing is one that you want to treat differently, you can run your own code, otherwise you can call the base class version of the function to use the default algorithm.

See the [Addressable variants project](https://github.com/Unity-Technologies/Addressables-Sample/tree/master/Advanced/Addressable%20Variants) in the [Addressables-Sample](https://github.com/Unity-Technologies/Addressables-Sample) repository for an example.

### Extending BuildScriptBase or implementing IDataBuilder

You can extend [BuildScriptBase](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.DataBuilders.BuildScriptBase.html) or implement [IDataBuilder](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.IDataBuilder.html) to substantially change the Addressables build system. To understand how the Addressables system builds content, first examine the default build script, BuildScriptPackedMode.cs, which you can find in the Addressables package folder, Addressables/EditorBuild/DataBuilders.

#### Save the content state

If you support [remote content distribution](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/RemoteContentDistribution.html) and update your content between player releases, you must record the state of your Addressables groups at the time of the build. Recording the state allows you to perform a differential build using the [Update a Previous Build](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html#building-content-updates) script.

See the implementation of the default build script, BuildScriptPackedMode.cs, for details.

# Remote content distribution 远程内容分发

## Overview

Distributing content remotely can reduce initial app download size and install time. You can also update remotely distributed assets without republishing your app or game  
远程分发内容可以减少初始应用程序下载大小和安装时间。您还可以更新远程分发的资产，而无需重新发布您的应用或游戏

When you assign a remote URL as the Load Path of a group, the Addressables system loads assets in the group from that URL. When you enable the Build Remote Catalog option, Addressables looks up the addresses of any remote assets in the remote catalog, allowing you to make changes to Addressable assets without forcing users to update and reinstall the entire game or application.  
当您将远程 URL 指定为组的加载路径时，Addressables 系统会从该 URL 加载组中的资产。当您启用构建远程目录选项时，Addressables 会在远程目录中查找任何远程资产的地址，从而允许您对可寻址资产进行更改，而无需强制用户更新和重新安装整个游戏或应用程序。

After [enabling remote distribution](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/RemoteContentDistribution.html#enabling-remote-distribution), you can build your content in two ways:  
启用远程分发后，您可以通过两种方式构建内容：

* A full [content build](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Builds.html) using the **New Build > Default Build Script**: builds all content bundles and catalogs. Always perform a full build before rebuilding your player when preparing to publish or update your full application.  
  使用New Build > Default Build Script 构建完整内容：构建所有内容包和目录。在准备发布或更新完整应用程序时，请始终在重建播放器之前执行完整构建。
* A [content update build](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html) using the **Update a Previous Build** script: builds all content bundles and catalogs, but sets up the remote catalog so that installed applications only need to download the changed bundles. Run the [**Check for Content Update Restrictions**](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html#check-for-content-update-restrictions-tool) tool to identify changes and prepare your groups before building an update.  
  使用Update a Previous Build 脚本的内容更新构建：构建所有内容包和目录，但设置远程目录以便已安装的应用程序只需要下载更改的包。运行检查内容更新限制工具以识别更改并在构建更新之前准备您的组。

After building a full build or an update, you must upload your remote catalog, catalog hash file, and remote bundles to your hosting service.  
构建完整构建或更新后，您必须将远程目录、目录哈希文件和远程包上传到托管服务。

See [Using Profiles to aid development](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/RemoteContentDistribution.html#using-profiles-to-aid-development) for tips on setting up Addressables Profiles to help you develop, test, and publish remote content.  
有关设置可寻址配置文件以帮助您开发、测试和发布远程内容的提示，请参阅使用配置文件来帮助开发。

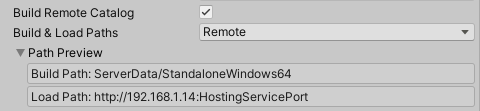
### Enabling remote distribution 启用远程分发

To enable remote distribution of your content, you must enable the remote catalog and set up the groups containing the assets you want to host remotely.  
要启用内容的远程分发，您必须启用远程目录并设置包含您要远程托管的资产的组。

#### Enabling the remote catalog 启用远程目录

Enable the remote catalog in your **Addressable Asset Settings** Inspector (menu: **Window > Asset Management > Addressables > Settings**).  
在可寻址资产设置检查器中启用远程目录（菜单：**Window > Asset Management > Addressables > Settings**）。

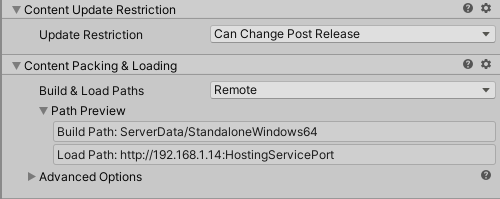
* **Build Remote Catalog**: enabled
* **Build & Load Paths**: Remote

  
Enabling the remote catalog

The catalog and its accompanying hash file are built to the folder specified by the **Build Path** setting. You must upload these files so that they can be accessed at the URL specified by your **Load Path** setting. Unless you have a specific reason not to, use the **Remote** location so that the catalog is built to and loaded from the same paths as your remote bundles.  
目录及其随附的哈希文件构建到构建路径设置指定的文件夹中。您必须上传这些文件，以便可以通过加载路径设置指定的 URL 访问它们。除非您有特殊原因不这样做，否则请使用**Remote** 位置，以便将目录构建到与远程包相同的路径并从中加载。

#### Setting up a remote group 设置远程组

To set up a group so that the assets in it can be hosted remotely, set the **Build & Load Paths** using the **Remote** location:  
要设置一个组以便可以远程托管其中的资产，请使用远程位置设置构建和加载路径：



If you plan to publish content updates between publishing full rebuilds of your application, set the **Update Restriction** value according to how often you expect to update content in a group.  
如果您计划在发布应用程序的完整重建之间发布内容更新，请根据您希望在组中更新内容的频率设置更新限制值。

Choose **Cannot Change Post Release** for groups that produce larger bundles, especially if you do not anticipate changing most of the assets in the group. If you do change assets in a group with this setting, the Addressables tools move the changed assets to a new group for the update. Only the new bundles are downloaded by installed applications.  
为生成较大捆绑包的组选择发布后无法更改，尤其是当您预计不会更改组中的大部分资产时。如果您使用此设置更改组中的资产，Addressables 工具会将更改的资产移动到新组以进行更新。已安装的应用程序只会下载新的捆绑包。

Choose **Can Change Post Release** for groups containing assets that you expect to change frequently. If you change assets in a group with this setting, the bundles containing those assets are rebuilt as a whole and will be redownloaded by installed applications. To reduce the amount of data that needs to be downloaded after an update, try to keep the bundles produced by groups with this setting as small as possible.  
为包含您希望经常更改的资产的组选择发布后可以更改。如果您使用此设置更改组中的资产，则包含这些资产的包将作为一个整体重建，并将由已安装的应用程序重新下载。要减少更新后需要下载的数据量，请尝试使使用此设置的组生成的捆绑包尽可能小。

See [Content update builds](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html) for more information about updating remote content.

The **Advanced Options** section contains some options that affect remote hosting and downloads (but aren't necessary to enable remote hosting). See [Advanced Options](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/GroupSettings.html#advanced-options) for more information.  
高级选项部分包含一些影响远程托管和下载的选项（但不是启用远程托管所必需的）。有关详细信息，请参阅高级选项。

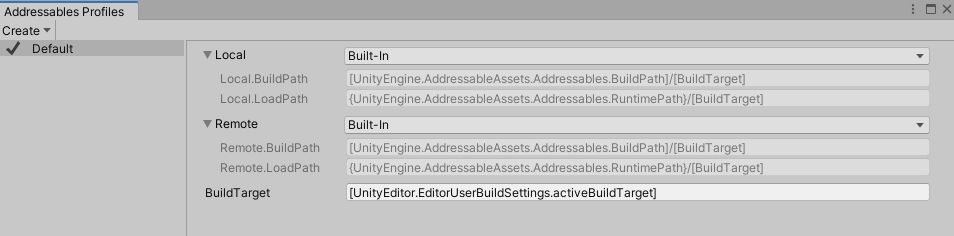
### Using Profiles to aid development 使用配置文件帮助开发

A [Profile](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html) defines variables that you can use to set the build and load paths for both your local and remote content.  
配置文件定义可用于设置本地和远程内容的构建和加载路径的变量。

When you distribute content remotely, you typically need to set different URLS (load paths) for your remote content depending on why you are making a build. Some examples of such situations include:  
远程分发内容时，通常需要根据构建的原因为远程内容设置不同的 URLS（加载路径）。这种情况的一些例子包括：

**Builds for testing general game play and function**用于测试一般游戏玩法和功能的构建

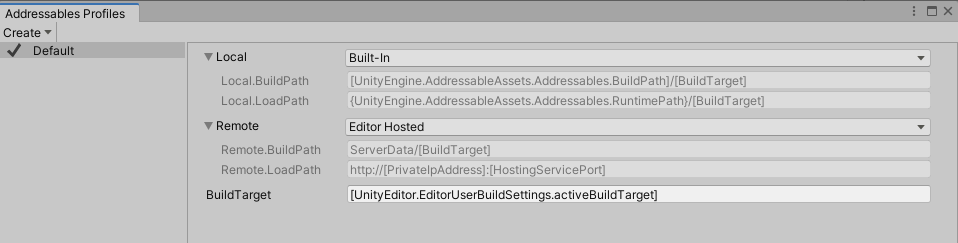
Early in development or when you need to test without access to a host, you might find it convenient to treat all your content as local content. For this situation set the **Local** and **Remote** profile variables using the **Built-In** location.  
在开发初期或需要在无法访问主机的情况下进行测试时，您可能会发现将所有内容视为本地内容很方便。对于这种情况，使用内置位置设置本地和远程配置文件变量。



All content treated as local

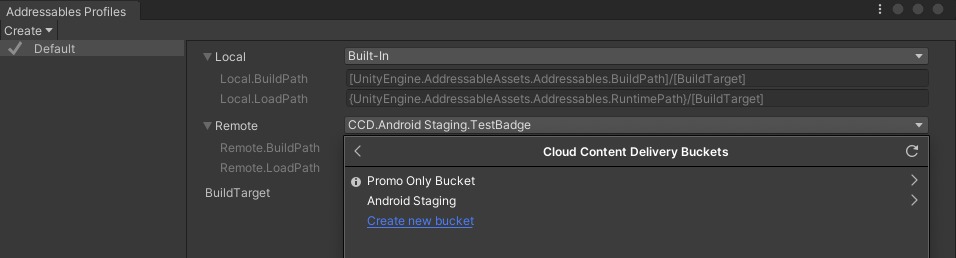
**Builds for local hosting**为本地托管构建

Later, when you set up a host on your local network (or localhost), you will need to change the Load Path for your remote groups to reflect the URL of the host. For example if using [Editor Hosting](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsHostingServices.html), set the **Remote** profile variable using the **Editor Hosting** location.  
稍后，当您在本地网络（或本地主机）上设置主机时，您需要更改远程组的加载路径以反映主机的 URL。例如，如果使用编辑器托管，请使用编辑器托管位置设置远程配置文件变量。

 Remote content served from Editor Hosting

**Builds for CDN**为 CDN 构建

As you get closer to production, you might use a staging server and then, your production Content Delivery Network (CDN). For example if using [Cloud Content Delivery](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressablesCCD.html), set the **Remote** profile variable using the **Cloud Content Delivery** location.  
当您接近生产时，您可能会使用暂存服务器，然后使用您的生产内容交付网络 (CDN)。例如，如果使用 Cloud Content Delivery，请使用 Cloud Content Delivery 位置设置远程配置文件变量。

  
Remote content hosted on the Unity Cloud Content Delivery service

**Other**

Even after release, you might want to use different host URLs for beta testing or other purposes.  
即使在发布之后，您也可能希望使用不同的主机 URL 进行 Beta 测试或其他目的。

Rather than hand configuring the build and load paths every time you build, you can create a different Profile and set the variables appropriately. Then, you can easily switch between Profiles before making a content build without worrying about misconfiguring the paths.  
您不必在每次构建时手动配置构建和加载路径，您可以创建一个不同的配置文件并适当地设置变量。然后，您可以在构建内容之前轻松地在配置文件之间切换，而不必担心路径配置错误。

If you use a script to launch your content builds, then you can use the Addressables API to choose a specific Profile for a build. See [Starting an Addressables build from a script](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildPlayerContent.html#starting-an-addressables-build-from-a-script).  
如果您使用脚本来启动您的内容构建，那么您可以使用 Addressables API 为构建选择特定的配置文件。请参阅从脚本启动 Addressables 构建。

If you have complex URLs, you can reference static fields or properties in your Profile variables that are evaluated at build- or runtime. For example, rather than entering your CCD ProjectID as a string, you could create an Editor class that provides the information as a static property and reference it as, [CCDInfo.ProjectID]. See [Profile variable syntax](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html#profile-variable-syntax) for more information. ([InternalIdTransformFunc](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/TransformInternalId.html) functions provide an additional method of handling complex URL requirements.)  
如果您有复杂的 URL，您可以在您的 Profile 变量中引用在构建或运行时评估的静态字段或属性。例如，与其以字符串形式输入 CCD ProjectID，不如创建一个 Editor 类，以静态属性形式提供信息并将其引用为 [CCDInfo.ProjectID]。有关详细信息，请参阅配置文件变量语法。 （InternalIdTransformFunc 函数提供了一种处理复杂 URL 要求的附加方法。）

##### NOTE

If your remote URL requires cannot be expressed as a static string see [Custom URL evaluation](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/TransformInternalId.html) for information about how you can rewrite the URL for assets, including AssetBundles, at runtime.  
如果您的远程 URL 要求不能表示为静态字符串，请参阅自定义 URL 评估，了解有关如何在运行时重写资产（包括 AssetBundle）的 URL 的信息。

### AssetBundle caching AssetBundle缓存

By default, AssetBundles produced for an Addressables build are cached on the client device after they are downloaded. Cached bundles are only downloaded again if they are updated or if they are deleted from the cache.  
默认情况下，为 Addressables 构建生成的 AssetBundle 在下载后缓存在客户端设备上。缓存的捆绑包只有在更新或从缓存中删除时才会再次下载。

An updated catalog can exclude bundle entries present in an older version of the catalog. When these entries are cached, their data is no longer needed on the device.  
更新的目录可以排除旧版本目录中存在的包条目。当这些条目被缓存时，设备上不再需要它们的数据。

When you have unneeded cache data on the device, you can choose one of three options:  
当设备上有不需要的缓存数据时，您可以选择以下三个选项之一：

* To delete the entire bundle cache, use [Caching.ClearCache](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Caching.ClearCache.html).
* To remove cache entries that are no longer referenced at any time, use [Addressables.CleanBundleCache](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.CleanBundleCache.html). You usually call this function after initializing Addressables (see [Customizing Addressables initialization](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/InitializeAsync.html)) or after loading additional catalogs (see [Managing catalogs at runtime](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadContentCatalogAsync.html)).
* To automatically call [Addressables.CleanBundleCache](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.CleanBundleCache.html) after updating catalogs, use the parameter autoCleanBundleCache in [Addressables.UpdateCatalogs](xref:UnityEngine.AddressableAssets.Addressables.UpdateCatalogs(System.Boolean,System.Collections.Generic.IEnumerable%7bSystem.String%7d,System.Boolean)). See [Checking for content updates at runtime](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html#checking-for-content-updates-at-runtime) for an example script.

If you disable caching for a group, the remote bundles produced for the group are stored in memory when they are downloaded until you unload them or the application exits. The next time the application loads the bundle, Addressables downloads it again.  
如果您禁用组的缓存，则为该组生成的远程包在下载时将存储在内存中，直到您卸载它们或应用程序退出。下次应用程序加载包时，Addressables 会再次下载它。

You can control whether the bundles produced by a group are cached or not with the **Use Asset Bundle Cache** setting under [Advanced Options](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/GroupSettings.html#advanced-options) in the Group Inspector.  
如果您禁用组的缓存，则为该组生成的远程包在下载时将存储在内存中，直到您卸载它们或应用程序退出。下次应用程序加载包时，Addressables 会再次下载它。

See [AssetBundle compression](https://docs.unity3d.com/2019.4/Documentation/Manual/AssetBundles-Cache.html) for additional information about AssetBundle caching. The Addressables system sets the cache-related parameters of the [UnityWebRequests](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Networking.UnityWebRequest.html) it uses to download Addressable bundles based on the group settings.  
有关 AssetBundle 缓存的更多信息，请参阅 AssetBundle 压缩。 Addressables 系统设置 UnityWebRequests 的缓存相关参数，它用于根据组设置下载 Addressable bundle。

Note that there are some limitations for WebGL AssetBundles. For more information, see [Building and running a WebGL project](https://docs.unity3d.com/2019.4/Documentation/Manual/webgl-building.html#AssetBundles).  
请注意，WebGL AssetBundle 有一些限制。有关详细信息，请参阅构建和运行 WebGL 项目。

### Pre-downloading remote content 预下载远程内容

In situations where you want to predownload content so that it is cached on disk and faster to access when the application needs it, you can use the [Addressables.DownloadDependenciesAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.DownloadDependenciesAsync.html) method. This method downloads an Addressable entity and any dependencies as a background task.  
在您想要预下载内容以便将其缓存在磁盘上并在应用程序需要时更快地访问的情况下，您可以使用 Addressables.DownloadDependenciesAsync 方法。此方法下载 Addressable 实体和任何依赖项作为后台任务。

Calling the [Addressables.DownloadDependenciesAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.DownloadDependenciesAsync.html) method loads the dependencies for the address or label that you pass in. Typically, this is the AssetBundle.  
调用 Addressables.DownloadDependenciesAsync 方法加载您传入的地址或标签的依赖项。通常，这是 AssetBundle。

The [AsyncOperationHandle](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.html) struct returned by this call includes a PercentComplete attribute that you can use to monitor and display download progress. You can also have the app wait until the content has loaded.  
此调用返回的 AsyncOperationHandle 结构包括可用于监视和显示下载进度的 PercentComplete 属性。您还可以让应用程序等待内容加载完毕。

#### Regarding PercentComplete 关于完成百分比

PercentComplete takes into account several aspects of the underlying operations being handled by a single AsyncOperationHandle. There may be instances where the progression isn't linear, or some semblance of linear. This can be due to quick operations being weighted the same as operations that will take longer.  
PercentComplete 考虑了由单个 AsyncOperationHandle 处理的基础操作的几个方面。在某些情况下，progression可能不是线性的，也可能是线性的。这可能是由于快速操作与需要更长时间的操作的权重相同。

For example, given an asset you wish to load from a remote location that takes a non-trivial amount of time to download and is reliant on a local bundle as a dependency you'll see your PercentComplete jump to 50% before continuing. This is because the local bundle is able to be loaded much quicker than the remote bundle. However, all the system is aware of is the need for two operations to be complete.  
例如，给定您希望从远程位置加载的资产，该资产需要花费大量时间来下载并且依赖于本地包作为依赖项，您会看到 PercentComplete 跳到 50%，然后再继续。这是因为本地包的加载速度比远程包快得多。但是，系统只知道需要完成两个操作。

If you wish to ask the user for consent prior to download, use [Addressables.GetDownloadSize](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.GetDownloadSize.html) to return how much space is needed to download the content from a given address or label. Note that this takes into account any previously downloaded bundles that are still in Unity's AssetBundle cache.  
如果您希望在下载之前征求用户的同意，请使用 Addressables.GetDownloadSize 返回从给定地址或标签下载内容需要多少空间。请注意，这会考虑仍在 Unity 的 AssetBundle 缓存中的所有先前下载的包。

While it can be advantageous to download assets for your app in advance, there are instances where you might choose not to do so. For example:  
虽然提前为您的应用下载资产可能是有利的，但在某些情况下您可能选择不这样做。例如：

* If your app has a large amount of online content, and you generally expect users to only ever interact with a portion of it.  
  如果您的应用程序包含大量在线内容，并且您通常希望用户只与其中的一部分进行交互。
* You have an app that must be connected online to function. If all your app's content is in small bundles, you might choose to download content as needed.  
  您有一个必须在线连接才能运行的应用程序。如果您应用程序的所有内容都打包成小包，您可以选择根据需要下载内容。

Rather than using the percent complete value to wait until the content is loaded, you can use the preload functionality to show that the download has started, then continue on. This implementation would require a loading or waiting screen to handle instances where the asset has not finished loading by the time it's needed.  
您可以使用预加载功能来显示下载已开始，然后继续，而不是使用完成百分比值等到内容加载完毕。此实现需要一个加载或等待屏幕来处理资产在需要时尚未完成加载的情况。

### Custom URL evaluation 自定义URL转换

There are several scenarios where you might need to customize the path or URL of an Asset (an AssetBundle generally) at runtime. The most common example is creating signed URLs. Another is dynamic host determination.  
在多种情况下，您可能需要在运行时自定义资产（通常是 AssetBundle）的路径或 URL。最常见的示例是创建签名 URL。另一个是动态主机确定。

See [ID transform function](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/TransformInternalId.html) for more information.

### Content catalogs 内容目录

Content Catalogs are the data stores Addressables uses to look up an asset's physical location based on the key(s) provided to the system. By default, Addressables builds a single catalog for all Addressable assets. The catalog is placed in the StreamingAsset folder when you build your application player. The local catalog can access remote as well as local assets, but if you want to update content between full builds of your application, you must also enable the [Build Remote Catalogs](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html#catalog) option.  
内容目录是 Addressables 用于根据提供给系统的键值查找资产物理位置的数据存储。默认情况下，Addressables 为所有 Addressable 资产构建一个目录。当您构建应用程序播放器时，该目录位于 StreamingAsset 文件夹中。本地目录可以访问远程和本地资产，但如果您想在应用程序的完整构建之间更新内容，您还必须启用构建远程目录选项。

The remote catalog is a separate copy of the catalog that you host along with your remote content. Ultimately, Addressables only uses one of these catalogs. If a remote catalog is built and it has a different hash than the local catalog, it is downloaded, cached, and used in place of the built-in local catalog. When you produce a [content update build](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html), the hash is updated and the new remote catalog points to the changed versions of any updated assets.  
远程目录是您与远程内容一起托管的目录的单独副本。最终，Addressables 只使用其中一个目录。如果构建了远程目录并且它具有与本地目录不同的哈希值，则会下载、缓存和使用它来代替内置的本地目录。当您生成内容更新构建时，哈希会更新，新的远程目录指向任何更新资产的更改版本。

##### NOTE

You must enable the remote catalog for the full player build that you publish. Otherwise, the Addressables system does not check for a remote catalog and thus cannot detect any content updates. See [Enabling the remote catalog](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/RemoteContentDistribution.html#enabling-the-remote-catalog).  
您必须为您发布的完整播放器版本启用远程目录。否则，Addressables 系统不会检查远程目录，因此无法检测到任何内容更新。请参阅启用远程目录。

### Loading additional catalogs 加载其他目录

It is possible, however, to specify additional Content Catalogs to be loaded. There are different reasons you might decide loading additional catalogs is right for your project, such as building an art-only project that you want to use across different projects.  
可以指定要加载的其他内容目录。您可能出于不同的原因决定加载其他目录以适合您的项目，例如构建一个您希望在不同项目中使用的纯艺术项目。

Should you find that loading additional catalogs is right for you, there is a method that can assist in this regard, LoadContentCatalogAsync.  
如果您发现加载其他目录适合您，有一种方法可以在这方面提供帮助，即 LoadContentCatalogAsync。

For LoadContentCatalogAsync, all that is required is for you to supply the location of the catalog you wish to load. However, this alone does not use catalog caching, so be careful if you're loading a catalog from a remote location. You will incur that WebRequest every time you need to load that catalog.  
对于 LoadContentCatalogAsync，您只需提供要加载的catalog的位置即可。但是，仅此一项不会使用目录缓存，因此如果您从远程位置加载目录，请务必小心。每次需要加载该目录时，您都会执行 WebRequest。

To help prevent you from needing to download a remote catalog every time, if you provide a .hash file with the hash of the catalog alongside the catalog you're loading, we can use this to properly cache your Content Catalog.  
为帮助避免您每次都需要下载远程目录，如果您在加载的catalog旁边提供一个包含catalog哈希的 .hash 文件，我们可以使用它来正确缓存您的内容目录。

##### NOTE

* The hash file does need to be in the same location and have the same name as your catalog. The only difference to the path should be the extension.  
  Hash文件需要与您的Catalog位于同一位置并具有相同的名称。路径的唯一区别应该是扩展名。
* You'll notice this method comes with a parameter autoReleaseHandle. In order for the system to download a new remote catalog, any prior calls to LoadContentCatalogAsync that point to the catalog you're attempting to load need to be released. Otherwise, the system picks up the Content Catalog load operation from our operation cache. If the cached operation is picked up, the new remote catalog is not downloaded. If set to true, the parameter autoReleaseHandle can ensure that the operation doesn't stick around in our operation cache after completing.  
  您会注意到此方法带有一个参数autoReleaseHandle。为了让系统下载新的远程目录，之前对 LoadContentCatalogAsync 的任何指向您尝试加载的目录的调用都需要释放。否则，系统会从我们的操作缓存中获取内容目录加载操作。如果获取到了缓存的操作，则不会下载新的远程目录。如果设置为 true，参数 autoReleaseHandle 可以确保操作完成后不会停留在我们的操作缓存中。

## Content update builds 构建内容更新

The Addressables package includes tools that you can use to reduce the size of updates to the content you distribute remotely.  
Addressables 包中包含一些工具，您可以使用这些工具来减少远程分发的内容的更新大小。

The content update tools include:

* [**Check for Content Update Restrictions**](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html#check-for-content-update-restrictions-tool) tool: prepares your group organization for a content update build based on group settings  
  检查内容更新限制工具：让您的小组组织为基于小组设置的内容更新构建做好准备
* [**Update a Previous Build**](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html#building-content-updates) script: a build script that performs the content update build

##### IMPORTANT

You must save the **addressables\_content\_state.bin** file produced by the Default Build Script for each build that you intend to update in the future. This file is updated every time you run the build script. Be sure to save the version produced for the content build that you publish.  
您必须为将来要更新的每个构建保存由默认构建脚本生成的 addressables\_content\_state.bin 文件。每次运行构建脚本时都会更新此文件。请务必保存为您发布的内容构建生成的版本。

For information on how to set up your Addressable groups for content updates, see [Group settings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html#settings).

For information on how to perform a content update build, see [Building content updates](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html#building-content-updates).

For general information about how content updates work, including examples, see [Overview](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html#overview).

##### NOTE

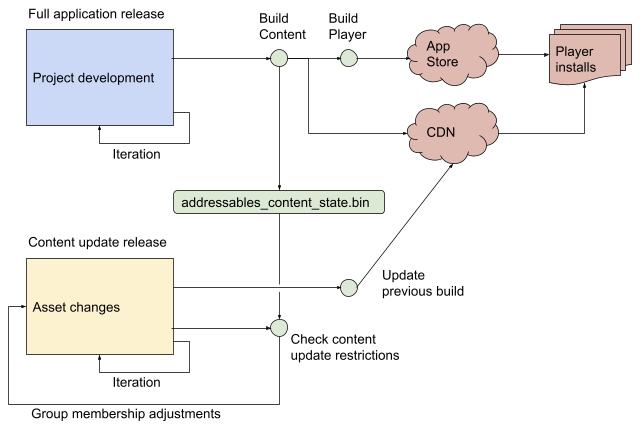
On platforms that provide their own patching systems (such as Switch or Steam) or that do not support remote content distribution, do not use content update builds. Every build of your game should be a complete fresh content build. (In this case you can discard or ignore the addressables\_content\_state.bin file that is generated after each build for the platform.)  
在提供自己的补丁系统（例如 Switch 或 Steam）或不支持远程内容分发的平台上，请勿使用内容更新构建。您游戏的每个版本都应该是一个完整的新内容版本。 （在这种情况下，您可以丢弃或忽略每次为平台构建后生成的 addressables\_content\_state.bin 文件。）

### Overview

When you distribute content remotely, you can make content changes without needing to rebuild and republish your entire application. When the Addressables system initializes at runtime, it checks for an updated content catalog. If one exists, the system downloads the new catalog and, when it loads assets, downloads the newer versions of all your AssetBundles.  
当您远程分发内容时，您可以更改内容而无需重建和重新发布整个应用程序。当 Addressables 系统在运行时初始化时，它会检查更新的内容目录。如果存在，系统会下载新目录，并在加载资产时下载所有 AssetBundle 的更新版本。

However, when you rebuild all of your content with a new content catalog, installed players must also redownload all of your remote AssetBundles, whether the assets in them have changed or not. If you have a large amount of content, redownloading everything can take a significant amount of time and may hurt player retention. To make this process more efficient, the Addressables package provides tools that you can run to identify changed assets and to produce a content update build.  
但是，当您使用新的内容目录**重建**所有内容时，已安装的播放器必须重新下载所有远程 AssetBundle，无论其中的资产是否已更改。如果您有大量内容，重新下载所有内容可能会花费大量时间，并且可能会影响玩家留存率。为了使此过程更加高效，Addressables 包提供了一些工具，您可以运行这些工具来识别已更改的资产并生成内容更新版本。

The following diagram illustrates how you can use the Addressables tools to produce smaller content updates that only require your players to download new or changed content:  
下图说明了如何使用 Addressables 工具来生成较小的内容更新，这些更新只需要您的播放器下载新的或更改的内容：

 The workflow for reducing the size of content updates

When you release your full application, you first build your Addressables content, as normal, and then make a player build. The player build contains your local AssetBundles and you upload your remote AssetBundles to your Content Delivery Network (CDN) or other hosting service.  
当您发布完整的应用程序时，您首先像往常一样构建您的 Addressables 内容，然后进行播放器构建。播放器构建包含您的本地 AssetBundle，您将远程 AssetBundle 上传到您的内容交付网络 (CDN) 或其他托管服务。

The Default Build Script that produces your Addressables content build always creates the addressables\_content\_state.bin file, which is required to efficiently publish content-only updates. You must save this file for each published full application release (on every platform).  
生成 Addressables 内容构建的默认构建脚本始终创建 addressables\_content\_state.bin 文件，这是有效发布仅内容更新所必需的。您必须为每个已发布的完整应用程序版本（在每个平台上）保存此文件。

Between full application releases, which require your users to download and install a new player build, you can make changes to your Addressable assets in the project. (Since AssetBundles do not include code, do not make code changes in the version of your project that you use to develop your asset changes.) You can change both local and remote assets.  
在（要求用户重新下载并安装）新的完整应用程序发布之前，您可以对项目中的可寻址资产进行更改。 （由于 AssetBundle 不包含代码，因此不要在用于开发资产更改的项目版本中更改代码。）您可以更改本地和远程资产。

When you are ready to publish a content update, run the **Check Content Update Restrictions** tool. This tool examines the addressables\_content\_state.bin file and moves changed assets to a new remote group, according to the settings of the group they are in.  
当您准备好发布内容更新时，请运行检查内容更新限制工具。此工具检查 addressables\_content\_state.bin 文件，并根据它们所在的组的设置将更改的资产移动到新的远程组。

To build the updated AssetBundles, run the **Update a Previous Build** script. This tool also uses the addressables\_content\_state.bin file. It rebuilds all of your content, but produces a modified catalog that accesses unchanged content from their original AssetBundles and changed content from the new AssetBundles.  
要构建更新后的 AssetBundle，请运行 Update a Previous Build 脚本。此工具还使用 addressables\_content\_state.bin 文件。它会重建您的所有内容，但会生成一个修改后的catalog文件，该catalog文件可以访问其原始 AssetBundle 中未更改的内容和新 AssetBundle 中已更改的内容。

The final step is to upload the updated content to your CDN. (You can upload all the new AssetBundles produced or just those with changed names -- bundles that haven't changed use the same names as the originals and will overwrite them.)  
最后一步是将更新的内容上传到您的 CDN。 （您可以上传所有新生成的 AssetBundle 或仅上传名称已更改的资源包——未更改的资源包使用与原始资源包相同的名称并将覆盖它们。）

You can make additional content updates following the same process. Always use the addressables\_content\_state.bin file from your original release.  
您可以按照相同的过程进行其他内容更新。始终使用原始版本中的 addressables\_content\_state.bin 文件。

See [Building content updates](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html#building-content-updates) for step-by-step instructions.

#### When a full rebuild is required 何时需要重新构建

Addressables can only distribute content, not code. As such, a code change generally requires a fresh player build, and usually a fresh build of content. Although a new player build can sometimes reuse old, existing content from a CDN, you must carefully analyze whether the type trees in the existing AssetBundles are compatible with your new code. This is advanced territory to explore carefully.  
Addressables 只能分发内容，不能分发代码。因此，代码更改通常需要全新的播放器构建，并且通常需要全新的内容构建。尽管新播放器构建有​​时可以重用来自 CDN 的旧内容，但您必须仔细分析现有 AssetBundle 中的类型树是否与您的新代码兼容。这是需要仔细探索的高级领域。

Note that Addressables itself is code, so updating Addressables or Unity version likely requires that you create a new player build and fresh content builds.  
请注意，Addressables 本身是代码，因此更新 Addressables 或 Unity 版本可能需要您创建新的播放器构建和新的内容构建。

### Settings 设置

To publish content updates, your application must already use a remote catalog and host its remote content on an accessible server. See [Enabling remote distribution](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/RemoteContentDistribution.html#enabling-remote-distribution) for information about setting up content hosting and distribution.  
要发布内容更新，您的应用程序必须已经使用远程目录并将其远程内容托管在可访问的服务器上。有关设置内容托管和分发的信息，请参阅启用远程分发。

In addition to enabling remote content distribution, you should also consider how to set each group's **Update Restriction** settings. These settings determine how the **Check for Content Update Restriction** tool treats changed content in your groups. Choose appropriate settings to help minimize the download size of your content updates. See [Group Update Restriction settings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html#group-update-restriction-settings).  
除了启用远程内容分发之外，您还应该考虑如何设置每个组的更新限制设置。这些设置决定了“检查内容更新限制”工具如何处理您组中更改的内容。选择适当的设置以帮助最小化内容更新的下载大小。请参阅组更新限制设置。

Another setting to consider if you want to update content on the fly (rather than at application startup), is the **Unique Bundle IDs** setting. Enabling this option can make it easier to load updated AssetBundles in the middle of an application session, but typically makes builds slower and updates larger. See [Unique Bundle IDs setting](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html#unique-bundle-ids-setting).  
如果您想动态更新内容（而不是在应用程序启动时），需要考虑的另一个设置是唯一捆绑 ID 设置。启用此选项可以更轻松地在应用程序会话中间加载更新的 AssetBundle，但通常会使构建速度变慢并且更新更大。请参阅唯一捆绑 ID 设置。

#### Group Update Restriction settings 组更新限制设置

For each group in your project, set the **Update Restriction** setting to either:  
对于项目中的每个组，将更新限制设置为：

* Cannot Change Post Release: Static content that you expect to update infrequently, if at all. All local content should use this setting.  
  发布后无法更改：您希望不经常更新（如果有的话）的静态内容。所有本地内容都应使用此设置。
* Can Change Post Release: Dynamic content that you expect to update often.  
  发布后可以更改：您希望经常更新的动态内容。

Choose the setting based on the type of content in a group and how frequently you expect to update that content (between full player builds of your application).  
根据组中的内容类型以及您希望更新该内容的频率（在应用程序的完整播放器构建之间）选择设置。

You can change content in a group no matter which setting you choose. The difference is how the **Check for Content Update** and **Update Previous Build** tools treat the assets in the group and ultimately, how the installed applications access the updated content.  
无论选择哪种设置，您都可以更改组中的内容。不同之处在于检查内容更新和更新以前的构建工具如何处理组中的资产，以及最终安装的应用程序如何访问更新的内容。

##### IMPORTANT

Do NOT change the **Update Restriction** setting of a group unless you are performing a full build. If you change your group settings before a content update, Addressables cannot generate the correct changes needed for the update build.  
除非您正在执行完整构建，否则不要更改组的更新限制设置。如果您在内容更新之前更改了组设置，Addressables 将无法生成更新构建所需的正确更改。

##### Cannot Change Post Release (static content) 无法更改发布后（静态内容）

When you set a group as **Cannot Change Post Release**, the **Check for Content Updates** tool moves any changed assets to a new group, which is set to build and load from your remote paths. When you subsequently build the updated content with the **Update a Previous Build** tool, it sets up the remote catalog so that the changed assets are accessed from the new bundles, but the unchanged assets are still accessed from the original bundles.  
当您将组设置为发布后无法更改时，检查内容更新工具会将任何更改的资产移动到新组，该组设置为从您的远程路径构建和加载。当您随后使用 Update a Previous Build 工具构建更新后的内容时，它会设置远程catalog，以便从新捆绑包访问更改的资产，但仍可从原始捆绑包访问未更改的资产。

###### NOTE

Although the update build produces versions of the original bundles without the changed assets, installed applications do not download these bundles unless the locally cached version is deleted for some reason.  
尽管更新版本生成了原始包的版本而没有更改资产，但已安装的应用程序不会下载这些包，除非本地缓存的版本因某种原因被删除。

Organize content that you don't expect to update frequently in groups set to **Cannot Change Post Release.** You can safely set up these groups to produce fewer, larger bundles since your users usually won't need to download these bundles more than once.  
将您不希望经常更新的内容组织在设置为“无法更改发布后”的组中。您可以安全地设置这些组以生成更少、更大的捆绑包，因为您的用户通常不需要多次下载这些捆绑包。

Any groups that you intend to load from the local load path should always be set to **Cannot Change Post Release**. Likewise, any groups that produce large, remote bundles should also be set to **Cannot Change Post Release** so that your users only need to download the changed assets if you do end up changing assets in these groups.  
您打算从本地加载路径加载的任何组都应始终设置为 Cannot Change Post Release。同样，任何产生大型远程捆绑包的组也应设置为无法更改发布后，以便您的用户仅在您最终更改这些组中的资产时才需要下载更改的资产。

##### Can Change Post Release (dynamic content) 可以更改发布后（动态内容）

When you set a group as **Can Change Post Release**, then a content update rebuilds the entire bundle if any assets inside the group have changed. The **Update a Previous Build** script sets the catalog up so that installed applications load all assets in the group from the new bundles.  
当您将组设置为 Can Change Post Release 时，如果组内的任何资产发生更改，内容更新将重建整个包。 Update a Previous Build 脚本会设置目录，以便已安装的应用程序从新包中加载组中的所有资产。

Organize content you expect to change frequently in groups set to **Can Change Post Release**. Since all the assets in these groups are republished when any single asset changes, you should typically set up these groups to produce smaller bundles containing fewer assets.  
在设置为“发布后可以更改”的组中组织您希望经常更改的内容。由于这些组中的所有资产都会在任何单个资产更改时重新发布，因此您通常应该设置这些组以生成包含较少资产的较小包。

#### Unique Bundle IDs setting 唯一捆绑 ID 设置

The [Addressable Asset settings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html) contain an option, **Unique Bundle IDs**, that affect content update builds. You can evaluate whether you need to enable this option if you run into AssetBundle ID conflicts when updating your application catalog at runtime.  
Addressable Asset 设置包含一个选项，Unique Bundle IDs，影响内容更新构建。如果在运行时更新应用程序目录时遇到 AssetBundle ID 冲突，您可以评估是否需要启用此选项。

Enabling the **Unique Bundle IDs** option allows you to load a changed version of an AssetBundle while the original bundle is still in memory. Building your AssetBundles with unique internal IDs makes it easier to update content at runtime without running into AssetBundle ID conflicts.  
启用 Unique Bundle IDs 选项允许您在原始包仍在内存中时加载 AssetBundle 的更改版本。使用唯一的内部 ID 构建 AssetBundle 可以更轻松地在运行时更新内容，而不会遇到 AssetBundle ID 冲突。

The option is not without drawbacks, however. When enabled, any AssetBundles containing assets that reference a changed asset must also be rebuilt. More bundles must be updated for a content update and all builds are slower.  
然而，该选项并非没有缺点。启用后，还必须重建任何包含引用已更改资产的资产的 AssetBundle。必须更新更多捆绑包才能进行内容更新，并且所有构建速度都较慢。

You typically only need to use unique bundle IDs when you update content catalogs after the Addressable system has already initialized and you have started loading assets.  
在 Addressable 系统已经初始化并且您已经开始加载资产之后更新内容目录时，您才会需要使用唯一的包 ID。

You can avoid AssetBundle loading conflicts and the need to enable unique IDs using one of the following methods:  
您可以使用以下方法之一避免 AssetBundle 加载冲突和启用唯一 ID 的需要：

* Update the content catalog as part of Addressables initialization. By default, Addressables checks for a new catalog at initialization (as long as you don't enable the [Disable Catalog Update on Startup](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html#catalog) option in your Addressable Asset settings). Choosing this method does preclude updating your application content in mid-session.  
  更新内容目录作为Addressables 初始化的一部分。默认情况下，Addressables 会在初始化时检查新目录（只要您未在 Addressable Asset settings中启用[Disable Catalog Update on Startup](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html#catalog) 选项）。选择此方法确实会排除在会话中期更新您的应用程序内容。
* Unload all remote AssetBundles before updating the content catalog. Unloading all your remote bundles and assets also avoids bundle name conflicts, but could interrupt your user's session while they wait for the new content to load.  
  在更新内容目录之前卸载所有远程AssetBundle。卸载所有远程包和资产也可以避免包名称冲突，但可能会在用户等待新内容加载时中断他们的会话。

### Building content updates 未完待续

To build a content update, run the **Update a Previous Build** script:

1. Run the [Check for Content Update Restrictions](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html#check-for-content-update-restrictions-tool) tool.
2. Open the **Addressables Groups** window in the Unity Editor (**Window** > **Asset Management** > **Addressables** > **Groups**).
3. From the **Build** menu on the toolbar, run the **Update a Previous Build** script.
4. In the **Build Data File** dialog that opens, again select the addressables\_content\_state.bin file produced by the build you are updating.
5. Click **Open** to start the content update build.

The build generates a content catalog, a hash file, and AssetBundles.

The generated content catalog has the same name as the catalog in the original application build, overwriting the old catalog and hash file. The application loads the hash file at runtime to determine if a new catalog is available. The system loads unmodified assets from existing bundles that were shipped with the application or that the application has already downloaded.

The system uses the content version string and location information from the addressables\_content\_state.bin file to create the AssetBundles. Asset bundles that do not contain updated content are written using the same file names as those in the build selected for the update. If an AssetBundle contains updated content, a new bundle is generated that contains the updated content, with a new file name so that it can coexist with the original on your content hosting service. Only AssetBundles with new file names must be copied to the location that hosts your content (though you can safely upload them all).

The system also builds AssetBundles for content that cannot change, such as any local AssetBundles, but you do not need to upload them to the content hosting location, as no Addressables Asset entries reference them.

Note that you should not change the build scripts between building a new player and making content updates (e.g., player code, Addressables). This could cause unpredictable behavior in your application.

Additionally, if you delete the local content bundles created by your Addressables build from the Project Library folder, attempts to load Assets in those bundles fail when you run your game or application in the Editor and use the **Use Existing Build (requires built groups)** Play Mode script.

#### Check for Content Update Restrictions tool

The **Check for Content Update Restrictions** tool prepares your group organization for a content update build. The tool examines the addressables\_content\_state.bin file and and group settings. If a group's **Update Restrictions** option was set to **Cannot Change Post Release** in the previous build, the tool moves any changed assets to a new remote group. When you create the update build, the new catalog maps the changed assets to their new, remote AssetBundles, while still mapping the unchanged assets to their original AssetBundles. Checking for content update restrictions does not modify groups set to **Can Change Post Release**.

To run the tool:

1. Open the **Addressables Groups** window in the Unity Editor (**Window** > **Asset Management** > **Addressables** > **Groups**).
2. In the groups window, run the **Check for Content Update Restrictions** from the toolbar **Tools** menu.
3. In the **Build Data File** dialog that opens, select the addressables\_content\_state.bin file produced by the build you are updating. (By default, this is located in the Assets/AddressableAssetsData/<platform> Project directory, where <platform> is your target platform.)
4. Click **Open** to select the file and launch the tool.
5. Review the group changes made by the tool, if desired. You can change the names of any new remote groups the tool created, but moving assets to different groups can have unintended consequences.

**Important**: Before you run the **Check for Content Update Restrictions** tool, you should make a branch with your version control system. The tool rearranges your asset groups in a way suited for updating content. Branching ensures that next time you ship a full player build, you can return to your preferred content arrangement.

### Checking for content updates at runtime

You can add a custom script to periodically check whether there are new Addressables content updates. Use the following function call to start the update:

[public static AsyncOperationHandle<List<string>> CheckForCatalogUpdates(bool autoReleaseHandle = true)](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.CheckForCatalogUpdates.html)

where List<string> contains the list of modified locator IDs. You can filter this list to only update specific IDs, or pass it entirely into the UpdateCatalogs API.

If there is new content, you can either present the user with a button to perform the update, or do it automatically. Note that it is up to the developer to make sure that stale Assets are released.

The list of catalogs can be null and if so, the following script updates all catalogs that need an update:

[public static AsyncOperationHandle<List<IResourceLocator>> UpdateCatalogs(IEnumerable<string> catalogs = null, bool autoReleaseHandle = true)](xref:UnityEngine.AddressableAssets.Addressables.UpdateCatalogs(System.Collections.Generic.IEnumerable%7bSystem.String%7d,System.Boolean))

The return value is the list of updated locators.

You may also want to remove any bundle cache entries that are no longer referenced as a result of updating the catalogs. If so, use this version of the UpdateCatalogs API instead where you can enable the additional parameter autoCleanBundleCache to remove any unneeded cache data:

[public static AsyncOperationHandle<List<IResourceLocator>> UpdateCatalogs(bool autoCleanBundleCache, IEnumerable<string> catalogs = null, bool autoReleaseHandle = true)](xref:UnityEngine.AddressableAssets.Addressables.UpdateCatalogs(System.Boolean,System.Collections.Generic.IEnumerable%7bSystem.String%7d,System.Boolean))

See [AssetBundle caching](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/RemoteContentDistribution.html#assetbundle-caching) for additional information about the bundle cache.

See [Unique Bundle IDs setting](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html#unique-bundle-ids-setting) for additional information about updating content at runtime.

### Content update examples

The following discussion walks through a hypothetical example to illustrate how Addressable content is handled during a content update. In this example, consider a shipped application built with the following Addressables groups:

| **Local\_Static** | **Remote\_Static** | **Remote\_NonStatic** |
| --- | --- | --- |
| AssetA | AssetL | AssetX |
| AssetB | AssetM | AssetY |
| AssetC | AssetN | AssetZ |

Note that Local\_Static and Remote\_Static are part of the Cannot Change Post Release groups.

Since this version is live, existing players have Local\_Static on their devices, and potentially have either or both of the remote bundles cached locally.

If you modify one Asset from each group (AssetA, AssetL, and AssetX), then run **Check for Content Update Restrictions**, the results in your local Addressable settings are now:

| **Local\_Static** | **Remote\_Static** | **Remote\_NonStatic** | **content\_update\_group (non-static)** |
| --- | --- | --- | --- |
|  |  | AssetX | AssetA |
| AssetB | AssetM | AssetY | AssetL |
| AssetC | AssetN | AssetZ |  |

Note that the prepare operation actually edits the Cannot Change Post Release groups, which may seem counterintuitive. The key, however, is that the system builds the above layout, but discards the build results for any such groups. As such, you end up with the following from a player's perspective:

| **Local\_Static** |
| --- |
| AssetA |
| AssetB |
| AssetC |

The Local\_Static bundle is already on player devices, which you can't change. This old version of AssetA is no longer referenced. Instead, it is stuck on player devices as dead data.

| **Remote\_Static** |
| --- |
| AssetL |
| AssetM |
| AssetN |

The Remote\_Static bundle is unchanged. If it is not already cached on a player's device, it will download when AssetM or AssetN is requested. Like AssetA, this old version of AssetL is no longer referenced.

| **Remote\_NonStatic (old)** |
| --- |
| AssetX |
| AssetY |
| AssetZ |

The Remote\_NonStatic bundle is now old. You can delete it from the server or leave it there; either way it will not be downloaded from this point forward. If cached, it remains on player devices indefinitely unless you remove it. See [AssetBundle caching](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/RemoteContentDistribution.html#assetbundle-caching) for more information. Like AssetA and AssetL, this old version of AssetX is no longer referenced.

| **Remote\_NonStatic (new)** |
| --- |
| AssetX |
| AssetY |
| AssetZ |

The old Remote\_NonStatic bundle is replaced with a new version, distinguished by its hash file. The modified version of AssetX is updated with this new bundle.

| **content\_update\_group** |
| --- |
| AssetA |
| AssetL |

The content\_update\_group bundle consists of the modified Assets that will be referenced moving forward.

Note that the example above has the following implications:

1. Any changed local Assets remain unused on the user's device forever.
2. If the user already cached a non-static bundle, they will need to redownload the bundle, including the unchanged Assets (in this instance, for example, AssetY and AssetZ). Ideally, the user has not cached the bundle, in which case they simply need to download the new Remote\_NonStatic bundle.
3. If the user has already cached the Static\_Remote bundle, they only need to download the updated asset (in this instance, AssetL via content\_update\_group). This is ideal in this case. If the user has not cached the bundle, they must download both the new AssetL via content\_update\_group and the now-defunct AssetL via the untouched Remote\_Static bundle. Regardless of the initial cache state, at some point the user will have the defunct AssetL on their device, cached indefinitely despite never being accessed.

The best setup for your remote content will depend on your specific use case.

### How Content Update Handles Dependencies

Directly changing an asset is not the only way to have it flagged as needing to be rebuilt as part of a content update. Changing an asset's dependencies is a less obvious factor that gets taken into account when building an update.

As an example, consider the Local\_Static group from the example above:

| **Local\_Static** |
| --- |
| AssetA |
| AssetB |
| AssetC |

but now suppose the assets in this group have a dependency chain that looks like this: AssetA depends on Dependency1, which depends on Dependency2, AssetB depends on Dependency2, and AssetC depends on Dependency3 and all three dependencies are a mix of Addressable and non-Addressable assets.

Now, if only Dependency1 is changed and Check For Content Update Restriction is run, the resulting project structure looks like:

| **Local\_Static** | **content\_update\_group** |
| --- | --- |
|  | AssetA |
| AssetB |  |
| AssetC |  |

If only Dependency2 is changed:

| **Local\_Static** | **content\_update\_group** |
| --- | --- |
|  | AssetA |
|  | AssetB |
| AssetC |  |

Finally, if only Dependency3 is changed:

| **Local\_Static** | **content\_update\_group** |
| --- | --- |
| AssetA |  |
| AssetB |  |
|  | AssetC |

This is because when a dependency is changed the entire dependency tree needs to be rebuilt.

Let's take a look at one more example with the following dependency tree. AssetA depends on AssetB, which depends on Dependency2, AssetB depends on Dependency2, and AssetC depends on Dependency3. Now, if Dependency2 is changed, the project structure looks like:

| **Local\_Static** | **content\_update\_group** |
| --- | --- |
|  | AssetA |
|  | AssetB |
| AssetC |  |

because AssetA relies on AssetB and AssetB relies on Dependency2. Since the entire chain needs to be rebuilt both AssetA and AssetB will get put into the **content\_update\_group**.

## Addressable Asset system with Cloud Content Delivery

You can use the [Addressable asset system](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/index.html) in conjunction with [Unity Cloud Content Delivery](https://docs.unity3d.com/Manual/UnityCCD.html) (CCD) to distribute your remote Addressables content.

**Note**: The purpose of this page is to describe how to link the concepts of Addressable Assets to CCD, and isn't meant to be an in-depth discussion of these ideas. Before you read this page, make sure you are familiar with both the Addressable system and Cloud Content Delivery.

To set up Addressable assets to work with CCD:

1. [Configure a profile to include your CCD URL](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressablesCCD.html#configure-profile-with-ccd-url)
2. [Build your AssetBundles, then upload them to CCD](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressablesCCD.html#build-upload-release)

See [Getting Started](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsGettingStarted.html) for information about installing and implementing the Addressables package.

See [Upgrading to the Addressables System](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsMigrationGuide.html) for information about integrating Addressables in an existing Unity Project.

See [Remote content distribution](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/RemoteContentDistribution.html) for information on how to set up your Project so that you can host your Addressables content on a remote server.

See [Unity Cloud Content Delivery](https://docs.unity3d.com/Manual/UnityCCD.html) for more information about CCD.

### Configure profile with CCD URL

##### TIP

The BuildPath and LoadPath variables stored in [Profiles](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html) specify where the Addressables system creates your build artifacts and where it looks for your assets at runtime. Configure the remote paths to work with CCD. (Leave the local paths with their standard, default values, unless you have a specific reason to change them.)

If necessary, create a new profile for publishing builds to CCD on the [Profiles](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html) window. Configure the remote path variables in this profile to access your content at the correct URL.

You can set the remote BuildPath to a convenient value. If you have multiple profiles, consider using a unique build path for each of them so that the build artifacts do not get mixed together, especially if you are hosting them from a different remote URL.

Set the remote LoadPath to one of the following two paths:

* If you publish content using a badge:

https://(ProjectID).client-api.unity3dusercontent.com/client\_api/v1/buckets/(BucketID)/release\_by\_badge/(BadgeName)/entry\_by\_path/content/?path=

* If you publish using a release:

https://(ProjectID).client-api.unity3dusercontent.com/client\_api/v1/buckets/(BucketID)/releases/(ReleaseID)/entry\_by\_path/content/?path=

where:

* (ProjectID) is your CCD project's ID string
* (BucketID) is the Bucket ID string for a CCD bucket within your project
* (ReleaseID) is the ID of a specific release within a bucket
* (BadgeName) is the name of the specific CCD badge

See [Profiles](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html) for information about how to create and edit profiles.

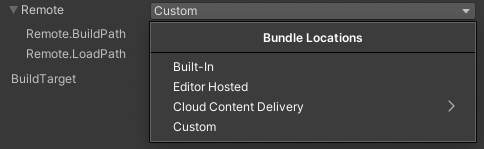
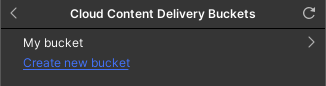
##### IMPORTANT

You must perform a full rebuild your Addressables content when you change the remote load path.

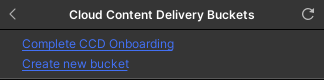
#### Using the Cloud Content Delivery Bundle Location Option in a Profile

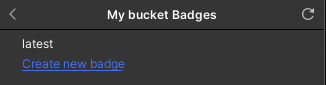
If your project is set up to use the [Unity Cloud Content Delivery](https://docs.unity3d.com/Manual/UnityCCD.html) service, you can set the profile's remote path pair to publish content to a designated bucket and badge.

To set up a Profile variable to use the CCD bundle location:

1. Open the Profile window (menu: **Window > Asset Management > Addressables > Profiles**).
2. Select the profile to change.
3. Change the **Remote** variable to use the **Cloud Content Delivery** **Bundle Location**.  
     
   Cloud Content Delivery Bundle Location Option
4. Choose the Bucket to use.  
     
   Cloud Content Delivery Bundle Location Option

##### NOTE

If no buckets are present, you will be shown this window before continuing.  
  
Cloud Content Delivery Bundle Location Option

1. Choose the Badge.  
     
   Cloud Content Delivery Bundle Location Option

Make this the active profile when building content for delivey with CCD.

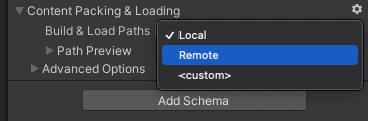
##### IMPORTANT

This feature requires the Content Delivery Management API package, which is currently in an experimental state.

See [Profiles](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html) for information about how to modify profiles.

### Configure groups with CCD URL

Configure groups to use **Remote** as their **Build & Load Path** in the inspector window.

  
Group Build & Load Paths

See [Groups](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html) for information about how to modify groups.

### Build, upload and release Addressable content

#### Using to CCD Dashboard/CLI

To generate and upload Addressable content to your CCD project:

1. Set the profile you have set up for CCD as the active profile.
2. Build your Addressables content.
   * If you are making a full content build, see [Building your Addressable content](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildingContent.html).
   * If your are updating an existing build with modified remote content, see [Building for content updates](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html#building-content-updates).
3. Upload the files created at the remote build path using the [CCD dashboard](https://docs.unity.com/ccd/Content/UnityCCDDashboard.htm) or [command-line interface](https://docs.unity.com/ccd/Content/UnityCCDCLI.htm).
4. Create a release and update the badge using the [CCD dashboard](https://docs.unity.com/ccd/Content/UnityCCDDashboard.htm) or [command-line interface](https://docs.unity.com/ccd/Content/UnityCCDCLI.htm).

Building your Addressable content generates a content catalog (.json), a hash file (.hash), and one or more AssetBundle (.bundle) files. Upload these files to the bucket corresponding to the URL used in your profile load path.

If you have made changes to local content, you must create a new Player build.

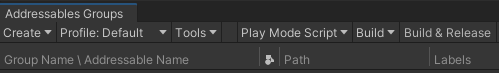
If you are using the Unity Cloud Build service, you can configure your cloud builds to send content to CCD. See [Using Addressables in Unity Cloud Build](https://docs.unity3d.com/2019.4/Documentation/Manual/UnityCloudBuildAddressables.html) for information.

#### Using CCD Management package

To generate, upload, and release Addressable content to your CCD project:

1. Open the Groups window (menu: **Window > Asset Management > Addressables > Groups**).
2. Use the **Build & Release** option.

The CCD Management package will use the default build script behavior to generate the Addressable bundles. Then, all groups associated with a path pair that is connected to a CCD bucket and badge via the drop-down window will have their generated bundles uploaded by the management package to those remote target. Finally, the management package will a create release for those remote target and update their badge.

  
Build & Release option

## Loading from Multiple Projects

Should your situation require a multi-project workflow, such as a large project broken up across multiple Unity projects that have the Addressables package installed, we have [Addressables.LoadContentCatalogAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadContentCatalogAsync.html) to link together code and content across the various projects. Studios with teams that works on many facets of an application simultaneously may find benefit with this workflow.

### Setting up multiple projects

The main items to note for a multi-project setup is to make sure:

1. Each project uses the same version of the Unity Editor
2. Each project uses the same version of the Addressables package

From there projects can contain whatever you see fit for your given situation. One of your projects must be your "main project" or "source project". This is the project that you'll actually build and deploy your game binaries from. Typically, this source project is primarily comprised of code and very little to no content. The main piece of content that you would want in the primary project would be a bootstrap scene at minimum. It may also be desirable to include any scenes that need to be local for performance purposes before any AssetBundles have had a chance to be downloaded and cached.

Secondary projects are, in most cases, the exact opposite. Mostly content and little to no code. **These projects need to have all remote Addressable Groups and Build Remote Catalog turned on.** Any local data built into these projects cannot be loaded in your source project's application. Non-critical scenes can live in these projects and be downloaded by the primary project when requested.

#### The Typical Workflow

Once you have your projects setup, the workflow generally is as follows:

1. Build remote content for all secondary projects
2. Build Addressables content for source project
3. Start source project Play Mode or build source project binaries
4. In source project, use [Addressables.LoadContentCatalogAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadContentCatalogAsync.html) to load the remote catalogs of your other various projects
5. Proceed with game runtime as normal. Now that the catalogs are loaded Addressables is able to load assets from any of these locations.

In regards to the source project, it should be noted that it may be worth having some minimal amount of content built locally with that project. Each project is unique, and so has unique needs, but having a small set of content needed to run your game in the event of internet connection issues or other various problems may be advisable.

#### Handling Shaders

Addressables builds a Unity built in shader bundle for each set of Addressables player data that gets built. This means that when multiple AssetBundles are loaded that were built in secondary projects, there could be multiple built in shader bundles loaded at the same time.

Depending on your specific situation, you may need to utilize the Shader Bundle Naming Prefix on the AddressableAssetSettings object. Each built in shader bundle needs to be named different from others built in your other projects. If they're not named differently you'll get The AssetBundle [bundle] can't be loaded because another AssetBundle with the same files is already loaded. errors.

# Using Addressables at runtime 在运行时使用可寻址对象

Once you have your Addressable assets organized into groups and built into AssetBundles, you must still load, instantiate, and, in the end release them at runtime.  
将可寻址资产组织成组并内置到 AssetBundle 中后，您仍必须在运行时加载、实例化并最终释放它们。

Addressables uses a reference counting system to make sure that assets are only kept in memory while they are needed. See [Memory management](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/MemoryManagement.html) for more information about reference counting and how you can minimize the amount of memory used by assets at any given time.  
可寻址对象使用引用计数系统来确保资产仅在需要时保留在内存中。有关引用计数以及如何在任何给定时间最大程度地减少资产使用的内存量的详细信息，请参阅[内存管理](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/MemoryManagement.html)。

Addressables provides several options and APIs for loading and instantiating Addressable assets. See [Loading Addressable assets](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadingAddressableAssets.html) for information and examples, including:  
可寻址对象提供了多个选项和 API，用于加载和实例化可寻址资产。有关信息和示例，请参阅[加载可寻址资源](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadingAddressableAssets.html)，包括：

* [Loading an single asset](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadingAddressableAssets.html#loading-a-single-asset)
* [Loading multiple assets](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadingAddressableAssets.html#loading-multiple-assets)
* [Loading an AssetReference](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadingAddressableAssets.html#loading-an-assetreference)
* [Loading Scenes](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadingAddressableAssets.html#loading-scenes)
* [Loading assets by location](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadingAddressableAssets.html#loading-assets-by-location)
* [Instantiating objects from Addressables](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadingAddressableAssets.html#instantiating-objects-from-addressables)
* [Releasing Addressable assets](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadingAddressableAssets.html#releasing-addressable-assets)
* [Using Addressables in a Scene](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadingAddressableAssets.html#using-addressables-in-a-scene)
* [Downloading dependencies in advance](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/DownloadDependenciesAsync.html)

Addressables uses asynchronous operations for most loading tasks. See [Operations](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsAsyncOperationHandle.html) for information on how to handle operations in your code, including:  
可寻址对象对大多数加载任务使用异步操作。有关如何处理代码中的操作的信息，请参阅[Operations](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsAsyncOperationHandle.html)，包括：

* [Releasing AsyncOperationHandle instances](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsAsyncOperationHandle.html#releasing-asyncoperationhandle-instances)
* [Coroutine- and IEnumerator-based operation handling](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsAsyncOperationHandle.html#coroutine-operation-handling)
* [Event-based operation handling](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsAsyncOperationHandle.html#event-based-operation-handling)
* [Task-based operation handling](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsAsyncOperationHandle.html#task-based-operation-handling)
* [Using operations synchronously](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsAsyncOperationHandle.html#using-operations-synchronously)
* [Custom operations](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsAsyncOperationHandle.html#custom-operations)
* [Using typed versus untyped operation handles](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsAsyncOperationHandle.html#using-typed-versus-typeless-operation-handles)
* [Reporting operation progress](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsAsyncOperationHandle.html#reporting-operation-progress)

See the following for information about other runtime topics:  
有关其他运行时主题的信息，请参阅以下内容：

* [Customizing initialization](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/InitializeAsync.html)
* [Loading additional catalogs](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadContentCatalogAsync.html#loading-additional-catalogs)
* [Updating catalogs](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadContentCatalogAsync.html#updating-catalogs)
* [Modifying resource URLs at runtime](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/TransformInternalId.html)
* [Getting the address of an asset at runtime](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/GetRuntimeAddress.html)

## Memory management 内存管理

The Addressables system manages the memory used to load assets and bundles by keeping a reference count of every item it loads.  
可寻址系统通过保留其加载的每个项目的引用计数来管理用于加载资产和捆绑包的内存。

When an Addressable is loaded, the system increments the reference count; when the asset is released, the system decrements the reference count. When the reference count of an Addressable returns to zero, it is eligible to be unloaded. When you explicitly load an Addressable asset, you must also release the asset when you are done with it.  
加载可寻址对象时，系统会递增引用计数;释放资产时，系统会递减引用计数。当可寻址对象的引用计数返回零时，它就可以被卸载。显式加载可寻址资产时，还必须在使用完资产后释放该资产。

The basic rule of thumb to avoid "memory leaks" (assets that remain in memory after they are no longer needed) is to mirror every call to a load function with a call to a release function. You can release an asset with a reference to the asset instance itself or with the result handle returned by the original load operation.  
避免“内存泄漏”（不再需要后保留在内存中的资产）的基本经验法则是成对使用release函数 和 load 函数。您可以使用对资产实例本身的引用或使用原始加载操作返回的结果句柄来释放资产。

Note, however, that released assets are not necessarily unloaded from memory immediately. The memory used by an asset is not freed until the AssetBundle it belongs to is also unloaded. (Released assets can also be unloaded by calling [Resources.UnloadUnusedAssets](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Resources.UnloadUnusedAssets.html), but that tends to be a slow operation which can cause frame rate hitches.)  
但请注意，释放的资产不一定会立即从内存中卸载。在资产所属的资产包也被卸载之前，不会释放资产使用的内存。（也可以通过调用 [Resources.UnloadUnusedAssets](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Resources.UnloadUnusedAssets.html) 来卸载已加载的资产，但这往往是一个缓慢的操作，可能会导致帧速率卡顿。

AssetBundles have their own reference count (the system treats them like Addressables with the assets they contain as dependencies). When you load an asset from a bundle, the bundle's reference count increases and when you release the asset, the bundle reference count decreases. When a bundle's reference count returns to zero, that means none of the assets it contains are still in use and the bundle and all the assets it contains are unloaded from memory.  
资产包有自己的引用计数（系统将它视为可寻址对象，并将它包含的资产视为它的依赖项）。从捆绑包加载资源时，捆绑包的引用计数会增加，而释放资源时，捆绑包引用计数会减少。当捆绑包的引用计数恢复为零时，这意味着它包含的任何资产都没有仍在使用中，并且捆绑包及其包含的所有资产都已从内存中卸载。

Use the [Event Viewer](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/EventViewer.html) to monitor your runtime memory management. The viewer shows when assets and their dependencies are loaded and unloaded.  
使用 [Event Viewer](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/EventViewer.html) 监视运行时内存管理。查看器显示何时加载和卸载资产及其依赖项。

### Understanding when memory is cleared 了解何时清除内存

An asset no longer being referenced (indicated by the end of a blue section in the [Event Viewer](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/EventViewer.html)) does not necessarily mean that asset was unloaded. A common applicable scenario involves multiple assets in an AssetBundle. For example:  
不再引用的资产（由 [Event Viewer](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/EventViewer.html)中的Resources蓝色条指示）并不一定意味着该资产已卸载。常见的适用场景涉及资产包中的多个资产。例如：

* You have three Assets (tree, tank, and cow) in an AssetBundle (stuff).  
  您在资产捆绑包（stuff）中有三个资产（tree、tank和cow）。
* When tree loads, the profiler displays a single ref-count for tree, and one for stuff.  
  当tree 加载时，探查器显示tree的单个引用计数，以及一个stuff的引用计数。
* Later, when tank loads, the profiler displays a single ref-count for both tree and tank, and two ref-counts for the stuff AssetBundle.  
  稍后，当 tank 加载时，分析器会显示 tree 和 tank 的单个引用计数，以及 stuff  AssetBundle 的两个引用计数。
* If you release tree, it's ref-count becomes zero, and the blue bar goes away.  
  如果你释放 tree，它的引用计数变为零，蓝色条消失。

In this example, the tree asset is not actually unloaded at this point. You can load an AssetBundle, or its partial contents, but you cannot partially unload an AssetBundle. No asset in stuff unloads until the AssetBundle itself is completely unloaded. The exception to this rule is the engine interface [Resources.UnloadUnusedAssets](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Resources.UnloadUnusedAssets.html). Executing this method in the above scenario causes tree to unload. Because the Addressables system cannot be aware of these events, the profiler graph only reflects the Addressables ref-counts (not exactly what memory holds). Note that if you choose to use [Resources.UnloadUnusedAssets](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Resources.UnloadUnusedAssets.html), it is a very slow operation, and should only be called on a screen that won't show any hitches (such as a loading screen).  
在此示例中，此时实际上并未卸载 tree 资产。您可以加载资源包或其部分内容，但不能部分卸载资产包。在资产包本身完全卸载之前，不会卸载任何资源。此规则的例外是引擎接口 [Resources.UnloadUnusedAssets](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Resources.UnloadUnusedAssets.html)。在上述方案中执行此方法会导致 tree 卸载。由于可寻址设备系统无法识别这些事件，因此探查器图仅反映可寻址对象引用计数（不完全反映内存所包含的内容）。请注意，如果您选择使用[Resources.UnloadUnusedAssets](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Resources.UnloadUnusedAssets.html)，则这是一个非常缓慢的操作，并且应仅在不会显示任何故障的屏幕上调用（例如加载屏幕）。

### Avoiding asset churn 避免资产流失

Asset churn is a problem that can arise if you release an object that happens to be the last item in an AssetBundle and then immediately reload either that asset or another asset in the bundle.  
如果您释放的对象恰好是 AssetBundle 中的最后一项，然后立即重新加载该资产或捆绑包中的其他资产，则可能会出现资产流失问题。

For example, say you have two materials, boat and plane that share a texture, cammo, which has been pulled into its own AssetBundle. Level 1 uses boat and level 2 uses plane. As you exit level 1 you release boat, and immediately load plane. When you release boat, Addressables unloads texture cammo. Then, when you load plane, Addressables immediately reloads cammo.  
例如，假设您有两种材质（boat 和plane ）共享一个纹理 cammo，它已被拉入自己的资产包中。第1关使用boat ，第2关使用plane。当你离开第1关时，你释放boat，并立即加载plane。当你释放boat时，可寻址对象会卸载纹理cammo。然后，当您加载plane时，可寻址对象会立即重新加载cammo。

You can use the [Event Viewer](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/EventViewer.html) to help detect asset churn by monitoring asset loading and unloading.  
您可以使用 [Event Viewer](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/EventViewer.html) 通过监视资产加载和卸载来帮助检测资产变动。

### AssetBundle memory overhead 资产捆绑包内存开销

When you load an AssetBundle, Unity allocates memory to store the bundle's internal data, which is in addition to the memory used for the assets it contains. The main types of internal data for a loaded AssetBundle include:  
加载资源包时，Unity 会分配内存来存储捆绑包的内部数据，这是用于其包含的资源的内存之外的内存。加载的资产包的主要内部数据类型包括：

* Loading cache: Stores recently accessed pages of an AssetBundle file. Use [AssetBundle.memoryBudgetKB](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/AssetBundle-memoryBudgetKB.html) to control its size.  
  加载缓存：存储最近访问的资产包文件的页面。使用[AssetBundle.memoryBudgetKB](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/AssetBundle-memoryBudgetKB.html)  来控制其大小。
* [TypeTrees](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/MemoryManagement.html#typetrees): Defines the serialized layout of your objects.  
  [类型树](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/MemoryManagement.html#typetrees)：定义对象的序列化布局。
* [Table of contents](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/MemoryManagement.html#table-of-contents): Lists the assets in a bundle.  
  [目录](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/MemoryManagement.html#table-of-contents)：列出捆绑包中的资产。
* [Preload table](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/MemoryManagement.html#preload-table): Lists the dependencies of each asset.  
  [预加载表](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/MemoryManagement.html#preload-table)：列出每个资产的依赖项。

When you organize your Addressable groups and AssetBundles, you typically must make trade-offs between the size and the number of AssetBundles you create and load. On the one hand, fewer, larger bundles can minimize the total memory usage of your AssetBundles. On the other hand, using a larger number of small bundles can minimize the peak memory usage because you can unload assets and AssetBundles more easily.  
在组织可寻址组和资产包时，通常必须在创建和加载的资产包的大小和数量之间进行权衡。一方面，更少、更大的捆绑包可以最大限度地减少资产包的总内存使用量。另一方面，使用大量小捆绑包可以最大限度地减少峰值内存使用量，因为您可以更轻松地卸载资产和资产包。

While the size of an AssetBundle on disk is not the same as its size at runtime, you can use the disk size as an approximate guide to the memory overhead of the AssetBundles in a build. You can get bundle size and other information you can use to help analyze your AssetBundles from the [Build Layout Report](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildLayoutReport.html).  
虽然磁盘上资产包的大小与其运行时的大小不同，但您可以使用磁盘大小作为构建中资产包内存开销的近似指南。您可以从[Build Layout Report](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildLayoutReport.html) 获取捆绑包大小和其他可用于帮助分析资产包的信息。

The following sections discuss the internal data used by AssetBundles and how you can minimize the amount of memory they require, where possible.  
以下各节讨论 AssetBundles 使用的内部数据，以及如何在可能的情况下最大程度地减少它们所需的内存量。

#### TypeTrees 类型树

A TypeTree describes the field layout of one of your data types.  
类型树描述一种数据类型的字段布局。

Each serialized file in an AssetBundle contains a TypeTree for each object type within the file. The TypeTree information allows you to load objects that are deserialized slightly differently from the way they were serialized. TypeTree information is not shared between AssetBundles; each bundle has a complete set of TypeTrees for the objects it contains.  
资产包中的每个序列化文件都包含文件中每个对象类型的类型树。类型树信息允许您在加载对象时进行反序列化（某些对象的反序列化和序列化之间可能会略有不同）。类型树信息不会在资产包之间共享;每个捆绑包都有一组完整的类型树，用于它所包含的对象。

All the TypeTrees are loaded when the AssetBundle is loaded and held in memory for the lifetime of the AssetBundle. The memory overhead associated with TypeTrees is proportional to the number of unique types in the serialized file and the complexity of those types.  
加载资产包时，将加载所有类型树，并在资产包的生存期内保留在内存中。与类型树关联的内存开销与序列化文件中各个类型的数量以及这些类型的复杂性成正比。

You can reduce the memory requirements of AssetBundle TypeTrees in the following ways:  
您可以通过以下方式降低资产捆绑类型树的内存需求：

* Keep assets of the same types together in the same bundles.  
  将相同类型的资源放在相同的捆绑包中。
* Turn off TypeTrees -- turning off TypeTrees makes your AssetBundles smaller by excluding this information from a bundle. However, without TypeTree information, you may encounter serialization errors or undefined behavior when loading older bundles with a newer version of Unity or after making even small script changes in your project.  
  关闭类型树 - 关闭类型树通过从捆绑包中排除此信息来缩小资产包。但是，如果没有类型树信息，在使用较新版本的 Unity 加载较旧的捆绑包时，或者在项目中进行即使是很小的脚本更改后，也可能会遇到序列化错误或未定义的行为。
* Prefer simpler data types to reduce TypeTree complexity.  
  首选更简单的数据类型以降低类型树的复杂性。

You can test the impact that TypeTrees have on the size of your AssetBundles by building them with and without TypeTrees disabled and comparing the sizes. Use [BuildAssetBundleOptions.DisableWriteTypeTree](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/BuildAssetBundleOptions.DisableWriteTypeTree.html) to disable TypeTrees in your AssetBundles. Note that not all platforms support TypeTrees and some platforms require TypeTrees (and ignore this setting).  
您可以通过在禁用和未禁用类型树的情况下构建类型树并比较大小来测试类型树对资产包大小的影响。使用 [BuildAssetBundleOptions.DisableWriteTypeTree](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/BuildAssetBundleOptions.DisableWriteTypeTree.html) 禁用AssetBundles中的类型树。请注意，并非所有平台都支持类型树，而另一些平台则可能必须要类型树（并忽略此设置）。

If you disable TypeTrees in a project, always rebuild local Addressable groups before building a new player. If you are distributing content remotely, only update content using the same version (including patch number) of Unity that you used to produce your current player and don't make even minor code changes. (When you are juggling multiple player versions, updates, and versions of Unity, you might not find the memory savings from disabling TypeTrees to be worth the trouble.)  
如果在项目中禁用类型树，请始终在生成新播放器之前重新生成local Addressable groups。如果要远程分发内容，请仅使用用于生成当前播放器的相同 Unity 版本（包括补丁号）更新内容，并且不要进行哪怕是微小的代码更改。（当您处理多个播放器版本、更新或 Unity 版本时，您可能发现禁用类型树所节省的内存不值得它所带来的麻烦）。

#### Table of contents 内容列表

The table of contents is a map within the bundle that allows you to look up each explicitly included asset by name. It scales linearly with the number of assets and the length of the string names by which they are mapped.  
内容列表是捆绑包中的地图，允许您按名称查找每个明确包含的资产。它与资产的数量和映射它们的字符串名称的长度呈线性比例。

The size of your table of contents data is based on the total number of assets. You can minimize the amount of memory dedicated to holding table of content data by minimizing the number of AssetBundles loaded at a given time.  
目录数据的大小基于资产总数。您可以通过最小化在给定时间加载的资产包数量来最小化专用于保存内容表数据的内存量。

#### Preload table 预加载表

The preload table is a list of all the other objects that an asset references. Unity uses the preload table to load these referenced objects when you load an asset from the AssetBundle.  
预加载表是资产引用的所有其他对象的列表。当您从资源包加载资源时，Unity 使用预加载表来加载这些引用的对象。

For example, a Prefab has a preload entry for each of its components as well as any other assets it may reference (materials, textures, etc). Each preload entry is 64 bits and can reference objects in other AssetBundles.  
例如，预制件的每个组件以及它可能引用的任何其他资源（材质、纹理等）都有一个预加载条目。每个预加载条目为 64 位，可以引用其他资产包中的对象。

When an asset references another asset that in turn references other assets, the preload table can become large because it contains the entries needed to load both assets. If two assets both reference a third asset, then the preload tables of both contain entries to load the third asset (whether or not the referenced asset is Addressable or in the same AssetBundle).  
当一个资产引用另一个资产而另一个资产又引用其他资产时，预加载表可能会变得很大，因为它包含加载两个资产所需的条目。如果两个资产都引用第三个资产，则两个资产的预加载表都包含用于加载第三个资产的条目（无论引用的资产是可寻址的还是在同一 AssetBundle 中）。

As an example, consider a situation in which you have two assets in an AssetBundle (PrefabA and PrefabB) and both of these prefabs reference a third prefab (PrefabC), which is large and contains several components and references to other assets. This AssetBundle contains two preload tables, one for PrefabA and one for PrefabB. Those tables contain entries for all the objects of their respective prefab, but also entries for all the objects in PrefabC and any objects referenced by PrefabC. Thus, the information required to load PrefabC ends up duplicated in both PrefabA and PrefabB. This happens whether or not PrefabC is explicitly added to an AssetBundle.  
例如，假设您在 AssetBundle 中有两个资产（PrefabA和PrefabB），并且这两个预制件都引用第三个预制件 （PrefabC），该预制件（PrefabC）很大，包含多个组件和对其他资源的引用。此资源包包含两个预加载表，一个用于PrefabA，一个用于PrefabB。这些表包含其各自预制件的所有对象的条目，还包含PrefabC 中所有对象和PrefabC引用的任何其他对象的条目。因此，加载PrefabC所需的信息最终会在PrefabA和PrefabB中重复。无论PrefabC是否显式添加到资产包中，都会发生这种情况。

Depending on how you organize your assets, the preload tables in AssetBundles could become quite large and contain many duplicate entries. This is especially true if you have several loadable assets that all reference a complex asset, such as PrefabC in the situation above. If you determine that the memory overhead from the preload table is a problem, you can structure your loadable assets so that they have fewer complex loading dependencies.  
根据您组织资产的方式，AssetBundles 中的预加载表可能会变得非常大，并且包含许多重复条目。如果您有多个可加载资产，这些资产都引用了一个复杂的资源，例如上述情况下的 PrefabC，则尤其如此。如果您确定预加载表中的内存开销是一个问题，则应该合理构建可加载资产，以便它们具有更少的复杂加载依赖项。

### AssetBundle dependencies 资产包依赖项

Loading an Addressable asset also loads all of the AssetBundles containing its dependencies. An AssetBundle dependency occurs when an asset in one bundle references an asset in another bundle. An example of this is a material referencing a texture.  
加载可寻址资产还会加载包含其依赖项的所有资产包。当一个捆绑包中的资产引用另一个捆绑包中的资产时，就会发生资产包依赖关系。这方面的一个例子是引用纹理的材质。

Addressables calculates dependencies between bundles at the bundle level. If one asset references an object in another bundle, then the entire bundle has a dependency on that bundle. This means that even if you load an asset in the first bundle that has no dependencies of its own, the second AssetBundle is still loaded into memory.  
可寻址对象在捆绑包级别计算捆绑包之间的依赖关系。如果一个资源引用另一个捆绑包中的对象，则整个捆绑包都依赖于该捆绑包。这意味着，即使您在第一个的捆绑包中加载没有自身依赖项的资产，第二个捆绑包仍会加载到内存中。

To avoid loading more bundles than are required, you should strive to keep the dependencies between AssetBundles as simple as possible.  
为避免加载的捆绑包过多，应尽量保持资产包之间的合理依赖关系。

##### NOTE

Prior to Addressables 1.13.0, the dependency graph was not as thorough as it is now. In the example above, RootAsset1 would not have had a dependency on BundleB. This previous behavior resulted in references breaking when an AssetBundle being referenced by another AssetBundle was unloaded and reloaded. This fix may result in additional data remaining in memory if the dependency graph is complex enough.  
在Addressables 1.13.0 之前，依赖关系图并不像现在这样彻底。在上面的例子中，RootAsset1不会依赖于BundleB。以前的行为会导致在卸载并重新加载另一个资产包引用的资产包时引用中断。如果依赖项关系图足够复杂，此修复可能会导致内存中保留其他数据。

## Loading Addressable assets 加载可寻址资产

The [Addressables](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.html) class provides several methods for loading Addressable assets. You can load assets one at a time or in batches. To identify the assets to load, you pass either a single key or a list of keys to the loading function. A key can be one of the following objects:  
[Addressables](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.html) 类提供了几种用于加载可寻址资产的方法。您可以一次加载一个资产，也可以批量加载资产。要标识要加载的资产，请将单个键或键列表传递给加载函数。键可以是以下对象之一：

* Address: a string containing the address you assigned to the asset  
  地址：包含您分配给资产的地址的字符串
* Label: a string containing a label assigned to one or more assets  
  标签：包含分配给一个或多个资产的标签的字符串
* AssetReference object: an instance of [AssetReference](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.AssetReference.html)  
  资产引用对象：[AssetReference](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.AssetReference.html)的实例
* [IResourceLocation](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceLocations.IResourceLocation.html) instance: an intermediate object that contains information to load an asset and its dependencies.  
  [IResourceLocation](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceLocations.IResourceLocation.html)  实例：一个中间对象，其中包含用于加载资产及其依赖项的信息。

When you call one of the asset loading functions, the Addressables system begins an asynchronous operation that carries out the following tasks:  
调用其中一个资产加载函数时，可寻址对象系统将开始执行以下任务的异步操作：

1. Looks up the resource locations for the specified keys (except IResourceLocation keys)  
   查找指定键的资源位置（IResourceLocation 键除外）
2. Gathers the list of dependencies  
   收集依赖项列表
3. Downloads any remote AssetBundles that are required  
   下载所需的任何远程资源包
4. Loads the AssetBundles into memory  
   将资源包加载到内存中
5. Sets the [Result](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.Result.html#UnityEngine_ResourceManagement_AsyncOperations_AsyncOperationHandle_Result) object of the operation to the loaded objects  
   将加载好的对象赋值给句柄的[Result](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.Result.html#UnityEngine_ResourceManagement_AsyncOperations_AsyncOperationHandle_Result) 属性
6. Updates the [Status](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.Status.html#UnityEngine_ResourceManagement_AsyncOperations_AsyncOperationHandle_Status) of the operation and calls any [Completed](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.Completed.html) event listeners  
   更新句柄的[Status](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.Status.html#UnityEngine_ResourceManagement_AsyncOperations_AsyncOperationHandle_Status) 并调用句柄的[Completed](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.Completed.html) 事件侦听器

If the load operation succeeds, the Status is set to Succeeded and the loaded object or objects can be accessed from the [Result](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.Result.html#UnityEngine_ResourceManagement_AsyncOperations_AsyncOperationHandle_Result) object.  
如果加载操作成功，则[Status](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.Status.html#UnityEngine_ResourceManagement_AsyncOperations_AsyncOperationHandle_Status) 设置为“成功”，并且可以从[Result](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.Result.html#UnityEngine_ResourceManagement_AsyncOperations_AsyncOperationHandle_Result) 属性访问加载的一个或多个对象。

If an error occurs, the exception is copied to the [OperationException](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.OperationException.html#UnityEngine_ResourceManagement_AsyncOperations_AsyncOperationHandle_OperationException) member of the operation object and the Status is set to Failed. By default, the exception is not thrown as part of the operation. However, you can assign a handler function to the [ResourceManager.ExceptionHandler](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceManager.ExceptionHandler.html#UnityEngine_ResourceManagement_ResourceManager_ExceptionHandler) property to handle any exceptions. Additionally, you can enable the [Log Runtime Exceptions](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html#diagnostics) option in your Addressable system settings to record errors to the Unity [Console](https://docs.unity3d.com/2019.4/Documentation/Manual/Console.html).  
如果发生错误，则将异常复制到句柄的[OperationException](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.OperationException.html#UnityEngine_ResourceManagement_AsyncOperations_AsyncOperationHandle_OperationException) 成员，并将[Status](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.Status.html#UnityEngine_ResourceManagement_AsyncOperations_AsyncOperationHandle_Status) 设置为Failed。默认情况下，异常不会作为操作的一部分被抛出。但是，可以将处理程序函数分配给[ResourceManager.ExceptionHandler](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceManager.ExceptionHandler.html#UnityEngine_ResourceManagement_ResourceManager_ExceptionHandler) 属性以处理任何异常。此外，还可以在可寻址系统设置中启用[Log Runtime Exceptions](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html#diagnostics) 选项，以将错误记录到 Unity  [Console](https://docs.unity3d.com/2019.4/Documentation/Manual/Console.html)。

When you call loading functions that can load multiple Addressable assets, you can specify whether the entire operation should abort if any single load operation fails or whether the operation should load any assets it can. In both cases, the operation status is set to failed. (Set the releaseDependenciesOnFailure parameter to true in the call to the loading function to abort the entire operation on any failure.)  
调用可加载多个可寻址资产的函数时，你可以决定是否整个加载过程在遇到一个资源失败时就终止还是尽可能加载全部资源。在这两种情况下，[Status](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.Status.html#UnityEngine_ResourceManagement_AsyncOperations_AsyncOperationHandle_Status) 都设置为Failed。（在函数调用时将 releaseDependenciesOnFailure 参数设置为 true，以便在任何失败时中止整个操作。）

See [Operations](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsAsyncOperationHandle.html) for more information about asynchronous operations and writing asynchronous code in Unity scripts.  
有关异步操作和在 Unity 脚本中编写异步代码的详细信息，请参阅[Operations](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsAsyncOperationHandle.html) 。

### Loading a single asset 加载单个资产

Use the [LoadAssetAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetAsync.html) method to load a single Addressable asset, typically with an address as the key:  
使用  [LoadAssetAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetAsync.html) **方法加载单个可寻**址**资产，通常使用地址作为键：**

using System.Collections;

using UnityEngine;

using UnityEngine.AddressableAssets;

using UnityEngine.ResourceManagement.AsyncOperations;

internal class LoadAddress : MonoBehaviour

{

public string key;

AsyncOperationHandle<GameObject> opHandle;

public IEnumerator Start() {

opHandle = Addressables.LoadAssetAsync<GameObject>(key);

yield return opHandle;

if (opHandle.Status == AsyncOperationStatus.Succeeded) {

GameObject obj = opHandle.Result;

Instantiate(obj, transform);

}

}

void OnDestroy() {

Addressables.Release(opHandle);

}

}

##### NOTE

You can use a label or other type of key when you call [LoadAssetAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetAsync.html), not just an address. However, if the key resolves to more than one asset, only the first asset found is loaded. For example, if you call this method with a label applied to several assets, Addressables returns whichever one of those assets that happens to be located first.  
调用[LoadAssetAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetAsync.html)时，可以使用label或其他类型的key，而不仅仅是地址。但是，如果key对应多个资产，则仅加载找到的第一个资产。例如，label 对应多个资产，则 Addressables 将返回随机找到的第一个资产。

### Loading multiple assets 加载多个资源

Use the [LoadAssetsAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetsAsync.html) method to load more than one Addressable asset in a single operation. When using this function, you can specify a single key, such as a label, or a list of keys.  
使用 [[LoadAssetsAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetsAsync.html)](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetsAsync.html) 方法在单个操作中加载多个可寻址资产。使用此函数时，可以指定单个key（例如label）或list of keys。

When you specify multiple keys, you can specify a [merge mode](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.MergeMode.html) to determine how the sets of assets matching each key are combined:  
指定多个键时，可以指定[合并模式](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.MergeMode.html)来确定如何组合与每个key匹配的资产集：

* **Union**: include assets that match any key  
  **Union**：包括与任何键匹配的资产
* **Intersection**: include assets that match every key  
  **Intersection**：只包含同时满足所有keys的资源
* **UseFirst**: include assets only from the first key that resolves to a valid location
* **UseFirst**: 仅包含第一个有效的key所对应的资产（注意：一个key也可能对应着多个资产）

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.AddressableAssets;

using UnityEngine.ResourceManagement.AsyncOperations;

internal class LoadMultiple : MonoBehaviour

{

// Label strings to load

public List<string> keys = new List<string>() { "characters", "animals" };

// Operation handle used to load and release assets

AsyncOperationHandle<IList<GameObject>> loadHandle;

// Load Addressables by Label

public IEnumerator Start() {

float x = 0, z = 0;

loadHandle = Addressables.LoadAssetsAsync<GameObject>(

keys,

addressable => {

//Gets called for every loaded asset

Instantiate<GameObject>(addressable,

new Vector3(x++ \* 2.0f, 0, z \* 2.0f),

Quaternion.identity,

transform);

if (x > 9) {

x = 0;

z++;

}

}, Addressables.MergeMode.Union, // How to combine multiple labels

false); // Whether to fail and release if any asset fails to load

yield return loadHandle;

}

private void OnDestroy() {

Addressables.Release(loadHandle);

// Release all the loaded assets associated with loadHandle

// Note that if you do not make loaded addressables a child of this object,

// then you will need to devise another way of releasing the handle when

// all the individual addressables are destroyed.

}

}

You can specify how to handle loading errors with the releaseDependenciesOnFailure parameter. If true, then the operation fails if it encounters an error loading any single asset. The operation and any assets that did successfully load are released.  
您可以使用releaseDependenciesOnFailure 参数指定如何处理加载错误。如果为 true，则操作在加载任何单个资产时遇到错误时将失败。将释放操作和成功加载的任何资产。

If false, then the operation loads any objects that it can and does not release the operation. In the case of failures, the operation still completes with a status of Failed. In addition, the list of assets returned has null values where the failed assets would otherwise appear.  
如果为 true，则操作将加载它可以加载的任何对象，并且也不会释放操作本身。如果失败，操作仍以“失败”状态完成。此外，返回的资产列表具有空值用于显示加载失败的资产。

Set releaseDependenciesOnFailure to true when loading a group of assets that must be loaded as a set in order to be used. For example, if you are loading the assets for a game level, it might make sense to fail the operation as a whole rather than load only some of the required assets.  
在加载必须作为集加载才能使用的资产组时，应该将releaseDependenciesOnFailure 设置为 true。例如，如果要加载游戏关卡的资源，使整个操作失败而不是仅加载部分必需资产可能是有意义的。

#### Correlating loaded assets to their keys 将加载的资产与其key相关联

When you load multiple assets in one operation, the order in which individual assets are loaded is not necessarily the same as the order of the keys in the list you pass to the loading function.  
在一个操作中加载多个资产时，单个资产的加载顺序不一定与传递给 load 函数的列表中key的顺序相同。

If you need to associate an asset in a combined operation with the key used to load it, you can perform the operation in two steps:  
如果需要将操作中的资产与加载该资产的key相关联，可以通过两个步骤执行该操作：

1. Load the [IResourceLocation](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceLocations.IResourceLocation.html) instances with the list of asset keys.  
   使用资产键列表加载[IResourceLocation](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceLocations.IResourceLocation.html) 实例。
2. Load the individual assets using their IResourceLocation instances as keys.  
   使用其 IResourceLocation 实例作为键加载资产。

The IResourceLocation object contains the key information so you can, for example, keep a dictionary to correlate the key to an asset. Note that when you call a loading function, such as [LoadAssetsAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetsAsync.html), the operation first looks up the [IResourceLocation](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceLocations.IResourceLocation.html) instances that correspond to a key and then uses that to load the asset. When you load an asset using an IResourceLocation, the operation skips the first step. Thus, performing the operation in two steps does not add significant additional work.  
IResourceLocation 对象包含key信息，因此您可以保留字典以将键与资产相关联。请注意，当您调用加载函数（如[[LoadAssetsAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetsAsync.html)](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetsAsync.html)）时，该操作首先查找与键对应的 [IResourceLocation](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceLocations.IResourceLocation.html)  实例，然后使用该实例加载资产。使用 IResourceLocation 加载资产时，该操作将跳过第一步。因此，分两个步骤执行操作不会增加大量额外工作。

The following example loads the assets for a list of keys and inserts them into a dictionary by their address ([PrimaryKey](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceLocations.IResourceLocation.PrimaryKey.html#UnityEngine_ResourceManagement_ResourceLocations_IResourceLocation_PrimaryKey)). The example first loads the resource locations for the specified keys. When that operation is complete, it loads the asset for each location, using the Completed event to insert the individual operation handles into the dictionary. The operation handles can be used to instantiate the assets, and, when the assets are no longer needed, to release them.  
以下示例加载list of keys的资产，并按其地址 （[PrimaryKey](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceLocations.IResourceLocation.PrimaryKey.html#UnityEngine_ResourceManagement_ResourceLocations_IResourceLocation_PrimaryKey)） 将它们插入字典中。该示例首先加载指定键的资源位置。该操作完成后，它将加载每个位置的资产，使用 Done 事件将各个操作句柄插入字典中。操作句柄可用于实例化资产，并在不再需要资产时释放资产。

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.AddressableAssets;

using UnityEngine.Events;

using UnityEngine.ResourceManagement.AsyncOperations;

using UnityEngine.ResourceManagement.ResourceLocations;

internal class LoadWithLocation : MonoBehaviour

{

public Dictionary<string, AsyncOperationHandle<GameObject>> operationDictionary;

public List<string> keys;

public UnityEvent Ready;

IEnumerator LoadAndAssociateResultWithKey(IList<string> keys) {

if (operationDictionary == null)

operationDictionary = new Dictionary<string, AsyncOperationHandle<GameObject>>();

AsyncOperationHandle<IList<IResourceLocation>> locations

= Addressables.LoadResourceLocationsAsync(keys,

Addressables.MergeMode.Union, typeof(GameObject));

yield return locations;

var loadOps = new List<AsyncOperationHandle>(locations.Result.Count);

foreach (IResourceLocation location in locations.Result) {

AsyncOperationHandle<GameObject> handle =

Addressables.LoadAssetAsync<GameObject>(location);

handle.Completed += obj => operationDictionary.Add(location.PrimaryKey, obj);

loadOps.Add(handle);

}

yield return Addressables.ResourceManager.CreateGenericGroupOperation(loadOps, true);

Ready.Invoke();

}

void Start() {

Ready.AddListener(OnAssetsReady);

StartCoroutine(LoadAndAssociateResultWithKey(keys));

}

private void OnAssetsReady() {

float x = 0, z = 0;

foreach (var item in operationDictionary) {

Debug.Log($"{item.Key} = {item.Value.Result.name}");

Instantiate(item.Value.Result,

new Vector3(x++ \* 2.0f, 0, z \* 2.0f),

Quaternion.identity, transform);

if (x > 9) {

x = 0;

z++;

}

}

}

private void OnDestroy() {

foreach (var item in operationDictionary) {

Addressables.Release(item.Value);

}

}

}

Note that the loading function creates a group operation with [ResourceManager.CreateGenericGroupOperation](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceManager.CreateGenericGroupOperation.html). This allows the function to continue after all the loading operations have finished. In this case, the function dispatches a "Ready" event to notify other scripts that the loaded data can be used.  
请注意，load 函数使用 [ResourceManager.CreateGenericGroupOperation](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceManager.CreateGenericGroupOperation.html) 创建组操作。这允许函数在所有加载操作完成后继续。在这种情况下，该函数调度Ready事件以通知其他脚本可以使用加载的数据。

### Loading an AssetReference 加载AssetReference

The [AssetReference](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.AssetReference.html) class has its own load method, [LoadAssetAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetAsync.html).  
[AssetReference](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.AssetReference.html) 类有自己本身的加载方法[LoadAssetAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetAsync.html)。

using UnityEngine;

using UnityEngine.AddressableAssets;

using UnityEngine.ResourceManagement.AsyncOperations;

internal class LoadFromReference : MonoBehaviour

{

// Assign in Editor

public AssetReference reference;

// Start the load operation on start

void Start() {

AsyncOperationHandle handle = reference.LoadAssetAsync<GameObject>();

handle.Completed += Handle\_Completed;

}

// Instantiate the loaded prefab on complete

private void Handle\_Completed(AsyncOperationHandle obj) {

if (obj.Status == AsyncOperationStatus.Succeeded) {

Instantiate(reference.Asset, transform);

} else {

Debug.LogError("AssetReference failed to load.");

}

}

// Release asset when parent object is destroyed

private void OnDestroy() {

reference.ReleaseAsset();

}

}

You can also use the AssetReference object as a key to the [Addressables.LoadAssetAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetAsync.html) methods. If you need to spawn multiple instances of the asset assigned to an AssetReference, use [Addressables.LoadAssetAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetAsync.html), which gives you an operation handle that you can use to release each instance.  
还可以使用 AssetReference 对象作为[Addressables.LoadAssetAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetAsync.html) 方法的key。如果需要生成分配给 AssetReference 的资产的多个实例，请使用 [Addressables.LoadAssetAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetAsync.html)，它为您提供了一个可用于释放每个实例的操作句柄。

See [AssetReference](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.AssetReference.html) for more information about using AssetReferences.  
有关使用资产引用的详细信息，请参阅[AssetReference](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.AssetReference.html) 。

### Loading Scenes 加载场景

Use the [Addressables.LoadSceneAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadSceneAsync.html) method to load an Addressable Scene asset by address or other Addressable key object.  
使用[Addressables.LoadSceneAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadSceneAsync.html) 方法按地址或其他可寻址key对象加载可寻址场景资产。

The remaining parameters of the method correspond to those used with the Unity Engine [SceneManager.LoadSceneAsync](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/SceneManagement.SceneManager.LoadSceneAsync.html) method:  
该方法的其余参数对应于与Unity Engine [SceneManager.LoadSceneAsync](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/SceneManagement.SceneManager.LoadSceneAsync.html) 一起使用的参数。

* **loadMode**: whether to add the loaded Scene into the current Scene or to unload and replace the current Scene.  
  **loadMode**：是将加载的场景添加到当前场景中，还是卸载并替换当前场景。
* **activateOnLoad**: whether to activate the scene as soon as it finishes loading or to wait until you call the SceneInstance object's [ActivateAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceProviders.SceneInstance.ActivateAsync.html) method. Corresponds to the [AsyncOperation.allowSceneActivation](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/AsyncOperation-allowSceneActivation.html) option. Defaults to true.  
  **activateOnLoad**：是在场景完成加载后立即激活场景，还是等到调用 SceneInstance 对象的 [ActivateAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceProviders.SceneInstance.ActivateAsync.html)  方法。对应于[AsyncOperation.allowSceneActivation](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/AsyncOperation-allowSceneActivation.html) 选项。默认值为 true。
* **priority**: the priority of the AsyncOperation used to load the Scene. Corresponds to the [AsyncOperation.priority](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/AsyncOperation-priority.html) option. Defaults to 100.  
  **priority**：用于加载场景的异步操作的优先级。对应于[AsyncOperation.priority](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/AsyncOperation-priority.html) 选项。默认值为 100。

##### WARNING

Setting the activateOnLoad parameter to false blocks the AsyncOperation queue, including the loading of any other Addressable assets, until you activate the scene. To activate the scene, call the [ActivateAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceProviders.SceneInstance.ActivateAsync.html) method of the [SceneInstance](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceProviders.SceneInstance.html) returned by [LoadSceneAsync](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/SceneManagement.SceneManager.LoadSceneAsync.html). See [AsyncOperation.allowSceneActivation](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/AsyncOperation-allowSceneActivation.html) for additional information.  
将 activateOnLoad  参数设置为 false 会阻止 AsyncOperation 队列，包括加载任何其他可寻址资源，直到激活场景。若要激活场景，请调用 [LoadSceneAsync](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/SceneManagement.SceneManager.LoadSceneAsync.html) 返回的 [SceneInstance](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceProviders.SceneInstance.html) 实例的 [ActivateAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceProviders.SceneInstance.ActivateAsync.html) 方法。有关其他信息，请参阅 [AsyncOperation.allowSceneActivation](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/AsyncOperation-allowSceneActivation.html) 。

The following example loads a scene additively. The Component that loads the Scene, stores the operation handle and uses it to unload and release the Scene when the parent GameObject is destroyed.  
以下示例以加法方式加载场景。加载场景、存储操作句柄并在父游戏对象被销毁时使用它来卸载和释放场景的组件。

using UnityEngine;

using UnityEngine.AddressableAssets;

using UnityEngine.ResourceManagement.AsyncOperations;

using UnityEngine.ResourceManagement.ResourceProviders;

using UnityEngine.SceneManagement;

internal class LoadSceneByAddress : MonoBehaviour

{

public string key; // address string

private AsyncOperationHandle<SceneInstance> loadHandle;

void Start() {

loadHandle = Addressables.LoadSceneAsync(key, LoadSceneMode.Additive);

}

void OnDestroy() {

Addressables.UnloadSceneAsync(loadHandle);

}

}

See the [Scene loading project](https://github.com/Unity-Technologies/Addressables-Sample/tree/master/Basic/Scene%20Loading) in the [Addressables-Sample](https://github.com/Unity-Technologies/Addressables-Sample) repository for additional examples.  
有关其他示例，请参阅[Addressables-Sample](https://github.com/Unity-Technologies/Addressables-Sample)存储库中的[Scene loading project](https://github.com/Unity-Technologies/Addressables-Sample/tree/master/Basic/Scene%20Loading)。

If you load a Scene with [LoadSceneMode.Single](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/SceneManagement.LoadSceneMode.Single.html), the Unity runtime unloads the current Scene and calls [Resources.UnloadUnusedAssets](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Resources.UnloadUnusedAssets.html). The unloaded Scene is released, which allows its AssetBundle to be unloaded. Individual Addressables and their operation handles that you loaded separately are not released; you must release them yourself. (The exception to this is that any Addressable assets that you instantiated using [Addressables.InstantiateAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.InstantiateAsync.html) with trackHandle set to true, the default, are automatically released.)  
如果使用[LoadSceneMode.Single](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/SceneManagement.LoadSceneMode.Single.html) 加载场景，Unity 运行时将卸载当前场景并调用[Resources.UnloadUnusedAssets](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Resources.UnloadUnusedAssets.html)。释放已卸载的场景，从而允许卸载其资源包。但是引擎不会释放单独加载的单个可寻址对象及其操作句柄;你必须自己释放它们。（例外情况是，对于使用[Addressables.InstantiateAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.InstantiateAsync.html) 方法实例化的任何可寻址资产，如果 trackHandle 设置为 true（默认值），将自动释放。）

##### NOTE

In the Editor, you can always load scenes in the current project, even when they are packaged in a remote bundle that is not available and you set the Play Mode Script to **Use Existing Build**. The Editor loads the Scene using the asset database.  
在编辑器中，您始终可以在当前项目中加载场景，即使它们打包在不可用的远程捆绑包中，并且您将播放模式脚本设置为**Use Existing Build**。编辑器使用asset database加载场景。

### Loading assets by location 按位置加载资产

When you load an Addressable asset by address, label, or AssetReference, the Addressables system first looks up the resource locations for the assets and uses these [IResourceLocation](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceLocations.IResourceLocation.html) instances to download the required AssetBundles and any dependencies. You can perform the asset load operation in two steps by first getting the IResourceLocation objects with [LoadResourceLocationsAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadResourceLocationsAsync.html) and then using those objects as keys to load or instantiate the assets.  
按address、label或AssetReference加载可寻址资产时，可寻址对象系统首先查找资产的资源位置，并使用这些[IResourceLocation](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceLocations.IResourceLocation.html) 实例下载所需的资产包和任何依赖项。您可以通过两个步骤执行资产加载操作，方法是首先使用 [LoadResourceLocationsAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadResourceLocationsAsync.html) 获取 IResourceLocation 对象，然后将这些对象用作加载或实例化资产的键。

[IResourceLocation](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceLocations.IResourceLocation.html) objects contain the information needed to load one or more assets.  
[IResourceLocation](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceLocations.IResourceLocation.html) 对象包含加载一个或多个资产所需的信息。

The [LoadResourceLocationsAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadResourceLocationsAsync.html) method never fails. If it cannot resolve the specified keys to the locations of any assets, it returns an empty list. You can restrict the types of asset locations returned by the function by specifying a specific type in the type parameter.  
[LoadResourceLocationsAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadResourceLocationsAsync.html) 方法永远不会失败。如果它无法将指定的键解析为任何资产的位置，它将返回一个空列表。您可以通过在 type  参数中指定特定类型来限制函数返回的资产位置的类型

The following example loads locations for all assets labeled with "knight" or "villager":

AsyncOperationHandle<IList<IResourceLocation>> handle

= Addressables.LoadResourceLocationsAsync(

new string[]{"knight", "villager"},

Addressables.MergeMode.Union);

yield return handle;

//...

Addressables.Release(handle);

### Loading locations of subobjects 加载子对象的位置

Locations for SubObjects are generated at runtime to reduce the size of the content catalogs and improve runtime performance. When you call [LoadResourceLocationsAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadResourceLocationsAsync.html) with the key of an asset with subobjects and don't specify a type, then the function generates IResourceLocation instances for all of the subobjects as well as the main object (if applicable). Likewise, if you do not specify which subobject to use for an AssetReference that points to an asset with subobjects, then the system generates IResourceLocations for every subobject.  
子对象的位置在运行时生成，以减小内容目录的大小并提高运行时性能。当您使用具有子对象的资产的key调用[[LoadResourceLocationsAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadResourceLocationsAsync.html)](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadResourceLocationsAsync.html) 并且未指定类型时，该函数会为所有子对象以及主对象（如果适用）生成 IResourceLocation 实例。同样，如果使用AssetReference时，没有指定具体需要使用的的子对象，则系统会为每个子对象生成 IResourceLocation。

For example, if you load the locations for an FBX asset, with the address, "myFBXObject", you might get locations for three assets: a GameObject, a Mesh, and a Material. If, instead, you specified the type in the address, "myFBXObject[Mesh]", you would only get the Mesh object. You can also specify the type using the type parameter of the [LoadResourceLocationsAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadResourceLocationsAsync.html) function.  
例如，如果使用地址"myFBXObject"加载 FBX 资产的位置，则可能会获得三个资产的位置：GameObject、Mesh和Material。相反，如果您在地址中指定了类型"myFBXObject[Mesh]"，则只会获得Mesh对象。您还可以使用 [[LoadResourceLocationsAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadResourceLocationsAsync.html)](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadResourceLocationsAsync.html)函数的type 参数指定类型。

### Instantiating objects from Addressables 从可寻址对象实例化对象

You can load an asset, such as a Prefab, and then create an instance of it with [Instantiate](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Object.Instantiate.html). You can also load and create an instance of an asset with [Addressables.InstantiateAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.InstantiateAsync.html). The primary difference between these two ways of instantiating objects is how the asset reference counts are affected.  
您可以加载资产（如预制件），然后使用[Instantiate](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Object.Instantiate.html)创建该资产的实例。您还可以使用[Addressables.InstantiateAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.InstantiateAsync.html) 加载和创建资产的实例。这两种实例化对象的方式之间的主要区别在于资产引用计数如何受到影响。

When you use InstantiateAsync, the reference counts of the loaded assets are incremented each time you call the method. Thus if you instantiate a Prefab five times, the reference count for the Prefab asset and any of its dependencies are incremented by five. You can then release each instance separately as they are destroyed in the game.  
使用 InstantiateAsync 时，每次调用该方法时，加载资产的引用计数都会递增。因此，如果实例化预制件五次，则预制件资产及其任何依赖项的引用计数将递增五。然后，您可以在每个实例在游戏中被销毁时单独释放它们。

When you use LoadAssetAsync and Object.Instantiate, then the asset reference counts are only incremented once, for the initial load. If you release the loaded asset (or its operation handle) and the reference count drops to zero, then the asset is unloaded and all the additional instantiated copies lose their subassets as well -- they still exist as GameObjects in the scene, but without Meshes, Materials, or other assets that they depend on.  
当您使用 LoadAssetAsync 和 Object.Instantiate 时，对于初始加载，资产引用计数仅递增一次。如果释放加载的资源（或其操作句柄）并且引用计数降至零，则资源将被卸载，所有其他实例化副本也会丢失其子资源 - 它们仍作为游戏对象存在于场景中，但没有网格体、材质或它们所依赖的其他资产。

Which scenario is better, depends on how you organize your object code. For example, if you have a single manager object that supplies a pool of Prefab enemies to spawn into a game level, it might be most convenient to release them all at the completion of the level with a single operation handle stored in the manager class. In other situations, you might want to instantiate and release assets individually.  
哪种方案更好，取决于您如何组织目标代码。例如，如果您有一个管理器对象，该对象提供一组预制件敌人以生成到游戏关卡中，则最方便的做法可能是在关卡完成后使用管理器类中存储的单个操作句柄将它们全部释放。在其他情况下，您可能希望单独实例化和释放资产。

The following example calls [InstantiateAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.InstantiateAsync.html) to instantiate a Prefab. The example adds a component to the instantiated GameObject that releases the asset when the GameObject is destroyed.  
下面的示例调用 [InstantiateAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.InstantiateAsync.html) 来实例化预制件。该示例将一个组件添加到实例化的游戏对象，该组件在销毁游戏对象时释放资产。

using UnityEngine;

using UnityEngine.AddressableAssets;

using UnityEngine.ResourceManagement.AsyncOperations;

internal class InstantiateFromKey : MonoBehaviour

{

public string key; // Identify the asset

void Start() {

// Load and instantiate

Addressables.InstantiateAsync(key).Completed += instantiate\_Completed;

}

private void instantiate\_Completed(AsyncOperationHandle<GameObject> obj) {

// Add component to release asset in GameObject OnDestroy event

obj.Result.AddComponent(typeof(SelfCleanup));

}

}

// Releases asset (trackHandle must be true in InstantiateAsync,

// which is the default)

internal class SelfCleanup : MonoBehaviour

{

void OnDestroy() {

Addressables.ReleaseInstance(gameObject);

}

}

When you call [InstantiateAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.InstantiateAsync.html) you have the same options as the [Object.Instantiate](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Object.Instantiate.html) method, and also the following additional parameters:  
调用 [InstantiateAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.InstantiateAsync.html) 时，您具有与[Object.Instantiate](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Object.Instantiate.html) 方法相同的选项，以及以下附加参数：

* **instantiationParameters**: this parameter takes a [InstantiationParameters](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceProviders.InstantiationParameters.html) struct that you can use to specify the instantiation options instead of specifying them in every call to the InstantiateAsync call. This can be convenient if you use the same values for multiple instantiations.  
  **instantiationParameters**：此参数采用 [InstantiationParameters](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceProviders.InstantiationParameters.html)结构，可用于指定实例化选项，而不是在每次调用 InstantiateAsync 调用时指定它们。如果对多个实例化使用相同的值，这会很方便。
* **trackHandle**: If true, which is the default, then the Addressables system keeps track of the operation handle for the instantiated instance. This allows you to release the asset with the [Addressables.ReleaseInstance](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.ReleaseInstance.html) method. If false, then the operation handle is not tracked for you and you must store a reference to the handle returned by InstantiateAsync in order to release the instance when you destroy it.  
  **trackHandle**：如果为 true（默认值），则可寻址对象系统会跟踪实例化实例的操作句柄。这允许您使用[Addressables.ReleaseInstance](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.ReleaseInstance.html) 方法释放资产。如果为false，则不会为您跟踪操作句柄，您必须存储对 InstantiateAsync返回的句柄的引用，以便在销毁实例时释放实例。

### Asynchronous Loading 异步加载

The Addressables system API is asynchronous and returns an [AsyncOperationHandle](xref:UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle-1) for use with managing operation progress and completion. Addressables is designed to content location agnostic. The content may need to be downloaded first or use other methods that can take a long time. To force synchronous execution, See [Synchronous Addressables](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/SynchronousAddressables.html) for more information.  
可寻址对象系统 API 是异步的，并返回一个[AsyncOperationHandle](xref:UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle-1) ，用于管理操作进度和完成。可寻址对象旨在与内容位置无关。内容可能需要先下载或使用可能需要很长时间的其他方法。若要强制同步执行，请参阅[Synchronous Addressables](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/SynchronousAddressables.html)以获取详细信息。

When loading an asset for the first time, the handle is done after a minimum of one frame. You can wait until the load has completed using different methods as shown below. If the content has already been loaded, execution times may differ between the various asynchronous loading options shown below.  
首次加载资源时，handle在至少一帧后完成。您可以使用如下所示的不同方法等待加载完成。如果内容已加载，则下面显示的各种异步加载选项之间的执行时间可能会有所不同。

* [Coroutine](xref:UnityEngine.Coroutine*): Always be delayed at minimum of one frame before execution continues.  
  [Coroutine](xref:UnityEngine.Coroutine*)：在继续执行之前，始终至少延迟一帧。
* [Completed callback](xref:UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle-1.Completed.html): Is a minimum of one frame if the content has not already been loaded, otherwise the callback is invoked in the same frame.  
  [Completed callback](xref:UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle-1.Completed.html)：如果内容尚未加载，则至少为一帧，否则在同一帧中调用回调。
* Awaiting [AsyncOperationHandle.Task](xref:UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle-1.Task.html): Is a minimum of one frame if the content has not already been loaded, otherwise the execution continues in the same frame.  
  Await [AsyncOperationHandle.Task](xref:UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle-1.Task.html)：如果内容尚未加载，则至少为一帧，否则执行将在同一帧中继续。

using System.Collections;

using UnityEngine;

using UnityEngine.AddressableAssets;

using UnityEngine.ResourceManagement.AsyncOperations;

internal class AsynchronousLoading : MonoBehaviour

{

private string address = "tree";

private AsyncOperationHandle loadHandle;

// always minimum of 1 frame

IEnumerator LoadAssetCoroutine() {

loadHandle = Addressables.LoadAssetAsync<GameObject>(address);

yield return loadHandle;

}

// minimum of 1 frame for new asset loads

// callback called in current frame for already loaded assets

void LoadAssetCallback() {

loadHandle = Addressables.LoadAssetAsync<GameObject>(address);

loadHandle.Completed += h =>

{

// Loaded here

};

}

// minimum of 1 frame for new asset loads

// await completes in current frame for already loaded assets

async void LoadAssetWait() {

loadHandle = Addressables.LoadAssetAsync<GameObject>(address);

await loadHandle.Task;

}

private void OnDestroy() {

Addressables.Release(loadHandle);

}

}

### Releasing Addressable assets 释放可寻址资产

Because the Addressables system uses reference counting to determine whether an asset is in use, you must release every asset that you load or instantiate when you are done with it. See [Memory Management](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/MemoryManagement.html) for more information.  
由于可寻址对象系统使用引用计数来确定资产是否正在使用中，因此您必须在完成加载或实例化后释放该资产的每个资产。 有关详细信息，请参阅[内存管理](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/MemoryManagement.html)。

When you unload a Scene, implicit assets in the Scene are not automatically unloaded. You must call [Resources.UnloadUnusedAssets](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Resources.UnloadUnusedAssets.html) or [Resources.UnloadAsset](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Resources.UnloadAsset.html) to free these assets. Note that the Unity runtime automatically calls UnloadUnusedAssets when you load a Scene using the [LoadSceneMode.Single](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/SceneManagement.LoadSceneMode.Single.html) mode.  
卸载场景时，场景中的隐式资源不会自动卸载。您必须调用[Resources.UnloadUnusedAssets](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Resources.UnloadUnusedAssets.html) 或[Resources.UnloadAsset](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Resources.UnloadAsset.html)来释放这些资产。请注意，当您使用[LoadSceneMode.Single](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/SceneManagement.LoadSceneMode.Single.html) 模式加载场景时，Unity 运行时会自动调用 UnloadUnusedAssets 。

### Using Addressables in a Scene 在场景中使用可寻址对象

If a Scene is itself Addressable, you can use Addressable assets in the scene just as you would any assets. You can place Prefabs and other assets in the Scene, assign assets to component properties, and so on. If you use an asset that is not Addressable, that asset becomes an implicit dependency of the Scene and the build system packs it in the same AssetBundle as the Scene when you make a content build. (Addressable assets are packed into their own AssetBundles according to the group they are in.)  
如果场景本身是可寻址的，则可以像使用任何资源一样在场景中使用可寻址资源。您可以在场景中放置预制件和其他资源，将资源分配给组件属性等。如果使用不可寻址的资产，则该资产将成为场景的隐式依赖项，并且在您进行内容构建时，构建系统会将其打包到与场景相同的 AssetBundle 中。（相对的，可寻址资产根据它们所在的组打包到它们自己的资产包中）。

##### NOTE

Implicit dependencies used in more than one place can be duplicated in multiple AssetBundles and in the built-in scene data. Use the [Check Duplicate Bundle Dependencies](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AnalyzeTool.html#check-duplicate-bundle-dependencies) rule in the Analyze tool to find unwanted duplication of assets.  
在多个位置使用的隐式依赖项会在多个 AssetBundles 和内置场景数据中复制。使用分析工具中的[Check Duplicate Bundle Dependencies](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AnalyzeTool.html" \l "check-duplicate-bundle-dependencies) 规则来查找不需要的资源重复项。

If a Scene is NOT Addressable, then any Addressable assets you add directly to the scene hierarchy become implicit dependencies and Unity includes copies of those assets in the built-in scene data even if they also exist in an Addressable group. The same is true for any assets, such as Materials, assigned to a component on a GameObject in the scene.  
如果场景不可寻址，则直接添加到场景层次结构的任何可寻址资源都将成为隐式依赖项，Unity 会在内置场景数据中包含这些资源的副本，即使它们也存在于可寻址组中也是如此。分配给场景中游戏对象上组件的任何资源（例如材质）也是如此。

In custom component classes, you can use [AssetReference](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.AssetReference.html) fields to allow the assignment of Addressable assets in non-Addressable scenes. Otherwise, you can use [addresses](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsOverview.html#asset-addresses) and [labels](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Labels.html) to load assets at runtime from a script. Note that you must load an AssetReference in code whether or not the Scene is Addressable.  
在自定义组件类中，可以使用[AssetReference](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.AssetReference.html) 字段允许在不可寻址场景中分配可寻址资源。否则，您可以使用[addresses](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsOverview.html#asset-addresses) 和[labels](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Labels.html) 在运行时从脚本加载资产。请注意，无论场景是否可寻址，都必须在代码中加载AssetReference。

## Operations 操作（句柄）

Many tasks in the Addressables need to load or download information before they can return a result. To avoid blocking program execution, Addressables implements such tasks as asynchronous operations.  
可寻址对象中的许多任务需要加载或下载信息，然后才能返回结果。为了避免阻塞程序执行，Addressables 实现了异步操作等任务。

In contrast to a synchronous operation, which doesn’t return control until the result is available, an asynchronous operation returns control to the calling function almost immediately. However, the results may not be available until some time in the future. When you call a function, such as [LoadAssetAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetAsync.html), it doesn't return the loaded assets directly. Instead, it returns an [AsyncOperationHandle](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.html) object, which you can use to access the loaded assets when they become available.  
与同步操作相反，同步操作在结果可用之前不会返回控制权，而异步操作几乎立即将控制权返回给调用函数。但是，结果可能要到将来的某个时候才能获得。调用函数（如 [LoadAssetAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetAsync.html)）时，它不会直接返回加载的资产。相反，它返回一个 [AsyncOperationHandle](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.html) 对象，您可以使用该对象在加载的资产可用时访问它们。

You can use the following techniques to wait for the results of an asynchronous operation (while allowing other scripts to continue processing).  
可以使用以下技术等待异步操作的结果（同时允许其他脚本继续处理）。

* [Coroutines and IEnumerator loops](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsAsyncOperationHandle.html#coroutine-operation-handling)
* [Events](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsAsyncOperationHandle.html#event-based-operation-handling)
* [Tasks](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsAsyncOperationHandle.html#task-based-operation-handling)

##### NOTE

You can block the current thread to wait for the completion of an asynchronous operation. Doing so can introduce performance problems and frame rate hitches. See [Using operations synchronously](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsAsyncOperationHandle.html#using-operations-synchronously).  
您可以阻止当前线程以等待异步操作完成。这样做可能会引入性能问题和帧速率卡顿。请参阅[同步使用操作](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsAsyncOperationHandle.html#using-operations-synchronously)。

### Releasing AsyncOperationHandle instances 释放异步操作句柄实例

Methods, like [LoadAssetsAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetsAsync.html), return [AsyncOperationHandle](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.html) instances that both provide the results of the operation and a way to release both the results and the operation object itself. You must retain the handle object for as long as you want to use the results. Depending on the situation, that might be one frame, until the end of a level, or even the lifetime of the application. Use the [Addressables.Release](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.Release.html) function to release operation handles and any associated addressable assets.  
方法（如 [LoadAssetsAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetsAsync.html)）返回 [AsyncOperationHandle](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.html) 实例，这些实例既提供操作的结果，也提供释放结果和操作对象本身的方法。如果要使用结果，就必须保留句柄对象。根据具体情况，这可能是一帧，一关，甚至是应用程序的整个生存周期。使用 [Addressables.Release](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.Release.html) 函数释放操作句柄和任何关联的可寻址资产。

Releasing an operation handle decrements the reference count of any assets loaded by the operation and invalidates the operation handle object itself. See [Memory management](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/MemoryManagement.html) for more information about reference counting in the Addressables system.  
释放操作句柄会递减操作加载的任何资产的引用计数，并使操作句柄对象本身失效。有关可寻址系统中引用计数的详细信息，请参阅[内存管理](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/MemoryManagement.html)。

In cases where you don’t need to use the results of an operation beyond a limited scope, you can release the handles right away. A few Addressables methods, such as [UnloadSceneAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.UnloadSceneAsync.html) allow you to automatically release the operation handle when it's complete.  
如果不需要使用操作的结果，则可以立即释放句柄。一些可寻址对象方法（如 [UnloadSceneAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.UnloadSceneAsync.html)）允许您在操作句柄完成时自动释放操作句柄。

If an operation is unsuccessful, you should still release the operation handle. Normally, Addressables releases any assets that it loaded during a failed operation, but releasing the handle still clears the handle’s instance data. Note that some functions, like [LoadAssetsAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetsAsync.html), which load multiple assets, give you the option to either retain any assets that it could load or to fail and release everything if any part of the load operation failed.  
如果操作不成功，仍应释放操作句柄。通常，Addressables 会释放在失败操作期间加载的任何资产，但释放句柄仍会清除句柄的实例数据。请注意，某些函数（如加载多个资产的 [LoadAssetsAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetsAsync.html)）允许您选择保留可以加载的任何资产，或者在加载操作的任何部分失败时失败并释放所有内容。

### Coroutine- and IEnumerator-based operation handling 基于协程和 IEnumerator的操作处理

The [AsyncOperationHandle](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.html) implements the [IEnumerator](https://docs.microsoft.com/dotnet/api/system.collections.ienumerator) interface and will continue iteration until the operation is complete. In a coroutine, you can yield the operation handle to wait for the next iteration. When complete, the execution flow continues to the following statements. Recall that you can implement the [MonoBehaviour Start](https://docs.unity3d.com/ScriptReference/MonoBehaviour.Start.html) function as a coroutine, which is a good way to have a GameObject load and instantiate the assets it needs.  
[AsyncOperationHandle](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.html) 实现 [IEnumerator](https://docs.microsoft.com/dotnet/api/system.collections.ienumerator) 接口，并将继续迭代，直到操作完成。在协程中，可以生成操作句柄以等待下一次迭代。完成后，执行流将继续执行以下语句。回想一下，您可以将 [MonoBehavior Start](https://docs.unity3d.com/ScriptReference/MonoBehaviour.Start.html) 函数实现为协程，这是加载游戏对象并实例化其所需资产的好方法。

The following script loads a Prefab as a child of its GameObject using a Start function coroutine. It yields the AsyncOperationHandle until the operation finishes and then uses the same handle to instantiate the Prefab.  
以下脚本使用 Start 函数协程将预制件作为其游戏对象的子级加载。它会生成 AsyncOperationHandle，直到操作完成，然后使用相同的句柄来实例化预制件。

using System.Collections;

using UnityEngine;

using UnityEngine.AddressableAssets;

using UnityEngine.ResourceManagement.AsyncOperations;

internal class LoadWithIEnumerator : MonoBehaviour

{

public string address;

AsyncOperationHandle<GameObject> opHandle;

public IEnumerator Start() {

opHandle = Addressables.LoadAssetAsync<GameObject>(address);

// yielding when already done still waits until the next frame

// so don't yield if done.

if (!opHandle.IsDone)

yield return opHandle;

if (opHandle.Status == AsyncOperationStatus.Succeeded) {

Instantiate(opHandle.Result, transform);

} else {

Addressables.Release(opHandle);

}

}

void OnDestroy() {

Addressables.Release(opHandle);

}

}

Note that [Addressables.LoadAssetsAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetsAsync.html) is not able to be canceled once started. However, releasing the handle before it has finished will decrement the handle reference count and it will automatically release when the load is complete.  
请注意，[Addressables.LoadAssetsAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetsAsync.html) 一旦启动就无法取消。但是，在句柄完成之前释放句柄将减少句柄引用计数，并在加载完成后自动释放。

See [Coroutines](https://docs.unity3d.com/2019.4/Documentation/Manual/Coroutines.html) for more information.

### Grouping operations in a coroutine 在协程中对操作进行分组

You will probably encounter situations in which you want to perform several operations before moving on to the next step in your game logic. For example, you want to load a number of Prefabs and other assets before you start a level.  
在继续游戏逻辑的下一步之前，您可能会遇到想要执行多个操作的情况。例如，您希望在开始关卡之前加载大量预制件和其他资源。

If the operations all load assets, you can combine them with a single call to the [Addressables.LoadAssetsAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetsAsync.html) function. The AsyncOperationhandle for this method works the same as [LoadAssetAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetAsync.html); you can yield the handle in a coroutine to wait until all the assets in the operation load. In addition, you can pass a callback function to LoadAssetsAsync and the operation calls that function when it finishes loading a specific asset. See [Loading multiple assets](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadingAddressableAssets.html#loading-multiple-assets) for an example.  
如果操作都是加载资产，则可以使用 [Addressables.LoadAssetsAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetsAsync.html) 函数一次性加载它们。此方法的 AsyncOperationHandle 的工作方式与 [LoadAssetAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetAsync.html) 相同;您可以在协程中生成句柄，以等待操作中的所有资产加载。此外，您可以将回调函数传递给 LoadAssetsAsync，操作会在该函数完成加载特定资产时调用该函数。有关示例，请参阅[Loading multiple assets](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadingAddressableAssets.html#loading-multiple-assets)。

Another option is to use the [ResourceManager.CreateGenericGroupOperation](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceManager.CreateGenericGroupOperation.html) to create a group operation that completes when all of its members finish.  
另一种选择是使用 [ResourceManager.CreateGenericGroupOperation](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceManager.CreateGenericGroupOperation.html) 创建一个组操作，该操作在其所有成员完成时完成。

### Event-based operation handling 基于事件的操作处理

You can add a delegate function to the [Completed](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.Completed.html) event of an [AsyncOperationHandle](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.html). The operation calls the delegate function when it's finished.  
可以将委托函数添加到 [AsyncOperationHandle](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.html) 的 [Completed](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.Completed.html) 事件中。该操作在完成后调用委托函数。

The following script performs the same function as the example in [Coroutine- and IEnumerator-based operation handling](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsAsyncOperationHandle.html#coroutine-operation-handling), but uses an event delegate instead of a coroutine.  
以下脚本执行 [Coroutine- and IEnumerator-based operation handling](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsAsyncOperationHandle.html#coroutine-operation-handling)中的示例相同的功能，但使用事件委托而不是协程。

using UnityEngine;

using UnityEngine.AddressableAssets;

using UnityEngine.ResourceManagement.AsyncOperations;

internal class LoadWithEvent : MonoBehaviour

{

public string address;

AsyncOperationHandle<GameObject> opHandle;

void Start() {

// Create operation

opHandle = Addressables.LoadAssetAsync<GameObject>(address);

// Add event handler

opHandle.Completed += Operation\_Completed;

}

private void Operation\_Completed(AsyncOperationHandle<GameObject> obj) {

if (obj.Status == AsyncOperationStatus.Succeeded) {

Instantiate(obj.Result, transform);

} else {

Addressables.Release(obj);

}

}

void OnDestroy() {

Addressables.Release(opHandle);

}

}

Note that the handle instance passed to the event delegate is the same as that returned by the original function call. You can use either to access the results and status of the operation and, ultimately, to release the operation handle and loaded assets.  
请注意，传递给事件委托的句柄实例与原始函数调用返回的句柄实例相同。您可以使用其中之一来访问操作的结果和状态，并最终释放操作句柄和它所加载的资产。

### Task-based operation handling 基于任务的操作处理

The [AsyncOperationHandle](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.html) provides a [Task](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.Task.html#UnityEngine_ResourceManagement_AsyncOperations_AsyncOperationHandle_Task) object that you can use with the C# [async](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/async) and [await](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/await) keywords to sequence code that calls asynchronous functions and handles the results.  
[AsyncOperationHandle](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.html) 提供了一个 [Task](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.Task.html#UnityEngine_ResourceManagement_AsyncOperations_AsyncOperationHandle_Task) 对象，您可以将其与 C# [async](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/async)和 [await](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/await) 关键字一起使用，以对调用异步函数和处理结果的代码进行排序。

The following example loads Addressable assets using a list of keys. The differences between this task-based approach and the coroutine or event-based approaches are in the signature of the calling function, which must include the [async](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/async) keyword and the use of the [await](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/await) keyword with the operation handle’s Task property. The calling function, Start() in this case, suspends operation while the task finishes. Execution then resumes and the example instantiates all the loaded Prefabs (in a grid pattern).  
以下示例使用键列表加载可寻址资产。这种基于Task的方法与协程或基于事件的方法之间的区别在于调用函数的签名，该签名必须包括 [async](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/async) 关键字以及将 [await](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/await) 关键字与操作句柄的 [Task](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.Task.html#UnityEngine_ResourceManagement_AsyncOperations_AsyncOperationHandle_Task) 属性一起使用。在本例中，调用函数 Start（） 在任务完成时暂停操作。然后恢复执行，该示例实例化所有加载的预制件（以网格模式）。

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.AddressableAssets;

using UnityEngine.ResourceManagement.AsyncOperations;

internal class LoadWithTask : MonoBehaviour

{

// Label or address strings to load

public List<string> keys = new List<string>() { "characters", "animals" };

// Operation handle used to load and release assets

AsyncOperationHandle<IList<GameObject>> loadHandle;

public async void Start() {

loadHandle = Addressables.LoadAssetsAsync<GameObject>(

keys, // Either a single key or a List of keys

addressable => {

// Called for every loaded asset

Debug.Log(addressable.name);

}, Addressables.MergeMode.Union, // How to combine multiple labels

false); // Whether to fail if any asset fails to load

// Wait for the operation to finish in the background

await loadHandle.Task;

// Instantiate the results

float x = 0, z = 0;

foreach (var addressable in loadHandle.Result) {

if (addressable != null) {

Instantiate<GameObject>(addressable,

new Vector3(x++ \* 2.0f, 0, z \* 2.0f),

Quaternion.identity,

transform); // make child of this object

if (x > 9) {

x = 0;

z++;

}

}

}

}

private void OnDestroy() {

Addressables.Release(loadHandle);

// Release all the loaded assets associated with loadHandle

// Note that if you do not make loaded addressables a child of this object,

// then you will need to devise another way of releasing the handle when

// all the individual addressables are destroyed.

}

}

##### IMPORTANT

The [AsyncOperationHandle.Task](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.Task.html#UnityEngine_ResourceManagement_AsyncOperations_AsyncOperationHandle_Task) property is not available on the Unity WebGL platform, which doesn't support multitasking.  
[AsyncOperationHandle.Task](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.Task.html#UnityEngine_ResourceManagement_AsyncOperations_AsyncOperationHandle_Task) 属性在不支持多任务处理的 Unity WebGL 平台上不可用

When you use Task-based operation handling, you can use the C# [Task](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.Task.html#UnityEngine_ResourceManagement_AsyncOperations_AsyncOperationHandle_Task) class methods such as [WhenAll](https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.task.whenall) to control which operations you run in parallel and which you want to run in sequence. The following example illustrates how to wait for more than one operation to finish before moving onto the next task:  
使用基于任务的操作处理时，可以使用 C# [Task](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.Task.html#UnityEngine_ResourceManagement_AsyncOperations_AsyncOperationHandle_Task)  类方法（如 [WhenAll](https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.task.whenall) ）来控制并行运行哪些操作以及要按顺序运行哪些操作。下面的示例说明如何等待多个操作完成，然后再转到下一个任务：

// Load the Prefabs

var prefabOpHandle = Addressables.LoadAssetsAsync<GameObject>(

keys, null, Addressables.MergeMode.Union, false);

// Load a Scene additively

var sceneOpHandle

= Addressables.LoadSceneAsync(nextScene,

UnityEngine.SceneManagement.LoadSceneMode.Additive);

await System.Threading.Tasks.Task.WhenAll(prefabOpHandle.Task, sceneOpHandle.Task);

### Using operations synchronously 同步操作

You can wait for an operation to finish without yielding, waiting for an event, or using async await by calling an operation’s [WaitForCompletion](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.WaitForCompletion.html) method. This method blocks the current program execution thread while it waits for the operation to finish before continuing in the current scope.  
可以通过调用操作的 [WaitForCompletion](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.WaitForCompletion.html) 方法来等待操作完成而不使用 yielding、event或 async await。此方法在当前程序执行线程等待操作完成时阻止当前程序执行线程，然后再在当前范围内继续。

Avoid calling WaitForCompletion on operations that can take a significant amount of time, such as those that must download data. Calling WaitForCompletion can cause frame hitches and interrupt UI responsiveness.  
避免对可能需要大量时间的操作（例如必须下载数据的操作）调用 WaitForComplete。调用 WaitForComplete可能会导致帧卡顿并中断 UI 响应能力。

In Unity 2020.1 or earlier, Unity also waits for all other pending asynchronous operations to finish, so the delay in execution can be much longer than that required for just the single operation for which you call this method. In Unity 2020.2 or later, the performance impact can be less pronounced, at least when loading assets that have already been downloaded.  
在 Unity 2020.1 或更早版本中，Unity 还会等待所有其他挂起的异步操作完成，因此执行延迟可能比仅调用此方法的单个操作所需的延迟长得多。在 Unity 2020.2 或更高版本中，性能影响可能不太明显，至少在加载已下载的资源时是这样。

The following example loads a Prefab asset by address, waits for the operation to complete, and then instantiates the Prefab:  
以下示例按地址加载预制件资产，等待操作完成，然后实例化预制件：

using UnityEngine;

using UnityEngine.AddressableAssets;

using UnityEngine.ResourceManagement.AsyncOperations;

internal class LoadSynchronously : MonoBehaviour

{

public string address;

AsyncOperationHandle<GameObject> opHandle;

void Start() {

opHandle = Addressables.LoadAssetAsync<GameObject>(address);

opHandle.WaitForCompletion(); // Returns when operation is complete

if (opHandle.Status == AsyncOperationStatus.Succeeded) {

Instantiate(opHandle.Result, transform);

} else {

Addressables.Release(opHandle);

}

}

void OnDestroy() {

Addressables.Release(opHandle);

}

}

### Custom operations 自定义操作

To create a custom operation, extend the [AsyncOperationBase](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationBase-1.html) class and override its virtual methods.  
若要创建自定义操作，请扩展[AsyncOperationBase](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationBase-1.html) 类并重写其虚拟方法。

You can pass the derived operation to the [ResourceManager.StartOperation](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceManager.StartOperation.html) method to start the operation and receive an [AsyncOperationHandle](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.html) struct. The [ResourceManager](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceManager.html) registers operations started this way and shows them in the Addressables [Event Viewer](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/EventViewer.html).  
可以将派生操作传递给 [ResourceManager.StartOperation](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceManager.StartOperation.html) 方法以启动该操作并接收 [AsyncOperationHandle](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.html) 结构。[ResourceManager](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceManager.html)注册以这种方式启动的操作，并在可寻址[Event Viewer](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/EventViewer.html)中显示这些操作。

#### Executing the operation 执行操作

The [ResourceManager](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceManager.html) invokes the [AsyncOperationBase.Execute](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationBase-1.Execute.html) method for your custom operation once the optional dependent operation completes.  
[ResourceManager](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceManager.html)在可选相关操作完成后调用自定义操作的 [AsyncOperationBase.Execute](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationBase-1.Execute.html) 方法。

#### Completion handling Completion处理

When your custom operation completes, call [AsyncOperationBase.Complete](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationBase-1.Complete.html) on your custom operation object. You can call this within the [Execute](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationBase-1.Execute.html) method or defer it to outside the call. AsyncOperationBase.Complete notifies the [ResourceManager](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceManager.html) that the operation has finished. The ResourceManager invokes the associated [AsyncOperationHandle.Completed](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.Completed.html) events for the relevant instances of the custom operation.  
自定义操作完成后，对自定义操作对象调用 [AsyncOperationBase.Complete](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationBase-1.Complete.html)。可以在 [Execute](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationBase-1.Execute.html) 方法中调用它，也可以将其推迟到调用外部。AsyncOperationBase.Complete 通知 [ResourceManager](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceManager.html) 操作已完成。ResourceManager调用自定义 操作的相关实例所关联的[AsyncOperationHandle.Completed](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.Completed.html)事件。

#### Terminating the operation 终止操作

The [ResourceManager](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceManager.html) invokes the [AsyncOperationBase.Destroy](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationBase-1.Destroy.html) method for your custom operation when you release the [AsyncOperationHandle](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.html) that references it. This is where you should release any memory or resources associated with your custom operation.  
释放引用自定义操作的 [AsyncOperationHandle](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.html) 时，[ResourceManager](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceManager.html)将调用自定义操作的 [AsyncOperationBase.Destroy](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationBase-1.Destroy.html) 方法。应在此处释放与自定义操作关联的任何内存或资源。

### Using typed versus typeless operation handles 使用类型化与无类型操作句柄

Most [Addressables](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.html) methods that start an operation return a generic [AsyncOperationHandle<T>](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle-1.html) struct, allowing type safety for the [AsyncOperationHandle.Completed](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.Completed.html) event and for the [AsyncOperationHandle.Result](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.Result.html#UnityEngine_ResourceManagement_AsyncOperations_AsyncOperationHandle_Result) object. You can also use a non-generic [AsyncOperationHandle](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.html) struct and convert between the two handle types as desired.  
operation的大多数[Addressables](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.html) 方法返回 [AsyncOperationHandle<T>](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle-1.html) 泛型结构，从而使得 [AsyncOperationHandle.Completed](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.Completed.html) 事件和 [AsyncOperationHandle.Result](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.Result.html#UnityEngine_ResourceManagement_AsyncOperations_AsyncOperationHandle_Result) 对象的类型安全。还可以使用非泛型 [AsyncOperationHandle](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.html) 结构，并根据需要在两种句柄类型之间进行转换。

Note that a runtime exception occurs if you attempt to cast a non-generic handle to a generic handle of an incorrect type. For example:  
请注意，如果尝试将非泛型句柄强制转换为类型不正确的泛型句柄，则会发生运行时异常。例如：

// Load asset using typed handle:

AsyncOperationHandle<Texture2D> textureHandle = Addressables.LoadAssetAsync<Texture2D>("mytexture");

// Convert the AsyncOperationHandle<Texture2D> to an AsyncOperationHandle:

AsyncOperationHandle nonGenericHandle = textureHandle;

// Convert the AsyncOperationHandle to an AsyncOperationHandle<Texture2D>:

AsyncOperationHandle<Texture2D> textureHandle2 = nonGenericHandle.Convert<Texture2D>();

// This will throw and exception because Texture2D is required:

AsyncOperationHandle<Texture> textureHandle3 = nonGenericHandle.Convert<Texture>();

### Reporting operation progress 报告操作进度

[AsyncOperationHandle](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.html) has two methods that you can use to monitor and report the progress of the operation:  
[AsyncOperationHandle](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.html) 有两种可用于监视和报告操作进度的方法：

* [GetDownloadStatus](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.GetDownloadStatus.html) returns a [DownloadStatus](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.DownloadStatus.html) struct. This struct contains information about how many bytes have been downloaded and how many bytes still need to be downloaded. The [DownloadStatus.Percent](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.DownloadStatus.Percent.html#UnityEngine_ResourceManagement_AsyncOperations_DownloadStatus_Percent) reports the percentage of bytes downloaded.  
  [GetDownloadStatus](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.GetDownloadStatus.html) 返回一个 [DownloadStatus](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.DownloadStatus.html) 结构。此结构包含有关已下载的字节数以及仍需要下载的字节数的信息。[DownloadStatus.Percent](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.DownloadStatus.Percent.html#UnityEngine_ResourceManagement_AsyncOperations_DownloadStatus_Percent) 报告下载的字节百分比。
* [AsyncOperationHandle.PercentComplete](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.PercentComplete.html#UnityEngine_ResourceManagement_AsyncOperations_AsyncOperationHandle_PercentComplete) returns an equally-weighted aggregate percentage of all the sub-operations that are complete. For example, if an operation has five sub-operations, each of them represents 20% of the total. The value doesn't factor in the amount of data that must be downloaded by the individual sub-operations.  
  [AsyncOperationHandle.PercentComplete](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.PercentComplete.html#UnityEngine_ResourceManagement_AsyncOperations_AsyncOperationHandle_PercentComplete) 返回所有已完成的子操作的等权聚合百分比。例如，如果一个操作有五个子操作，则每个子操作占总数的 20%。该值不考虑各个子操作必须下载的数据量。

For example, if you called [Addressables.DownloadDependenciesAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.DownloadDependenciesAsync.html) and five AssetBundles needed to be downloaded, GetDownloadStatus would tell you what percentage of the total number of bytes for all sub-operations had been downloaded so far. PercentComplete would tell you what percentage of the number of operations had finished, regardless of their size.  
例如，如果您调用 [Addressables.DownloadDependenciesAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.DownloadDependenciesAsync.html)，并且需要下载五个 AssetBundle，则 GetDownloadStatus 将告诉您到目前为止已下载的所有子操作的总字节数的百分比。PercentComplete 会告诉您已完成操作数的百分比，无论其大小如何。

On the other hand, if you called [LoadAssetAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetAsync.html), and one bundle had to be downloaded before an asset could be loaded from it, the download percentage might be misleading. The values obtained from GetDownloadStatus would reach 100% before the operation finished, because the operation had additional sub-operations to conduct. The value of PercentComplete would be 50% when the download sub-operation finished and 100% when the actual load into memory was complete.  
另一方面，如果您调用 [LoadAssetAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadAssetAsync.html)，并且必须先下载一个捆绑包，然后才能从中加载资产，则下载百分比可能会产生误导。在操作完成之前，从 GetDownloadStatus 获取的值将达到 100%，因为该操作有其他子操作要执行。下载子操作完成时，PercentComplete 的值为 50%，实际加载到内存时为 100%。

## Synchronous Workflow 同步工作流

Synchronous Addressables APIs help to more closely mirror Unity asset loading workflows. AsyncOperationHandles now have a method called WaitForCompletion() that force the async operation to complete and return the Result of the operation.  
同步可寻址 API 有助于更紧密地反映 Unity 资源加载工作流程。 AsyncOperationHandles现在有一个名为WaitForCompletion（）的方法，它强制异步操作完成并返回操作 的结果。

### API

TObject WaitForCompletion()

### Result 结果

The result of WaitForCompletion is the Result of the async operation it is called on. If the operation fails, this returns default(TObject).  
 WaitForCompletion的结果是 调用它的异步操作的Result。如果操作失败，则返回 default(TObject)。

It is possible to get a default(TObject) for a result when the operation doesn't fail. Async operations that auto release their AsyncOperationHandles on completion are such cases. Addressables.InitializeAsync() and any API with a autoReleaseHandle parameter set to true will return default(TObject) even though the operations themselves succeeded.  
当操作未失败时，Resule也可能是default(TObject)。在完成时自动释放其AsyncOperationHandles 的异步操作就是这种情况。 Addressables.InitializeAsync（） 和任何将 autoReleaseHandle 参数设置为 true 的 API 都将返回 default（TObject），即使操作本身成功也是如此。

### Performance 性能

It is worth noting that calling WaitForCompletion may have performance implications on your runtime when compared to Resources.Load or Instantiate calls directly. If your AssetBundle is local or has been previously downloaded and cached, these performance hits are likely to be negligible. However, this may not be the case for your individual project setup.  
值得注意的是，与直接调用 Resources.Load 或 Instantiate调用相比，调用 WaitForCompletion 可能会对运行时产生性能影响。如果您的 AssetBundle 是本地的，或者之前已下载并缓存过，则这些性能影响可能可以忽略不计。但是，对于您的个人项目设置，情况可能并非如此。

All currently active Asset Load operations are completed when WaitForCompletion is called on any Asset Load operation, due to how Async operations are handled in the Engine. To avoid unexpected stalls, use WaitForCompletion when the current operation count is known, and the intention is for all active operations to complete synchronously.  
由于引擎中处理异步操作的方式，当对任何资产加载操作调用 WaitForFinish 时，所有当前处于活动状态的资源加载操作都将完成。若要避免意外停顿，请在当前操作计数已知时使用 WaitForComplete，并且目的是同步完成所有活动操作。

When using WaitForCompletion, there are performance implications. When using 2021.2.0 or newer, these are minimal. Using an older version can result in delays that scale with the number of Engine Asset load calls that are loading when WaitForCompletion is called.  
使用 WaitForCompletion时，会对性能产生影响。使用 2021.2.0 或更高版本时，这些是最小的。使用旧版本可能会导致延迟，延迟会随着调用WaitForCompletion时Engine Asset load calls的数量而扩展 。

It is not recommended that you call WaitForCompletion on an operation that is going to fetch and download a remote AssetBundle. Though, it is possible if that fits your specific situation.  
建议您不要在将要获取和下载远程AssetBundle的操作上调用 WaitForComplete。但是，如果这符合您的具体情况，这是可能的。

### Code Sample

void Start()

{

//Basic use case of forcing a synchronous load of a GameObject

var op = Addressables.LoadAssetAsync<GameObject>("myGameObjectKey");

GameObject go = op.WaitForCompletion();

//Do work...

Addressables.Release(op);

}

#### Scenes 场景

Due to engine limitations scenes cannot be completed synchronously. Calling WaitForCompletion on an operation returned from [Addressables.LoadSceneAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadSceneAsync.html) will not completely load the scene, even if activateOnLoad is set to true. It will wait for dependencies and assets to complete but the scene activation must be done asynchronously. This can be done using the sceneHandle, or by the [AsyncOperation](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/AsyncOperation.html) from ActivateAsync on the SceneInstance as shown below.  
由于引擎限制，场景无法同步完成。对从 [Addressables.LoadSceneAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadSceneAsync.html) 返回的操作调用 WaitForCompletion时，并不会完全加载场景，即使 activateOnLoad 设置为 true。它将等待依赖项和资产完成，但场景激活必须异步完成。这可以使用 sceneHandle 来完成，也可以通过 SceneInstance 上 ActivateAsync 的[AsyncOperation](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/AsyncOperation.html) 来完成，如下所示。

IEnumerator LoadScene(string myScene)

{

var sceneHandle = Addressables.LoadSceneAsync(myScene, LoadSceneMode.Additive);

SceneInstance sceneInstance = sceneHandle.WaitForCompletion();

yield return sceneInstance.ActivateAsync();

//Do work... the scene is now complete and integrated

}

##### NOTE

Unloading a scene cannot be completed synchronously. Calling WaitForCompleted on a scene unload will not unload the scene or any assets, and a warning will be logged to the console.  
无法同步完成场景的卸载。在场景卸载时调用WaitForCompleted 不会卸载场景或任何资产，并且会在控制台中记录警告。

##### NOTE

Due to limitations with Scene integration on the main thread through the SceneManager API, it is possible to lock the Editor or Player when calling WaitForCompletion in association with scene loading. The issue primarily surfaces when loading two scenes in succession, with the second scene load request having WaitForCompletion called from its AsyncOperationHandle. Since scene loading takes extra frames to fully integrate on the main thread, and WaitForCompletion locks the main thread, you could hit a situation where Addressables has been informed by the SceneManager that the first scene is fully loaded, even though it hasn't completed finished all the required operations. At this point, the scene is fully loaded, but the SceneManager attempts to call UnloadUnusedAssets, on the main thread, if the scene was loaded in Single mode. Then, the second scene load request locks the main thread with WaitForCompletion, but cannot begin loading because SceneManager requires the UnloadUnusedAssets to complete before the next scene can begin loading. In order to avoid this deadlock, it is advised that you either load successive scenes asynchronously, or ensure a sufficient delay is added between scene load requests.  
由于通过 SceneManager API 在主线程上集成场景的限制，在调用与场景加载关联的 WaitForCompletion 时，可能锁死编辑器或播放器。该问题主要在连续加载两个场景时出现，第二个场景加载请求从其 AsyncOperationHandle 调用了WaitForComplete。由于场景加载需要额外的帧才能完全集成到主线程上，并且 WaitForFinish 锁定主线程，因此您可能会遇到以下情况：SceneManager 已通知可寻址对象第一个场景已完全加载，即使它尚未完成所有必需的操作。此时，场景已完全加载，但如果场景是在单一模式下加载的，则场景管理器会尝试在主线程上调用 UnloadUnusedAssets。然后，第二个场景加载请求使用 WaitForComplete 锁定主线程，但无法开始加载，因为 SceneManager 要求在下一个场景开始加载之前完成 UnloadUnusedAssets。为了避免这种死锁，建议您异步加载连续的场景，或者确保在场景加载请求之间添加足够的延迟。

#### Synchronous Addressables with Custom Operations 具有自定义操作的同步可寻址对象

Addressables supports custom AsyncOperations which support unique implementations of AsyncOperations . This method can be overridden to implement custom synchronous operations.  
Addressables 支持自定义 AsyncOperations ，这些操作支持 AsyncOperations 的自定义实现。可以重写此方法以实现自定义同步操作。

Custom operations work with ChainOperations and GroupsOperations. If you require chained operations to be completed synchronously, ensure that your custom operations implement InvokeWaitForCompletion and create a ChainOperation using your custom operations. Similarly, GroupOperations are well suited to ensure a collection of AsyncOperations, including custom operations, complete together. Both ChainOperation and GroupOperation have their own implementations of InvokeWaitForCompletion that relies on the InvokeWaitForCompletion implementations of the operations they depend on.  
自定义操作适用于 ChainOperations 和 GroupsOperations。如果需要同步完成链接操作，请确保自定义操作实现 InvokeWaitForCompletion ，并使用自定义操作创建 ChainOperation 。同样，GroupOperations 非常适合确保 AsyncOperations（包括自定义操作）的集合一起完成。 ChainOperation 和 GroupOperation 都有自己的 InvokeWaitForCompletion  实现，这些实现依赖于它们所依赖的操作的 InvokeWaitForCompletion 实现。

#### WebGL

WebGL does not support WaitForCompletion. On WebGL, all files are loaded using a web request. On other platforms, a web request gets started on a background thread and the main thread spins in a tight loop while waiting for the web request to finish. This is how Addressables does it for WaitForCompletion when a web request is used.  
WebGL不支持WaitForCompletion。在WebGL上，所有文件都是使用Web请求加载的。在其他平台上，Web 请求在后台线程上启动，主线程在等待 Web 请求完成时在紧密循环中旋转。这就是 Addressables 在使用 Web 请求时为 WaitForCompletion 执行此操作的方式

Since WebGL is single-threaded, the tight loop blocks the web request and the operation is never allowed to finish. If a web request finishes the same frame it was created, then WaitForCompletion wouldn't have any issue. However, we cannot guarantee this to be the case, and likely it isn't the case for most instances.  
由于 WebGL 是单线程的，因此紧密循环会阻止 Web 请求，并且永远不允许操作完成。如果 Web 请求的创建和完成在同一帧，则 WaitForCompletion 不会有任何问题。但是，我们不能保证情况确实如此，而且在大多数情况下可能并非如此。

## Managing catalogs at runtime 在运行时管理catalogs

By default, the Addressables system manages the catalog automatically at runtime. If you built your application with a remote catalog, the Addressables system automatically checks to see if you have uploaded a new catalog, and, if so, downloads the new version and loads it into memory.  
默认情况下，可寻址对象系统在运行时自动管理目录。如果构建的应用程序含有远程目录，则 Addressables 系统会自动检查您是否上传了新目录，如果是，则下载新版本并将其加载到内存中。

You can load additional catalogs at runtime. For example, you could load a catalog produced by a separate, compatible project to load Addressable assets built by that project. (See [Loading Content from Multiple Projects](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/MultiProject.html).)  
您可以在运行时加载附加目录。例如，您可以加载由单独的兼容项目生成的目录，以加载该项目构建的可寻址资产。（请参阅[Loading Content from Multiple Projects](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/MultiProject.html)。）

If you want to change the default catalog update behavior of the Addressables system, you can turn off the automatic check and check for updates manually. See [Updating catalogs](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadContentCatalogAsync.html#updating-catalogs).  
如果要更改可寻址对象系统的默认目录更新行为，可以关闭自动检查并手动检查更新。请参阅 [Updating catalogs](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadContentCatalogAsync.html#updating-catalogs)。

### Loading additional catalogs 加载附加目录

Use [Addressables.LoadContentCatalogAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadContentCatalogAsync.html) to load additional content catalogs, either from your hosting service or from the local file system. After the operation to load the catalog is finished, you can call any Addressables loading functions using the keys in the new catalog.  
使用[Addressables.LoadContentCatalogAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadContentCatalogAsync.html) 从托管服务或本地文件系统加载附加内容目录。加载目录的操作完成后，可以使用新目录中的键调用任何可寻址对象加载函数。

If you provide the catalog hash file at the same URL as the catalog, Addressables caches the secondary catalog. When the client application loads the catalog in the future, it only downloads a new version of the catalog if the hash changes.   
如果在与目录相同的 URL 处提供目录哈希文件，则可寻址对象将缓存辅助目录。当客户端应用程序将来加载目录时，它仅在哈希更改时下载新版本的目录。

Once you load a catalog, you cannot unload it. You can, however, update a loaded catalog. You must release the operation handle for the operation that loaded the catalog before updating a catalog. See [Updating catalogs](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadContentCatalogAsync.html#updating-catalogs) for more information.  
一旦加载目录，就无法卸载它。但是，您可以更新加载的目录。在更新目录之前，必须释放加载目录的operation handle。有关详细信息，请参阅 [Updating catalogs](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadContentCatalogAsync.html#updating-catalogs) 。

In general, there is no reason to hold on to the operation handle after loading a catalog. You can release it automatically by setting the autoReleaseHandle parameter to true when loading a catalog, as shown in the following example:  
通常，加载目录后没有理由保留操作句柄。通过在加载目录时将 autoReleaseHandle 参数设置为 true，可以自动释放它，如以下示例所示：

public IEnumerator Start() {

//Load a catalog and automatically release the operation handle.

AsyncOperationHandle<IResourceLocator> handle

= Addressables.LoadContentCatalogAsync("path\_to\_secondary\_catalog", true);

yield return handle;

//...

}

##### NOTE

You can use the [Catalog Download Timeout](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html#downloads) property of your Addressables settings to specify a timeout for downloading catalogs.  
可以使用可寻址对象设置的 [Catalog Download Timeout](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetSettings.html#downloads)属性来指定下载目录的超时。

### Updating catalogs 更新目录

If the catalog hash file is available, Addressables checks the hash when loading a catalog to determine if the version at the provided URL is more recent than the cached version of the catalog. You can turn off the default catalog check, if desired, and call the [Addressables.UpdateCatalogs](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.UpdateCatalogs.html) function when you want to update the catalog. If you loaded a catalog manually with [LoadContentCatalogAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadContentCatalogAsync.html), you must release the operation handle before you can update the catalog.  
如果目录哈希文件可用，则 Addressables 将在加载目录时检查哈希，以确定提供的 URL 上的版本是否比本地的缓存版本更新。如果需要，可以关闭默认目录检查，并在要更新目录时调用  [Addressables.UpdateCatalogs](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.UpdateCatalogs.html)  函数。如果使用 [LoadContentCatalogAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.LoadContentCatalogAsync.html)手动加载目录，则必须先释放操作句柄，然后才能更新目录。

When you call the UpdateCatalog function, all other Addressable requests are blocked until the operation is finished. You can release the operation handle returned by UpdateCatalogs immediately after the operation finishes (or set the autoRelease parameter to true).  
调用 UpdateCatalog 函数时，将阻止所有其他可寻址请求，直到操作完成。您可以在操作完成后立即释放 UpdateCatalog 返回的操作句柄（或将自动释放参数设置为 true）。

If you call UpdateCatalog without providing a list of catalogs, Addressables checks all of the currently loaded catalogs for updates.  
如果在不提供目录列表的情况下调用 UpdateCatalog，则可寻址对象将检查所有当前加载的目录以获取更新。

IEnumerator UpdateCatalogs() {

AsyncOperationHandle<List<IResourceLocator>> updateHandle

= Addressables.UpdateCatalogs();

yield return updateHandle;

Addressables.Release(updateHandle);

}

You can also call [Addressables.CheckForCatalogUpdates](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.CheckForCatalogUpdates.html) directly to get the list of catalogs that have updates and then perform the update:  
您还可以直接调用[Addressables.CheckForCatalogUpdates](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.CheckForCatalogUpdates.html) 以获取具有更新的目录列表，然后执行更新：

IEnumerator CheckCatalogs() {

List<string> catalogsToUpdate = new List<string>();

AsyncOperationHandle<List<string>> checkForUpdateHandle

= Addressables.CheckForCatalogUpdates();

checkForUpdateHandle.Completed += op =>

{

catalogsToUpdate.AddRange(op.Result);

};

yield return checkForUpdateHandle;

if (catalogsToUpdate.Count > 0) {

AsyncOperationHandle<List<IResourceLocator>> updateHandle

= Addressables.UpdateCatalogs(catalogsToUpdate);

yield return updateHandle;

Addressables.Release(updateHandle);

}

Addressables.Release(checkForUpdateHandle);

}

##### IMPORTANT

If you update a catalog when you have already loaded content from the related AssetBundles, you can encounter conflicts between the loaded AssetBundles and the updated versions. You can enable the [Unique Bundle Ids](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html#unique-bundle-ids-setting) option in your Addressable settings to eliminate the possibility of bundle ID collisions at runtime. However, enabling this option also means that more AssetBundles must typically be rebuilt when you perform a content update. See [Content update builds](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html) for more information. Another option is to first unload any content and AssetBundles that must be updated, which can be a slow operation.  
如果在已从相关资源包加载内容时更新目录，则加载的资产包与更新版本之间可能会发生冲突。您可以在可寻址设置中启用 [Unique Bundle Ids](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html#unique-bundle-ids-setting)选项，以消除运行时捆绑包 ID 冲突的可能性。但是，启用此选项还意味着在执行内容更新时通常必须重建更多资产包。有关详细信息，请参阅[Content update builds](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html)。另一种选择是首先卸载必须更新的任何内容和资产包，这可能是一个缓慢的操作。

## Transforming resource URLs 转换 URL 资源

Addressables provides the following ways to modify the URLs it uses to load assets at runtime:  
可寻址对象提供了以下方法来修改用于在运行时加载资产的 URL：

* Static properties in a Profile variable
* Implementing an ID transform function
* Implementing a WebRequestOverride method

### Static Profile variables 静态配置文件变量

You can use a static property when defining the [RemoteLoadPath Profile variable](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html) to specify all or part of the URL from which your application loads remote content, including catalogs, catalog hash files, and AssetBundles. See [Profile variable syntax](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html#profile-variable-syntax) for information about specifying a property name in a Profile variable. The value of your static property must be set before Addressables initializes. Changing the value after initialization has no effect.  
在定义[RemoteLoadPath Profile variable](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html)时，您可以使用静态属性来指定应用程序从中加载远程内容（包括目录、目录哈希文件和资产包）的全部或部分 URL。有关在配置文件变量中指定属性名称的信息，请参阅 [Profile variable syntax](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsProfiles.html#profile-variable-syntax) 语法。必须在可寻址对象初始化之前设置静态属性的值。初始化后更改值不起作用。

### ID transform function ID转换函数

You can assign a function to the [Addressables.ResourceManager](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.ResourceManager.html#UnityEngine_AddressableAssets_Addressables_ResourceManager) object's [InternalIdTransformFunc](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceManager.InternalIdTransformFunc.html#UnityEngine_ResourceManagement_ResourceManager_InternalIdTransformFunc) property to individually change the URLs from which Addressables loads assets. You must assign the function before the relevant operation starts, otherwise the default URL is used.  
可以将函数分配给 [Addressables.ResourceManager](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.ResourceManager.html#UnityEngine_AddressableAssets_Addressables_ResourceManager)对象的[InternalIdTransformFunc](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceManager.InternalIdTransformFunc.html#UnityEngine_ResourceManagement_ResourceManager_InternalIdTransformFunc) 属性，以单独更改 Addressables 从中加载资产的 URL。您必须在相关操作开始之前分配函数，否则将使用默认 URL。

Using TransformInternalId grants a fair amount of flexibility, especially in regards to remote hosting. Given a single IResourceLocation, you can transform the ID to point towards a server specified at runtime. This is particularly useful if your server IP address changes or if you use different URLS to provide different variants of your application assets.  
使用 TransformInternalId 提供了相当多的灵活性，尤其是在远程托管方面。给定单个 IResourceLocation，可以将 ID 转换为指向运行时指定的服务器。如果您的服务器 IP 地址发生更改，或者如果您使用不同的 URL 来提供应用程序资产的不同变体，这将特别有用。

The ResourceManager calls your TransformInternalId function when it looks up an asset, passing the [IResourceLocation](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceLocations.IResourceLocation.html) instance for the asset to your function. You can change the [InternalId](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceLocations.IResourceLocation.InternalId.html#UnityEngine_ResourceManagement_ResourceLocations_IResourceLocation_InternalId) property of this IResourceLocation and return the modified object to the ResourceManager.  
资源管理器在查找资产时调用 TransformInternalId 函数，将资产的 [IResourceLocation](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceLocations.IResourceLocation.html)实例传递给函数。您可以更改此 [IResourceLocation](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceLocations.IResourceLocation.html)的 [InternalId](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceLocations.IResourceLocation.InternalId.html#UnityEngine_ResourceManagement_ResourceLocations_IResourceLocation_InternalId) 属性，并将修改的对象返回到资源管理器。

The following example illustrates how you could append a query string to all URLs for AssetBundles:  
以下示例说明了如何将查询字符串追加到资产包的所有URL：

using UnityEngine.ResourceManagement.ResourceLocations;

using UnityEngine.ResourceManagement.ResourceProviders;

using UnityEngine.AddressableAssets;

static class IDTransformer

{

//Implement a method to transform the internal ids of locations

static string MyCustomTransform(IResourceLocation location) {

if (location.ResourceType == typeof(IAssetBundleResource)

&& location.InternalId.StartsWith("http"))

return location.InternalId + "?customQueryTag=customQueryValue";

return location.InternalId;

}

//Override the Addressables transform method with your custom method.

//This can be set to null to revert to default behavior.

[RuntimeInitializeOnLoadMethod]

static void SetInternalIdTransform() {

Addressables.InternalIdTransformFunc = MyCustomTransform;

}

}

### WebRequest override WebRequest 覆盖

You can assign a function to the [Addressables](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.html) object's [WebRequestOverride](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.WebRequestOverride.html#UnityEngine_AddressableAssets_Addressables_WebRequestOverride) property to individually modify the [UnityWebRequest](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Networking.UnityWebRequest.html) from which is used to download files, such as an AssetBundle or catalog json file. You must assign the function before the relevant operation starts, otherwise the default UnityWebRequest is used.  
您可以将函数分配给[Addressables](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.html)对象的 [WebRequestOverride](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.WebRequestOverride.html#UnityEngine_AddressableAssets_Addressables_WebRequestOverride)属性，以单独修改用于从中下载文件（如资产包或目录 json 文件）的 [UnityWebRequest](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Networking.UnityWebRequest.html)。您必须在相关操作开始之前分配函数，否则将使用默认的 UnityWebRequest。

The ResourceManager calls your [WebRequestOverride](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.WebRequestOverride.html#UnityEngine_AddressableAssets_Addressables_WebRequestOverride) function before [UnityWebRequest.SendWebRequest](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Networking.UnityWebRequest.SendWebRequest.html) is called. Passing the UnityWebRequest for the download to your function.  
ResourceManager 在调用 [UnityWebRequest.SendWebRequest](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Networking.UnityWebRequest.SendWebRequest.html) 之前调用您的[WebRequestOverride](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.WebRequestOverride.html#UnityEngine_AddressableAssets_Addressables_WebRequestOverride) 函数。将用于下载的UnityWebRequest传递给您的函数。

The following example illustrates how you could append a query string to all URLs for AssetBundles and catalogs:

using UnityEngine;

using UnityEngine.Networking;

using UnityEngine.AddressableAssets;

internal class WebRequestOverride : MonoBehaviour

{

//Register to override WebRequests Addressables creates to download

private void Start()

{

Addressables.WebRequestOverride = EditWebRequestURL;

}

//Override the url of the WebRequest, the request passed to the method is what would be used as standard by Addressables.

private void EditWebRequestURL(UnityWebRequest request)

{

if (request.url.EndsWith(".bundle"))

request.url = request.url + "?customQueryTag=customQueryValue";

else if (request.url.EndsWith(".json") || request.url.EndsWith(".hash"))

request.url = request.url + "?customQueryTag=customQueryValue";

}

}

## Preloading dependencies 预加载依赖项

When you distribute content remotely, you can sometimes improve perceived performance by downloading dependencies in advance of when your application needs them. For example, you can download essential content on start up the first time a player launches your game to make sure that they don't have to wait for content in the middle of game play.  
远程分发内容时，有时可以通过提前下载依赖项来提高感知性能。例如，您可以在玩家首次启动游戏时下载基本内容，以确保他们不必在游戏过程中等待内容。

### Downloading dependencies 下载依赖项

Use the [Addressables.DownloadDependenciesAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.DownloadDependenciesAsync.html) method to make sure that all the dependencies needed to load an Addressable key are available either in local content installed with the app or the download cache.  
使用  [Addressables.DownloadDependenciesAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.DownloadDependenciesAsync.html) 方法确保加载Addressable key所需的所有依赖项在随应用或下载缓存一起安装的本地内容中可用。

string key = "assetKey";

// Check the download size

AsyncOperationHandle<long> getDownloadSize = Addressables.GetDownloadSizeAsync(key);

yield return getDownloadSize;

//If the download size is greater than 0, download all the dependencies.

if (getDownloadSize.Result > 0) {

AsyncOperationHandle downloadDependencies = Addressables.DownloadDependenciesAsync(key);

yield return downloadDependencies;

}

##### TIP

if you have a set of assets that you want to pre-download, you can assign the same label, such as "preload", to the assets and use that label as the key when calling [Addressables.DownloadDependenciesAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.DownloadDependenciesAsync.html). Addressables downloads all the AssetBundles containing an asset with that label if not already available (along with any bundles containing the assets' dependencies).  
如果您有一组要预下载的资产，则可以为资产分配相同的标签，例如“preload”，并在调用 [Addressables.DownloadDependenciesAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.DownloadDependenciesAsync.html) 时使用该标签作为键。Addressables下载包含具有该标签的资产的所有资产包（如果尚不可用）（以及包含资产依赖项的任何捆绑包）。

### Progress 进度

An [AsyncOperationHandle](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.html) instance provides two ways to get progress:  
[AsyncOperationHandle](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.html) 实例提供两种获取进度的方法：

* [AsyncOperationHandle.PercentComplete](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.PercentComplete.html#UnityEngine_ResourceManagement_AsyncOperations_AsyncOperationHandle_PercentComplete): reports the percentage of sub-operations that have finished. For example, if an operation uses six sub-operations to perform its task, the PercentComplete indicates the entire operation is 50% complete when three of those operations have finished (it doesn't matter how much data each operation loads).  
  [AsyncOperationHandle.PercentComplete](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.PercentComplete.html#UnityEngine_ResourceManagement_AsyncOperations_AsyncOperationHandle_PercentComplete)：报告已完成的子操作的百分比。例如，如果某个操作使用六个子操作来执行其任务，则当其中三个操作完成时，PercentComplete 指示整个操作已完成 50%（每个操作加载多少数据无关紧要）。
* [AsyncOperationHandle.GetDownloadStatus](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.GetDownloadStatus.html#UnityEngine_ResourceManagement_AsyncOperations_AsyncOperationHandle_GetDownloadStatus): returns a [DownloadStatus](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.DownloadStatus.html) struct that reports the percentage in terms of total download size. For example, if an operation has six sub-operations, but the first operation represented 50% of the total download size, then GetDownloadStatus indicates the operation is 50% complete when the first operation finishes.  
  [AsyncOperationHandle.GetDownloadStatus](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.GetDownloadStatus.html#UnityEngine_ResourceManagement_AsyncOperations_AsyncOperationHandle_GetDownloadStatus)：返回一个 [DownloadStatus](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.DownloadStatus.html) 结构，该结构报告总下载大小的百分比。例如，如果某个操作有六个子操作，但第一个操作占总下载大小的 50%，则 GetDownloadStatus 指示第一个操作完成时该操作已完成 50%。

The following example illustrates how you could use [GetDownloadStatus](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.GetDownloadStatus.html#UnityEngine_ResourceManagement_AsyncOperations_AsyncOperationHandle_GetDownloadStatus) to check the status and dispatch progress events during the download:  
以下示例说明如何使用 [GetDownloadStatus](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.AsyncOperations.AsyncOperationHandle.GetDownloadStatus.html#UnityEngine_ResourceManagement_AsyncOperations_AsyncOperationHandle_GetDownloadStatus) 在下载过程中检查状态和调度进度事件：

using System.Collections;

using UnityEngine;

using UnityEngine.AddressableAssets;

using UnityEngine.Events;

using UnityEngine.ResourceManagement.AsyncOperations;

internal class PreloadWithProgress : MonoBehaviour

{

public string preloadLabel = "preload";

public UnityEvent<float> ProgressEvent;

public UnityEvent<bool> CompletionEvent;

private AsyncOperationHandle downloadHandle;

IEnumerator Start() {

downloadHandle = Addressables.DownloadDependenciesAsync(preloadLabel, false);

float progress = 0;

while (downloadHandle.Status == AsyncOperationStatus.None) {

float percentageComplete = downloadHandle.GetDownloadStatus().Percent;

if (percentageComplete > progress \* 1.1) // Report at most every 10% or so

{

progress = percentageComplete; // More accurate %

ProgressEvent.Invoke(progress);

}

yield return null;

}

CompletionEvent.Invoke(downloadHandle.Status == AsyncOperationStatus.Succeeded);

Addressables.Release(downloadHandle); //Release the operation handle

}

}

To discover how much data you need to download in order to load one or more assets, you can call [Addressables.GetDownloadSizeAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.GetDownloadSizeAsync.html):  
要了解需要下载多少数据才能加载一个或多个资产，您可以调用 [Addressables.GetDownloadSizeAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.GetDownloadSizeAsync.html)：

AsyncOperationHandle<long> getDownloadSize =

Addressables.GetDownloadSizeAsync(key);

The Result of the completed operation is the number of bytes that must be downloaded. If Addressables has already cached all the required AssetBundles, then Result is zero.  
AsyncOperationHandle<long>.Result是必须下载的字节数。如果可寻址对象已缓存所有必需的资产包，则结果为零。

Always release the download operation handle after you have read the Result object. If you don't need to access the results of the download operation, you can automatically release the handle by setting the autoReleaseHandle parameter to true, as shown in the following example:  
始终在读取 Result 对象后释放下载操作句柄。如果不需要访问下载操作的结果，可以通过将 autoReleaseHandle 参数设置为 true 来自动释放句柄，如以下示例所示：

using System.Collections;

using UnityEngine;

using UnityEngine.AddressableAssets;

internal class Preload : MonoBehaviour

{

public IEnumerator Start() {

yield return Addressables.DownloadDependenciesAsync("preload", true);

}

}

#### Clearing the dependency cache 清除依赖缓存

If you want to clear any AssetBundles cached by Addressables, call [Addressables.ClearDependencyCacheAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.ClearDependencyCacheAsync.html). This function clears the cached AssetBundles containing the assets identified by a key along with any bundles containing those assets' dependencies.  
如果要清除可寻址对象缓存的任何资产包，请调用 [Addressables.ClearDependencyCacheAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.ClearDependencyCacheAsync.html)。此函数清除包含键标识的资产的缓存资产包以及包含这些资产依赖项的任何捆绑包。

Note that ClearDependencyCacheAsync only clears assets bundles related to the specified key. If you updated the content catalog such that the key no longer exists or it no longer depends on the same AssetBundles, then these no-longer-referenced bundles remain in the cache until they expire (based on [cache settings](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Cache.html)).  
请注意，[Addressables.ClearDependencyCacheAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.ClearDependencyCacheAsync.html) 仅清除与指定键相关的资产包。如果您更新了内容目录，使密钥不再存在或不再依赖于相同的 AssetBundle，则这些不再引用的捆绑包将保留在缓存中，直到它们过期（基于 [cache settings](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Cache.html)）。

To clear all AssetBundles, you can use functions in the [UnityEngine.Caching](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Caching.html) class.  
要清除所有资源包，您可以使用 [UnityEngine.Caching](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Caching.html) 类中的函数。

## AssetBundle Loading 加载 AssetBundle

The Addressables system packs your assets in AssetBundles and loads these bundles "behind the scenes" as you load individual assets. You can control how AssetBundles load which are exposed on the BundledAssetGroupSchema class. You can set these options through the scripting API or under the Advanced options in the inspector of the AddressablesAssetGroup inspector.  
可寻址对象系统将您的资产打包在资产包中，并在您加载单个资产时“在后台”加载这些捆绑包。您可以控制如何加载在BundledAssetGroupSchema 类上公开的资产包。您可以通过脚本 API 或在AddressablesAssetGroup 检查器检查器中的“高级”选项下设置这些选项。

### UnityWebRequestForLocalBundles 通过UnityWebRequest加载本地资产包

Addressables can load AssetBundles via two engine APIs: UnityWebRequest.GetAssetBundle, and AssetBundle.LoadFromFileAsync. The default behavior is to use AssetBundle.LoadFromFileAsync when the AssetBundle is in local storage and use UnityWebRequest when the AssetBundle path is a URL.  
可寻址对象可以通过两个引擎 API 加载 AssetBundle：UnityWebRequest.GetAssetBundle 和 AssetBundle.LoadFromFileAsync。默认行为是当 AssetBundle 位于本地存储中时使用 AssetBundle.LoadFromFileAsync，当 AssetBundle 路径为 URL 时使用 UnityWebRequest 。

You can override this behavior to use UnityWebRequest for local Asset Bundles by setting BundledAssetGroupSchema.UseUnityWebRequestForLocalBundles to true. It can also be set through the BundledAssetGroupSchema GUI.  
您可以通过将BundledAssetGroupSchema.[UseUnityWebRequestForLocalBundles](#_Advanced_Options_高级选项)设置为 true 来覆盖此行为以将UnityWebRequest 用于本地资源包。它也可以通过 BundledAssetGroupSchema GUI 进行设置。

A few of these situations would include:  
下面列举了使用此功能的一些可能情况：

1. You are shipping local AssetBundles that use LZMA compression because you want your shipped game package to be as small as possible. In this case, you would want to use UnityWebRequest to recompress those AssetBundles LZ4 into the local disk cache.  
   您正在运送使用 LZMA 压缩的本地资产包，因为您希望发布的游戏包尽可能小。在这种情况下，您可能希望使用 UnityWebRequest 将这些 AssetBundles 用LZ4格式重新压缩到本地磁盘缓存中。
2. You are shipping an Android game and your APK contains AssetBundles that are compressed with the default APK compression.  
   您正在发布安卓游戏，并且您的 APK 包含使用默认 APK 压缩格式的资产包。
3. You want the entire local AssetBundle to be loaded into memory to avoid disk seeks. If you use UnityWebRequest and have caching disabled, the entire AssetBundle file will be loaded into the memory cache. This increases your runtime memory usage, but may improve loading performance as it eliminates disk seeking after the initial AssetBundle load. Both situations 1 and 2 above result in the AssetBundle existing on the player device twice (original and cached representations). This means the initial loads (decompressing and copying to cache) are slower than subsequent loads (loading from cache)  
   您希望将整个本地资产包加载到内存中以避免磁盘寻道。如果您使用 UnityWebRequest 并禁用了缓存，则整个 AssetBundle 文件将被加载到内存缓存中。这会增加运行时内存使用量，但可能会提高加载性能，因为它消除了初始 AssetBundle 加载后的磁盘查找。上述情况 1 和 2 都会导致资产包在播放器设备上存在两次（原始表示和缓存表示）。这意味着初始加载（解压缩和复制到缓存）比后续加载（从缓存加载）慢

### Handling Download Errors 处理下载错误

When a download fails, the RemoteProviderException contains errors that can be used to determine how to handle the failure. The RemoteProviderException is either the AsyncOperationHandle.OperationException or an inner exception. As shown below:  
当下载失败时，RemoteProviderException包含可用于确定如何处理失败的错误。RemoteProviderException 可能是AsyncOperationHandle.OperationException或AsyncOperationHandle.OperationException的InnerException（嵌套）。如下面例子所示：

using UnityEngine;

using UnityEngine.AddressableAssets;

using UnityEngine.ResourceManagement.AsyncOperations;

using UnityEngine.ResourceManagement.Exceptions;

internal class HandleDownloadError : MonoBehaviour

{

private AsyncOperationHandle m\_Handle;

void LoadAsset()

{

m\_Handle = Addressables.LoadAssetAsync<GameObject>("addressKey");

m\_Handle.Completed += handle =>

{

string dlError = GetDownloadError(m\_Handle);

if (!string.IsNullOrEmpty(dlError))

{

// handle what error

}

};

}

string GetDownloadError(AsyncOperationHandle fromHandle)

{

if (fromHandle.Status != AsyncOperationStatus.Failed)

return null;

RemoteProviderException remoteException;

System.Exception e = fromHandle.OperationException;

while (e != null)

{

remoteException = e as RemoteProviderException;

if (remoteException != null)

return remoteException.WebRequestResult.Error;

e = e.InnerException;

}

return null;

}

}

Possible error strings:

* "Request aborted"
* "Unable to write data"
* "Malformed URL"
* "Out of memory"
* "No Internet Connection"
* "Encountered invalid redirect (missing Location header?)"
* "Cannot modify request at this time"
* "Unsupported Protocol"
* "Destination host has an erroneous SSL certificate"
* "Unable to load SSL Cipher for verification"
* "SSL CA certificate error"
* "Unrecognized content-encoding"
* "Request already transmitted"
* "Invalid HTTP Method"
* "Header name contains invalid characters"
* "Header value contains invalid characters"
* "Cannot override system-specified headers"
* "Backend Initialization Error"
* "Cannot resolve proxy"
* "Cannot resolve destination host"
* "Cannot connect to destination host"
* "Access denied"
* "Generic/unknown HTTP error"
* "Unable to read data"
* "Request timeout"
* "Error during HTTP POST transmission"
* "Unable to complete SSL connection"
* "Redirect limit exceeded"
* "Received no data in response"
* "Destination host does not support SSL"
* "Failed to transmit data"
* "Failed to receive data"
* "Login failed"
* "SSL shutdown failed"
* "Redirect limit is invalid"
* "Not implemented"
* "Data Processing Error, see Download Handler error"
* "Unknown Error"

## Getting addresses at runtime 运行时获取address键值

By default, Addressables uses the address you assign to an asset as the [PrimaryKey](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceLocations.IResourceLocation.PrimaryKey.html#UnityEngine_ResourceManagement_ResourceLocations_IResourceLocation_PrimaryKey) value of its [IResourceLocation](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceLocations.IResourceLocation.html) instance. (If you disable the [**Include Addresses in Catalog**](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/GroupSettings.html#advanced-options) option of the Addressables group to which the asset belongs, the PrimaryKey could be a GUID, label, or an empty string.) If you want to get the address of an asset that you load with an AssetReference or label, you can first load the asset's locations, as described in [Loading Assets by Location](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadingAddressableAssets.html#loading-assets-by-location). You can then use the IResourceLocation instance to both access the PrimaryKey value and to load the asset.  
默认情况下，可寻址对象使用您分配给资产的address作为其[IResourceLocation](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceLocations.IResourceLocation.html)实例的 [PrimaryKey](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.ResourceLocations.IResourceLocation.PrimaryKey.html#UnityEngine_ResourceManagement_ResourceLocations_IResourceLocation_PrimaryKey) 。（如果禁用资产所属的可寻址对象组的[**Include Addresses in Catalog**](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/GroupSettings.html#advanced-options)选项，则PrimaryKey可能是 GUID、label或空字符串。如果要获取使用AssetReference或label加载的资产的地址，可以先加载资产的位置，如 [Loading Assets by Location](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadingAddressableAssets.html#loading-assets-by-location)中所述。然后，您可以使用 IResourceLocation 实例来访问 PrimaryKey 值并加载资产。

The following example gets the address of the asset assigned to an [AssetReference](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.AssetReference.html) object named MyRef1:  
下面的示例获取分配给名为MyRef1的[AssetReference](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.AssetReference.html) 对象的资产的地址：

var opHandle = Addressables.LoadResourceLocationsAsync(MyRef1);

yield return opHandle;

if (opHandle.Status == AsyncOperationStatus.Succeeded &&

opHandle.Result != null &&

opHandle.Result.Count > 0)

{

Debug.Log("address is: " + opHandle.Result[0].PrimaryKey);

}

Labels often refer to multiple assets. The following example illustrates how to load multiple Prefab assets and use their primary key value to add them to a dictionary:  
Labels通常指多个资产。以下示例说明了如何加载多个预制件资产并使用其主键值将它们添加到字典中：

Dictionary<string, GameObject> \_preloadedObjects

= new Dictionary<string, GameObject>();

private IEnumerator PreloadHazards() {

//find all the locations with label "SpaceHazards"

var loadResourceLocationsHandle

= Addressables.LoadResourceLocationsAsync("SpaceHazards", typeof(GameObject));

if (!loadResourceLocationsHandle.IsDone)

yield return loadResourceLocationsHandle;

//start each location loading

List<AsyncOperationHandle> opList = new List<AsyncOperationHandle>();

foreach (IResourceLocation location in loadResourceLocationsHandle.Result) {

AsyncOperationHandle<GameObject> loadAssetHandle

= Addressables.LoadAssetAsync<GameObject>(location);

loadAssetHandle.Completed +=

obj => { \_preloadedObjects.Add(location.PrimaryKey, obj.Result); };

opList.Add(loadAssetHandle);

}

//create a GroupOperation to wait on all the above loads at once.

var groupOp = Addressables.ResourceManager.CreateGenericGroupOperation(opList);

if (!groupOp.IsDone)

yield return groupOp;

Addressables.Release(loadResourceLocationsHandle);

//take a gander at our results.

foreach (var item in \_preloadedObjects) {

Debug.Log(item.Key + " - " + item.Value.name);

}

}

## Modification events Modification 事件

Modification events are used to signal to parts of the Addressables system when certain data is manipulated, such as an AddressableAssetGroup or an AddressableAssetEntry getting added or removed.  
Modification事件用于在操作某些数据（例如添加或删除 AddressableAssetGroup 或 AddressableAssetEntry ）时向可寻址系统的各个部分发出信号。

Modification events are triggered as part of SetDirty calls inside of Addressables. SetDirty is used to indicate when an asset needs to be re-serialized by the AssetDatabase. As part of this SetDirty, two modification event callbacks can trigger:  
Modification事件作为可寻址对象内部 SetDirty 调用的一部分触发。SetDirty 用于指示资产何时需要由资产数据库重新序列化。作为此 SetDirty 的一部分，可以触发两个修改事件回调：

* public static event Action<AddressableAssetSettings, ModificationEvent, object> OnModificationGlobal
* public Action<AddressableAssetSettings, ModificationEvent, object> OnModification { get; set; }

which can be found on [AddressableAssetSettings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.html) through a static, or instance, accessors respectively.

#### Code Samples

AddressableAssetSettings.OnModificationGlobal += (settings, modificationEvent, data) =>

{

if(modificationEvent == AddressableAssetSettings.ModificationEvent.EntryAdded)

{

//Do work

}

};

AddressableAssetSettingsDefaultObject.Settings.OnModification += (settings, modificationEvent, data) =>

{

if (modificationEvent == AddressableAssetSettings.ModificationEvent.EntryAdded)

{

//Do work

}

};

Modification events pass in a generic object for the data associated with the event. Below is a list of the modification events and the data types that are passed with them.  
Modification事件传入与事件关联的数据的通用object 对象。下面是Modification事件以及随它们传递的数据类型的列表。

#### The Data Passed with Each ModificationEvent:

* GroupAdded The data passed with this event is the [AddressableAssetGroup](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetGroup.html), or list of groups, that were added.
* GroupRemoved The data passed with this event is the [AddressableAssetGroup](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetGroup.html), or list of groups, that were removed.
* GroupRenamed The data passed with this event is the [AddressableAssetGroup](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetGroup.html), or list of groups, that were renamed.
* GroupSchemaAdded The data passed with this event is the [AddressableAssetGroup](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetGroup.html), or list of groups, that had schemas added to them.
* GroupSchemaRemoved The data passed with this event is the [AddressableAssetGroup](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetGroup.html), or list of groups, that had schemas removed from them.
* GroupSchemaModified The data passed with this event is the [AddressableAssetGroupSchema](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetGroupSchema.html) that was modified.
* GroupTemplateAdded The data passed with this event is the ScriptableObject, typically one that implements [IGroupTemplate](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.IGroupTemplate.html), that was the added Group Template object.
* GroupTemplateRemoved The data passed with this event is the ScriptableObject, typically one that implements [IGroupTemplate](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.IGroupTemplate.html), that was the removed Group Template object.
* GroupTemplateSchemaAdded The data passed with this event is the [AddressableAssetGroupTemplate](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetGroupTemplate.html) that had a schema added.
* GroupTemplateSchemaRemoved The data passed with this event is the [AddressableAssetGroupTemplate](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetGroupTemplate.html) that had a schema removed.
* EntryCreated The data passed with this event is the [AddressableAssetEntry](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetEntry.html) that was created.
* EntryAdded The data passed with this event is the [AddressableAssetEntry](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetEntry.html), or list of entries, that were added.
* EntryMoved The data passed with this event is the [AddressableAssetEntry](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetEntry.html), or list of entries, that were moved from one group to another.
* EntryRemoved The data passed with this event is the [AddressableAssetEntry](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetEntry.html), or list of entries, that were removed.
* LabelAdded The data passed with this event is the string label that was added.
* LabelRemoved The data passed with this event is the string label that was removed.
* ProfileAdded The data passed with this event is [BuildProfile] that was added.
* ProfileRemoved The data passed with this event is the string of the profile ID that was removed.
* ProfileModified The data passed with this event is [BuildProfile] that was modified, or null if a batch of BuildProfiles were modified.
* ActiveProfileSet The data passed with this event if the string of the profile ID that is set as the active profile.
* EntryModified The data passed with this event is the [AddressableAssetEntry](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetEntry.html), or list of entries, that were modified.
* BuildSettingsChanged The data passed with this event is the [AddressableAssetBuildSettings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetBuildSettings.html) object that was modified.
* ActiveBuildScriptChanged The data passed with this event is the [IDataBuilder](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.IDataBuilder.html) build script that was set as the active builder.
* DataBuilderAdded The data passed with this event is the ScriptableObject, typically one that implements [IDataBuilder](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.IDataBuilder.html), that was added to the list of DataBuilders.
* DataBuilderRemoved The data passed with this event is the ScriptableObject, typically one that implements [IDataBuilder](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.IDataBuilder.html), that was removed from the list of DataBuilders.
* InitializationObjectAdded The data passed with this event is the ScriptableObject, typically one that implements [IObjectInitializationDataProvider](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.Util.IObjectInitializationDataProvider.html), that was added to the list of InitializationObjects.
* InitializationObjectRemoved The data passed with this event is the ScriptableObject, typically one that implements [IObjectInitializationDataProvider](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.ResourceManagement.Util.IObjectInitializationDataProvider.html), that was removed from the list of InitializationObjects.
* ActivePlayModeScriptChanged The data passed with this event is the [IDataBuilder](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.IDataBuilder.html) that was set as the new active play mode data builder.
* BatchModification The data passed with this event is null. This event is primarily used to indicate several modification events happening at the same time and the [AddressableAssetSettings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.html) object needed to be marked dirty.
* HostingServicesManagerModified The data passed is either going to be the [HostingServicesManager](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.HostingServices.HostingServicesManager.html), or the [HttpHostingService](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.HostingServices.HttpHostingService.html) that were modified.
* GroupMoved The data passed with this event is the full list of [AddressableAssetGroups](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetGroup.html).
* CertificateHandlerChanged The data passed with this event is the new System.Type of the Certificate Handler to be used.

# Diagnostic tools 诊断工具

The Addressables packages provides the following tools for analyzing your Addressables setup, performance, and build results:  
可寻址包提供以下工具，用于分析可寻址设置、性能和生成结果：

* [Analyze tool](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AnalyzeTool.html): provides a set of utilities and reports that you can use to find and fix asset duplication and to see how the system packs the assets in your groups into bundles. The reports and utilities include the following (you can also create your own analyze rule classes to produce additional reports):  
  [Analyze tool](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AnalyzeTool.html)：提供一组实用程序和报告，可用于查找和修复资产重复，以及查看系统如何将组中的资产打包到捆绑包中。报告和实用程序包括以下内容（您还可以创建自己的分析规则类以生成其他报告）：
  + Check Duplicate Bundle Dependencies
  + Check Resources to Addressable Duplicate Dependencies
  + Check Scene to Addressable Duplicate Dependencies
  + Bundle Layout Preview
* [Event viewer](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/EventViewer.html): provides a runtime profile view of your Addressable operations, including asset loading and unloading.  
  [Event viewer](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/EventViewer.html)：提供可寻址操作的运行时profile视图，包括资产加载和卸载。
* [Build layout report](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildLayoutReport.html): describes how Addressables packs the assets in your groups into AssetBundles.  
  [Build layout report](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildLayoutReport.html): 描述AssetGroup中的资产如何打包到资产包中.
* [Build profile log](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildProfileLog.html): provides a build-time profiling file that you can view in a Chromium-based browser.  
  [Build profile log](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildProfileLog.html): 提供可在基于 Chromium 的浏览器中查看的构建时分析文件。

## Event Viewer 事件查看器

Use the **Addressables Event Viewer** window to monitor memory management of your Addressables assets. The window shows when your application loads and unloads assets and displays the reference counts of all Addressables system operations. The window also shows approximate views of the application frame rate and the total amount of allocated managed memory. You can use these charts to detect how Addressables events such as loading and releasing assets affect application performance and to detect assets that you never release.  
使用**Addressables Event Viewer**窗口可以监视可寻址资产的内存管理。该窗口显示应用程序何时加载和卸载资产，并显示所有可寻址系统操作的引用计数。该窗口还显示应用程序帧速率和分配的托管内存总量的近似视图。您可以使用这些图表来检测可寻址对象事件（如加载和释放资产）如何影响应用程序性能，并检测您从未释放的资产。

Use the **Use Existing Build** Play Mode script to get the most accurate information in the Event Viewer in Play mode. The **Use Asset Database** script doesn't account for any shared dependencies among the Assets and the **Simulate Groups** script gives a less accurate monitoring of reference counts.  
使用**Use Existing Build**脚本在运行模式下的事件查看器中获取最准确的信息。**Use Asset Database**脚本不考虑资产之间的任何共享依赖项， **Simulate Groups** 脚本对引用计数的监视不太准确。

**IMPORTANT**

To view data in the Event Viewer, you must enable the **Send Profiler Events** setting in your [AddressableAssetSettings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.html) object's Inspector and make a new content build.  
若要在事件查看器中查看数据，必须在[AddressableAssetSettings](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.html)对象的检查器中启用**Send Profiler Events**设置，然后重新Build。

See [Memory management](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/MemoryManagement.html) for more information about managing the memory associated with your Addressable assets.

### Viewing Addressables events 查看可寻址对象事件

View Addressables asset lifespan charts and events in the **Event Viewer** window:  
在 **Event Viewer**窗口中查看可寻址资产生命周期图表和事件：

1. Enable **Send Profiler Events** in your Addressables settings: a. Open your Addressable settings Inspector (menu: **Window > Asset Management > Addressables > Settings**) b. Under **Diagnostics**, check the **Send Profiler Events** option.
2. Rebuild your Addressables content using the **Default Build Script** from the Addressables Groups window.
3. Open the **Event Viewer** (menu: **Window > Asset Management > Addressables > Event Viewer**).
4. Enter Play mode in the Editor.

### Viewing Addressables events in a standalone player 在独立播放器中查看可寻址对象事件

To connect the Event Viewer to a standalone player, follow the steps under [Viewing Addressables Events](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/EventViewer.html#viewing-addressables-events) except for the last step where you enter Play mode in the Editor. Instead, perform the following, additional steps:  
要将事件查看器连接到独立播放器（发布的程序），请按照 [Viewing Addressables Events](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/EventViewer.html#viewing-addressables-events)下的步骤操作，但在编辑器中进入播放模式的最后一步除外。相反，请执行以下附加步骤：

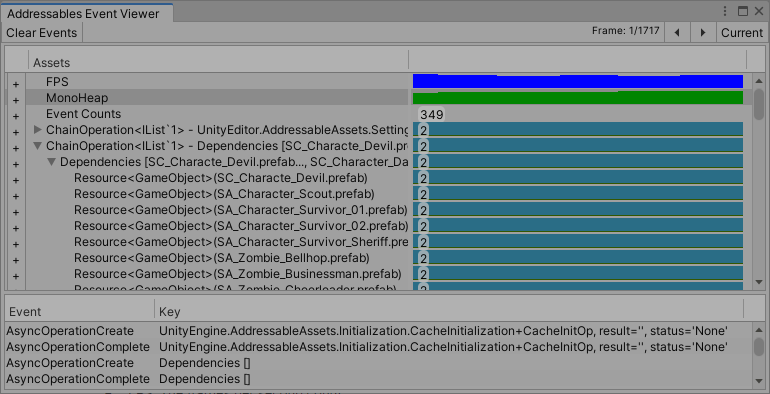
1. Open the **Build Settings** window (menu: **File > Build Settings**).
2. Check the **Development Build** option.
3. Check the **Autoconnect Profiler** option.
4. Open the Unity **Profiler window (menu: Window > Analysis > Profiler**).
5. On the Build Settings window, click **Build and Run**.

The Event Viewer automatically connects to your standalone player and displays the Addressables events that occur.  
事件查看器会自动连接到独立播放器，并显示发生的可寻址事件。

See [Profiler overview](https://docs.unity3d.com/2019.4/Documentation/Manual/Profiler.html) for more information about using the Unity Profiler.

### Event Viewer window 事件查看器窗口

To access the window in the Editor, select **Window** > **Asset Management** > **Addressables** > **Event Viewer**.



*The Event Viewer window*

The window has three sections:

* Toolbar:
  + **Clear Events** button: clears all recorded frames, erasing everything in the window.
  + **Unhide All Hidden Events**: returns any asset or operation lifelines that you have hidden to their normal, displayed state. Only shown when you have hidden events.
  + **Frame** counter: displays the position of the frame cursor and the number of recorded frames. (If the toolbar doesn't display the **Frame** counter, the frame cursor is at the current frame.)
  + Frame step (**<>**) buttons: steps the frame cursor through recorded frames. (You can also use the keyboard arrow keys.)
  + **Current** button: moves the frame cursor to the current frame.
* **Assets** display: shows profiler and lifespan charts related to Addressables operations and assets.
  + **FPS** chart: the application frame rate.
  + **MonoHeap** chart: the amount of managed memory in use.
  + **Event Counts**: the number of Addressables events that occurred in a frame (view the events in the **Event** list).
  + **Instantiation Counts**: the number of calls to [Addressables.InstantiateAsync](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.InstantiateAsync.html) in a frame.
  + Operation and asset lifespan charts: show when the system loads and releases operations or assets and display the reference counts. See [Asset lifespan chart](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/EventViewer.html#asset-lifespan-chart).
* **Event** list: shows the Addressable operation events that occurred in the frame.

You can click in the chart area of the window to pause your application and set the position of the **frame cursor**. The frame cursor appears in the window as a vertical line and shows summary information for the selected frame.

**NOTE**

The FPS and MonoHeap charts include all factors that affect performance, not just those related to Addressable assets.

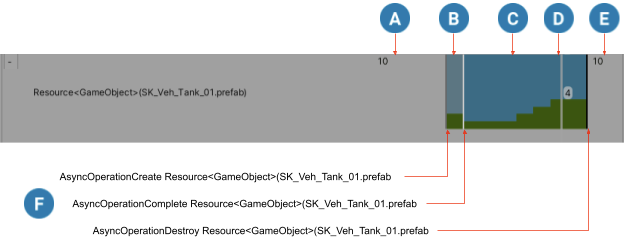
### Asset lifespan chart 资源生命周期图表

The asset lifespan chart shows when an asset or operation is created, its current reference count, and when it is destroyed.

To expand the display line showing the lifespan of an asset or operation, click the **+** button for the entry in the **Assets** list.

To view the sub-operations or subassets of an entry, click the expand icon (**>**) of the parent object.

You can remove the lifeline of an asset or operation from the chart by right-clicking it and choosing **Hide Selected Events** on the context menu. Click the **Unhide All Hidden Events** button on the toolbar to reveal the lifelines of any hidden events.

  
*An example asset lifespan and the related Addressables events*

A lifespan chart shows the following information:

  
Before the asset was loaded (no display).

  
The asset load is in progress (dull blue).

  
The asset is loaded (blue). The green bar shows the current reference count of the asset.

  
The Event Viewer frame cursor, which shows information for the selected frame, in this case, the cursor indicates that the reference count of this asset is 4 during the selected frame.

  
After the asset is destroyed and unloaded (no display).

  
The events associated with specific frames (events are only shown at the position of the frame cursor).

## Analyze tool 分析工具

Analyze is a tool that gathers information on your Projects' Addressables layout. In some cases, Analyze may take appropriate actions to clean up the state of your Project. In others, Analyze is purely an informational tool that allows you to make more informed decisions about your Addressables layout.

### Using Analyze 使用分析器

In the Editor, open the **Addressables Analyze** window (**Window** > **Asset Management** > **Addressables** > **Analyze**), or open it via the **Addressables Groups** window by clicking the **Tools** > **Window** > **Analyze** button.

The Analyze window displays a list of Analyze rules, along with the following operations:

* Analyze Selected Rules
* Clear Selected Rules
* Fix Selected Rules

#### The analyze operation 分析操作

The analyze operation gathers the information needed by the rule. Run this action on a rule or set of rules to gather data about the build, dependency maps, and more. Each rule must gather any required data and report it back as a list of [AnalyzeResult](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.AnalyzeRules.AnalyzeRule.AnalyzeResult.html) objects.  
分析操作收集规则所需的信息。对一个规则或一组规则运行此操作，以收集有关生成、依赖项映射等数据。每个规则都必须收集任何必需的数据，并将其报告为 [AnalyzeResult](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.AnalyzeRules.AnalyzeRule.AnalyzeResult.html) 对象列表。

No action should be taken to modify any data or the state of the Project during the analyze step. Based on the data gathered in this step, the [fix](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AnalyzeTool.html#the-fix-operation) operation may be the appropriate course of action. Some rules, however, only contain an analyze step, as no reasonably appropriate and universal action can be taken based on the information gathered. [Check Scene to Addressable Duplicate Dependencies](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AnalyzeTool.html#check-scene-to-addressable-duplicate-dependencies) and [Check Resources to Addressable Duplicate Dependencies](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AnalyzeTool.html#check-resources-to-addressable-duplicate-dependencies) are examples of such rules.  
在分析步骤中，不应执行任何操作来修改任何数据或项目状态。根据此步骤中收集的数据， [fix](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AnalyzeTool.html" \l "the-fix-operation) 操作会进行适当的修复。然而，有些规则只包含一个分析步骤，因为无法根据收集到的信息采取合理适当和普遍的行动。[Check Scene to Addressable Duplicate Dependencies](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AnalyzeTool.html#check-scene-to-addressable-duplicate-dependencies)和[Check Resources to Addressable Duplicate Dependencies](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AnalyzeTool.html#check-resources-to-addressable-duplicate-dependencies)是此类规则的示例。

Rules that are purely informational and contain no fix operation are categorized as **Unfixable Rules**. Those that do have a fix operation are categorized as **Fixable Rules**.  
纯信息性且不包含修复操作的规则被归类为**Unfixable Rules**。具有修复操作的那些被归类为**Fixable Rules**。

#### The clear step 清理分析结果

The clear operation removes any data gathered by the analysis and updates the TreeView accordingly.

#### The fix operation 修复操作

For **Fixable Rules**, you may choose to run the fix operation. The fix operation uses data gathered during the analyze step to perform any necessary modifications and resolve the issues.  
对于**Fixable Rules**，您可以选择运行修复操作。修复操作根据分析步骤期间收集的数据来执行任何必要的修改并解决问题。

The provided [Check Duplicate Bundle Dependencies](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AnalyzeTool.html#check-duplicate-bundle-dependencies) rule is an example of a fixable rule. Problems detected by this rule's analysis can be fixed because there is a reasonably appropriate action that can be taken to resolve them.  
提供的 [Check Duplicate Bundle Dependencies](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AnalyzeTool.html#check-duplicate-bundle-dependencies)规则是可修复规则的一个示例。可以修复此规则检测到的问题，因为可以采取合理适当的操作来解决它们。

### Provided Analyze rules 提供的分析规则

#### Fixable rules 可修复规则

##### Check Duplicate Bundle Dependencies

This rule checks for potentially duplicated assets, by scanning all groups with [BundledAssetGroupSchemas](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.GroupSchemas.BundledAssetGroupSchema.html) and projecting the asset group layout. This essentially requires triggering a full build, so this check is time-consuming and performance-intensive.  
此规则通过使用 [BundledAssetGroupSchemas](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Settings.GroupSchemas.BundledAssetGroupSchema.html)扫描所有组并投影资产组布局来检查可能重复的资产。这基本上需要触发完整构建，因此此检查非常耗时且性能密集型。

**Issues**: Duplicated assets result from assets in different groups sharing dependencies, for example two Prefabs that share a material existing in different Addressable groups. That material (and any of its dependencies) would be pulled into both groups containing the Prefabs. To prevent this, the material must be marked as Addressable, either with one of the Prefabs, or in its own space, thereby putting the material and its dependencies in a separate Addressable group.  
**问题：**重复的资源是由于不同组中的资源共享依赖项，例如，不同可寻址组中的两个预制件共享同一个材质时，该材质（及其任何依赖项）将被拉入包含预制件的两个组中。为防止出现这种情况，必须将材质标记为可寻址，而且这个材质要么在其中一个预制件的组里，要么放在材质自己的组里。

**Resolution**: If this check discovers any issues, run the fix operation on this rule to create a new Addressable group to which to move all dependent assets.  
**解决方法：**如果此检查发现任何问题，请对此规则运行修复操作，以创建新的可寻址组，将所有重复的依赖资产移动到该组。

**Exceptions**: If you have an asset containing multiple objects, it is possible for different groups to only pull in portions of the asset, and not actually duplicate. An FBX with many meshes is an example of this. If one mesh is in "GroupA" and another is in "GroupB", this rule will think that the FBX is shared, and extract it into its own group if you run the fix operation. In this edge case, running the fix operation is actually harmful, as neither group would have the full FBX asset.  
**例外情况：**如果您的资产包含多个对象，则不同的组可能只拉入资产的一部分，而实际上不会重复。具有许多Mesh的 FBX 就是一个例子。如果一个mesh位于“组 A”中，另一个mesh位于“组 B”中，则此规则将认为 FBX 已共享，并在运行修复操作时将其提取到自己的组中。在这种边缘情况下，运行修复操作实际上是有害的，因为两个组都不会拥有完整的 FBX 资产。

Also note that duplicate assets may not always be an issue. If assets will never be requested by the same set of users (for example, region-specific assets), then duplicate dependencies may be desired, or at least be inconsequential. Each Project is unique, so fixing duplicate asset dependencies should be evaluated on a case-by-case basis.  
另请注意，重复资产可能并不总是一个问题。如果同一组用户永远不会请求资产（例如，特定于区域的资产），则可能需要重复的依赖项，或者至少无关紧要。每个项目都是唯一的，因此修复重复的资产依赖关系应根据具体情况进行评估。

#### Unfixable rules 无法修复的规则

##### Check Resources to Addressable Duplicate Dependencies

This rule detects if any assets or asset dependencies are duplicated between built Addressable data and assets residing in a Resources folder.  
此规则检测在构建的可寻址数据和驻留在 Resources 文件夹中的资产之间是否复制了任何资产或资产依赖项。

**Issues**: These duplicates mean that data will be included in both the application build and the Addressables build.  
**问题：**这些重复项意味着数据将同时包含在应用程序生成和可寻址对象生成中。

**Resolution**: This rule is unfixable, because no appropriate action exists. It is purely informational, alerting you to the redundancy. You must decide how to proceed and what action to take, if any. One example of a possible manual fix is to move the offending asset(s) out of the Resources folder, and make them Addressable.  
**解决方法：**此规则不可修复，因为不存在适当的操作。它纯粹是信息性的，提醒您冗余。您必须决定如何继续以及要采取什么操作（如果有）。可能的手动修复的一个示例是将有问题的资产移出 Resources 文件夹，并使其可寻址。

##### Check Scene to Addressable Duplicate Dependencies

This rule detects any assets or asset dependencies that are shared between the Scenes in the Editor Scene list and Addressables.  
此规则检测编辑器场景列表中的场景与可寻址对象之间共享的任何资产或资产依赖关系。

**Issues**: These duplicates mean that data will be included in both the application build and the Addressables build.  
**问题：**这些重复项意味着数据将同时包含在应用程序生成和可寻址对象生成中。

**Resolution**: It is purely informational, alerting you to the redundancy. You must decide how to proceed and what action to take, if any. One example of a possible manual fix is to pull the built-in Scene(s) with duplicated references out of Build Settings and make it an Addressable Scene.  
**解决方法：**它纯粹是信息性的，提醒您冗余。您必须决定如何继续以及要采取什么操作（如果有）。可能的手动修复的一个示例是将具有重复引用的内置场景从构建设置中提取出来，并使其成为可寻址场景。

##### Bundle Layout Preview

This rule will show how assets explicitly marked as Addressable will be laid out in the Addressable build. Given these explicit assets, we also show what assets are implicitly referenced by, and therefore will be pulled into, the build.  
此规则将显示显式标记为可寻址的资产在可寻址版本中的布局方式。给定这些显式资产，我们还显示哪些资产被隐式引用，因此将被拉入构建。

Data gathered by this rule does not indicate any particular issues. It is purely informational.  
此规则收集的数据并不表示任何特定问题。它纯粹是信息性的。

### Extending Analyze 扩展分析工具

Each unique Project may require additional Analyze rules beyond what comes pre-packaged. The Addressable Assets System allows you to create your own custom rule classes.

See the [Custom analyze rule project](https://github.com/Unity-Technologies/Addressables-Sample/tree/master/Advanced/CustomAnalyzeRule) in the [Addressables-Sample](https://github.com/Unity-Technologies/Addressables-Sample) repository for an example.

#### AnalyzeRule objects

Create a new child class of the [AnalyzeRule](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.AnalyzeRules.AnalyzeRule.html) class, overriding the following properties:

* [CanFix](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.AnalyzeRules.AnalyzeRule.CanFix.html#UnityEditor_AddressableAssets_Build_AnalyzeRules_AnalyzeRule_CanFix) tells Analyze if the rule is deemed fixable or not.
* [ruleName](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.AnalyzeRules.AnalyzeRule.ruleName.html#UnityEditor_AddressableAssets_Build_AnalyzeRules_AnalyzeRule_ruleName) is the display name you'll see for this rule in the **Analyze window**.

You'll also need to override the following methods, which are detailed below:

* [List<AnalyzeResult> RefreshAnalysis(AddressableAssetSettings settings)](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.AnalyzeRules.AnalyzeRule.RefreshAnalysis.html)
* [void FixIssues(AddressableAssetSettings settings)](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.AnalyzeRules.AnalyzeRule.FixIssues.html)
* [void ClearAnalysis()](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.AnalyzeRules.AnalyzeRule.ClearAnalysis.html#UnityEditor_AddressableAssets_Build_AnalyzeRules_AnalyzeRule_ClearAnalysis)

##### TIP

If your rule is designated unfixable, you don't have to override the FixIssues method.

##### RefreshAnalysis

This is your analyze operation. In this method, perform any calculations you'd like and cache any data you might need for a potential fix. The return value is a List<AnalyzeResult> list. After you'd gathered your data, create a new [AnalyzeResult](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.AnalyzeRules.AnalyzeRule.AnalyzeResult.html) for each entry in your analysis, containing the data as a string for the first parameter and a [MessageType](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.AnalyzeRules.AnalyzeRule.AnalyzeResult.severity.html#UnityEditor_AddressableAssets_Build_AnalyzeRules_AnalyzeRule_AnalyzeResult_severity) for the second (to optionally designate the message type as a warning or error). Return the list of objects you create.

If you need to make child elements in the TreeView for a particular [AnalyzeResult](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.AnalyzeRules.AnalyzeRule.AnalyzeResult.html) object, you can delineate the parent item and any children with [kDelimiter](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.AnalyzeRules.AnalyzeRule.kDelimiter.html). Include the delimiter between the parent and child items.

##### FixIssues

This is your fix operation. If there is an appropriate action to take in response to the analyze step, execute it here.

##### ClearAnalysis

This is your clear operation. Any data you cached in the analyze step can be cleaned or removed in this function. The TreeView will update to reflect the lack of data.

#### Adding custom rules to the GUI

A custom rule must register itself with the GUI class using AnalyzeSystem.RegisterNewRule<RuleType>(), in order to show up in the **Analyze** window. For example:

using UnityEditor;

using UnityEditor.AddressableAssets.Build;

using UnityEditor.AddressableAssets.Build.AnalyzeRules;

class MyRule : AnalyzeRule

{

// Rule code...

}

// Register rule

[InitializeOnLoad]

class RegisterMyRule

{

static RegisterMyRule() {

AnalyzeSystem.RegisterNewRule<MyRule>();

}

}

##### AnalyzeRule classes

In order to make it faster to setup custom rules, Addressables includes the following classes, which inherit from [AnalyzeRule](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.AnalyzeRules.AnalyzeRule.html):

* [BundleRuleBase](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.AnalyzeRules.BundleRuleBase.html) is a base class for handling [AnalyzeRule](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.AnalyzeRules.AnalyzeRule.html) tasks. It includes some basic methods to retrieve information about bundle and resource dependencies.
* **Check bundle duplicates** base classes help check for bundle dependency duplicates. Override the [FixIssues](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.AnalyzeRules.CheckBundleDupeDependencies.FixIssues.html) method implementation to perform some custom action.
  + [CheckBundleDupeDependencies](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.AnalyzeRules.CheckBundleDupeDependencies.html) inherits from [BundleRuleBase](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.AnalyzeRules.BundleRuleBase.html) and includes further methods for [AnalyzeRule](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.AnalyzeRules.AnalyzeRule.html) to check bundle dependencies for duplicates and a method to attempt to resolve these duplicates.
  + [CheckResourcesDupeDependencies](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.AnalyzeRules.CheckResourcesDupeDependencies.html) is the same, but resource dependencies specific.
  + [CheckSceneDupeDependencies](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEditor.AddressableAssets.Build.AnalyzeRules.CheckSceneDupeDependencies.html) is the same, but for scene dependencies specific.

## Build layout report

The build layout report provides detailed information and statistics about your Addressables builds, including:

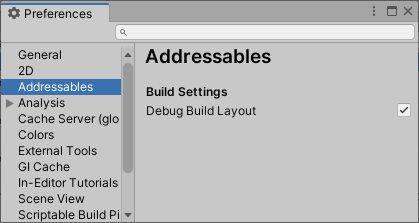
* Description of AssetBundles
* Sizes of each Asset and AssetBundle
* Explanation of non-Addressable Assets implicitly included in AssetBundles as dependencies
* AssetBundle dependencies

When enabled, the Addressables build script creates the report whenever you build Addressables content. You can enable the report in the Addressables section of the [Preferences window](https://docs.unity3d.com/Manual/Preferences.html). You can find the report in your project folder at Library/com.unity.addressables/buildlayout.txt. Producing the report does increase build time.

See [Building your Addressable content](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildingContent.html) for more information about building content.

### Creating a build report

To create a build report:

1. Enable the build report.
   1. Open the Unity Preferences window (menu: Edit > Preferences).
   2. Select **Addressables** from the list of preference types.
   3. Check the **Debug Build Layout** option. 
2. Perform a full build of your Addressables content. (See [Builds](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Builds.html) for more information.)
3. In a file system window, navigate to the Library/com.unity.addressables/ folder of your Unity project.
4. Open the buildlayout.txt file in a suitable text editor.

### Report data

A build layout report contains the following information:

* [Summary](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildLayoutReport.html#summary-section): provides an overview of the build
* [Group](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildLayoutReport.html#group-section): provides information for each group
* [Asset bundle](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildLayoutReport.html#assetbundle-information): provides information about each bundle built for a group
* [Asset](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildLayoutReport.html#asset-information): provides information about each explicit asset in a bundle
* [File](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildLayoutReport.html#file-information): provides information about each serialized file in an AssetBundle archive
* [Built-in bundles](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildLayoutReport.html#built-in-bundles): provides information about bundles created for assets, such as the default shader, that are built into Unity

#### Summary section

Provides a summary of the build.

| **Name** | **Purpose** |
| --- | --- |
| Addressable Groups | The number of groups included in the build. |
| Explicit Assets Addressed | The number of Addressable assets in the build (this number doesn't include assets in the build that are referenced by an Addressable asset, but which aren't marked as Addressable). |
| Total Bundle | The number of AssetBundles created by the build, including how many contain Scene data. |
| Total Build Size | The combined size of all AssetBundles. |
| Total MonoScript Size | The size of serialized MonoBehaviour and SerializedObject instances. |
| Total AssetBundle Object Size |  |

#### Group section

Reports how Addressables packed the assets in a group into AssetBundles.

| **Name** | **Purpose** |
| --- | --- |
| Group summary | Name, number of bundles created for group, total size, and number of explicit assets built for the group. |
| Schemas | Schemas and settings for the group. |
| Asset bundles | See [AssetBundle information](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildLayoutReport.html#assetbundle-information). |

#### AssetBundle information

Reports details for each AssetBundle built for a group.

| **Name** | **Purpose** |
| --- | --- |
| File name | The file name of the AssetBundle. |
| Size |  |
| Compression | The compression setting used for the bundle. |
| Object size |  |
| Bundle Dependencies | The list of other AssetBundles the current bundle depends upon. These bundles are always loaded with the current bundle. |
| Expanded Bundle Dependencies |  |
| Explicit Assets | [Asset information](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildLayoutReport.html#asset-information) about Addressables included in the bundle. |
| Files | [File information](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/BuildLayoutReport.html#file-information) about the files in the AssetBundle archive. Scene bundles contain up to two files per Scene, non-Scene bundles contain only one file. |
|  |  |

#### Asset information

Provides Information for each asset in the Explicit Assets section.

| **Name** | **Purpose** |
| --- | --- |
| Asset path | The path to the asset in your project |
| Total Size |  |
| Size from Objects |  |
| Size from Streamed Data |  |
| File Index | The index of the file in the AssetBundle in which this asset is located. |
| Addressable Name | The address of the asset. |
| External References |  |
| Internal References |  |

#### File information

Provides details about each serialized file in an AssetBundle archive

| **Name** | **Purpose** |
| --- | --- |
| File summary | Index in file list, number and size of serialized MonoScripts in the file |
| File sections | A serialized file can have one or more of the following sections: |

No extension -- .resS -- .resource -- .sharedAssets -- | | Data from Other Assets| Dependent assets referenced by assets in the file. |

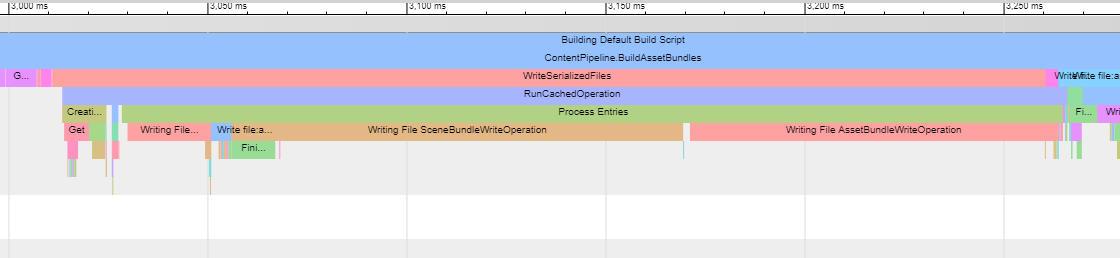
#### Built-in Bundles

Lists any bundles that Addressables created from assets, such as the default shaders, that are provided as part of the Unity Engine. The Addressables build places such assets in the separate bundles listed here when needed to avoid duplicating the assets across multiple bundles as implicit dependencies.

## Build Profiling

The Addressables build process always creates a .json log file that contains build performance information. You can find the log file in your project folder at Library/com.unity.addressables/AddressablesBuildTEP.json.

View the log file with the chrome://tracing tool in Google Chrome or another [Chromium](https://www.chromium.org/Home)-based browser.



*A sample log file displayed in chrome://tracing*

**To view the build profile:**

1. Open a [Chromium](https://www.chromium.org/Home)-based browser.
2. Enter [chrome://tracing](chrome://tracing/) in the browser to open the [Trace Event Profiling Tool](https://www.chromium.org/developers/how-tos/trace-event-profiling-tool).
3. Click the **Load** button.
4. In the file browser, navigate to your Unity project’s Library/com.unity.addressables folder.
5. Open the AddressablesBuildTEP.json file.

See [Unity Scriptable Build Pipeline](https://docs.unity3d.com/Packages/com.unity.scriptablebuildpipeline@latest) for more information about build performance logging.

# Upgrading to the Addressables system

This article covers how to modify your existing project to take advantage of Addressable assets.

Outside of the Addressables system, Unity provides a few "traditional" ways to reference and load assets:

* **Scene data**: Assets you add directly to a Scene or to a component in a Scene, which the application loads automatically. Unity packages serialized scene data and the assets directly referenced by a scene into a single archive that it includes in your built player application. See [Converting Scenes](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsMigrationGuide.html#converting-scenes) and [Using Addressable assets in non-Addressable Scenes](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsMigrationGuide.html#addressables-in-regular-scenes).
* Prefabs: Assets you create using GameObjects and components, and save outside a Scene. See [Converting Prefabs](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsMigrationGuide.html#converting-prefabs).
* **Resources folders**: Assets you place in your project’s Resources folders and load using the Resources API. Unity packages assets in Resources files into a single archive that it includes in your built player application. The Resources archive is separate from the Scene data archive. See [Converting Resources folders](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsMigrationGuide.html#converting-resources-folders).
* **AssetBundles**: Assets you package in AssetBundles and load with the AssetBundle API. See [Converting AssetBundles](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsMigrationGuide.html#converting-assetbundles).
* **StreamingAssets**: Files you place in the StreamingAssets folder. Unity includes any files in the StreamingAssets folder in your built player application as is. See [Files in StreamingAssets](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AddressableAssetsMigrationGuide.html#files-in-streamingassets)

## Converting to Addressables

Content built using Addressables only reference other assets built in that Addressables build. Content that is used or referenced to that is included within both Addressables, and the Player build through the **Scene data** and **Resource folders** is duplicated on disk and in memory if they are both loaded. Due to this limitation the recommended best practice is to convert all **Scene data** and **Resource folders** to the Addressables build system. Reducing the memory overhead due to duplication and allowing all content to be managed using Addressables. Allowing for the content to be either local or remote as well as updatable through [Content Update](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/ContentUpdateWorkflow.html) builds.

### Converting Scenes

The easiest way to integrate Addressables into a project is to move your Scenes out of the [**Build Settings**](https://docs.unity3d.com/2019.4/Documentation/Manual/BuildSettings.html) list and make those scenes Addressable. You do need to have one Scene in the list, which is the Scene Unity loads at application startup. You can make a new Scene for this that does nothing else than load your first Addressable Scene.

To convert your Scenes:

1. Make a new "initialization" Scene.
2. Open the **Build Settings** window (menu: **File > Build Settings**).
3. Add the initialization Scene to the Scene list.
4. Remove the other Scenes from the list.
5. Click on each Scene in the project list and check the Addressable option in its Inspector window. Alternatively, you can drag Scene assets to a group in the Addressables Groups window. (Don't make your new initialization Scene Addressable.)
6. Update the code you use to load Scenes to use the [Addressables](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.html) class Scene loading methods rather than the SceneManager methods.

At this point, you have included all the assets you have in your Scenes in an Addressable group and the Addressables system packages them in an AssetBundle when you build your Addressables content. If you only use one group for all your Scenes, the runtime loading and memory performance should be roughly equivalent to your project’s pre-Addressables state.

You can now split your one, large Addressable Scene group into multiple groups. The best way to do that depends on the project goals. To proceed, you can move your Scenes into their own groups so that you can load and unload each of them independently of each other. As you do this, you can use the [Analyze tool](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AnalyzeTool.html) to check for duplicated assets that are shared between multiple Scenes. You can avoid duplicating an asset referenced from two different bundles by making the asset itself Addressable. It's often better to move shared assets to their own group as well to reduce interdependencies among your AssetBundles.

#### Using Addressable assets in non-Addressable Scenes

For Scenes that you don't want to make Addressable, you can still use Addressable assets as part of the Scene data through [AssetReferences](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AssetReferences.html).

When you add an AssetReference field to a custom MonoBehaviour or ScriptableObject class, you can assign an Addressable asset to the field in the Unity Editor in much the same way that you would assign an asset as a direct reference. The main difference is that you need to add code to your class to load and release the asset assigned to the AssetReference field (whereas Unity loads direct references automatically when it instantiates your object in the Scene).

##### NOTE

You cannot use Addressable assets for the fields of any UnityEngine components in a non-Addressable Scene. For example, if you assign an Addressable mesh asset to a MeshFilter component in a non-Addressable Scene, Unity does not use the Addressable version of that mesh data for the Scene. Instead, Unity copies the mesh asset and includes two versions of the mesh in your application, one in the AssetBundle built for the Addressable group containing the mesh and one in the built-in Scene data of the non-Addressable Scene. (When used in an Addressable Scene, Unity does not copy the mesh data and always loads it from the AssetBundle.)

To replace direct references with AssetReferences in your custom classes, follow these steps:

1. Replace your direct references to objects with asset references (for example, public GameObject directRefMember; becomes public AssetReference assetRefMember;).
2. Drag assets onto the appropriate component’s Inspector, as you would for a direct reference.
3. Add runtime code to load the assigned asset using the [Addressables](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.html) API.
4. Add code to release the loaded asset when no longer needed.

See [Asset References](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AssetReferences.html) for more information about using AssetReference fields.

See [Loading Addressable assets](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/LoadingAddressableAssets.html) for more information about loading Addressable assets.

### Converting Prefabs

To convert a Prefab into an Addressable asset, check the **Addressables** option in its Inspector window or drag it to a group in the Addressables [Groups](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html) window.

You don't always need to make Prefabs Addressable when used in an Addressable Scene; Addressables automatically includes Prefabs that you add to the Scene hierarchy as part of the data contained in the Scene’s AssetBundle. If you use a Prefab in more than one Scene, however, you should make the Prefab into an Addressable asset so that the Prefab data isn't duplicated in each Scene that uses it. You must also make a Prefab Addressable if you want to load and instantiate it dynamically at runtime.

##### NOTE

If you use a Prefab in a non-Addressable Scene, Unity copies the Prefab data into the built-in Scene data whether the Prefab is Addressable or not. You can identify assets duplicated between your Addressable asset groups and your non-Addressable Scene data using the **Check Scene to Addressable Duplicate Dependencies** rule in the [Analyze tool](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AnalyzeTool.html).

### Converting Resources folders

If your project loads assets in Resources folders, you can migrate those assets to the Addressables system:

1. Make the assets Addressable. To do this, either enable the **Addressable** option in each asset's Inspector window or drag the assets to groups in the Addressables [Groups](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html) window.
2. Change any runtime code that loads assets using the [Resources](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Resources.html) API to load them with the [Addressables](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.html) API.
3. Add code to release loaded assets when no longer needed.

As with Scenes, if you keep all the former Resources assets in one group, the loading and memory performance should be equivalent. Depending on your project, you can improve performance and flexibility by dividing your assets into separate groups. Use the [Analyze tool](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AnalyzeTool.html) to check for unwanted duplication across AssetBundles.

When you mark an asset in a Resources folder as Addressable, the system automatically moves the asset to a new folder in your project named Resources\_moved. The default address for a moved asset is the old path, omitting the folder name. For example, your loading code might change from:

Resources.LoadAsync\<GameObject\>("desert/tank.prefab");

to:

Addressables.LoadAssetAsync\<GameObject\>("desert/tank.prefab");.

You might have to implement some functionality of the [Resources](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Resources.html) class differently after modifying your project to use the Addressables system.

For example, consider the [Resources.LoadAll](https://docs.unity3d.com/ScriptReference/Resources.LoadAll.html) function. Previously, if you had assets in a folder Resources/MyPrefabs/, and ran Resources.LoadAll<SampleType>("MyPrefabs");, it would have loaded all the assets in Resources/MyPrefabs/ matching type SampleType. The Addressables system doesn't support this exact functionality, but you can achieve similar results using Addressable [labels](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Labels.html).

### Converting AssetBundles

When you first open the **Addressables Groups** window, Unity offers to convert all AssetBundles into Addressables groups. This is the easiest way to migrate your AssetBundle setup to the Addressables system. You must still update your runtime code to load and release assets using the [Addressables](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.html) API.

If you want to convert your AssetBundle setup manually, click the **Ignore** button. The process for manually migrating your AssetBundles to Addressables is similar to that described for Scenes and the Resources folder:

1. Make the assets Addressable by enabling the **Addressable** option on each asset’s Inspector window or by dragging the asset to a group in the Addressables [Groups](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/Groups.html) window. The Addressables system ignores existing AssetBundle and Label settings for an asset.
2. Change any runtime code that loads assets using the [AssetBundle](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/AssetBundle.html) or [UnityWebRequestAssetBundle](https://docs.unity3d.com/2019.4/Documentation/ScriptReference/Networking.UnityWebRequestAssetBundle.html) APIs to load them with the [Addressables](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/api/UnityEngine.AddressableAssets.Addressables.html) API. You don't need to explicitly load AssetBundle objects themselves or the dependencies of an asset; the Addressables system handles those aspects automatically.
3. Add code to release loaded assets when no longer needed.

##### NOTE

The default path for the address of an asset is its file path. If you use the path as the asset's address, you'd load the asset in the same manner as you would load from a bundle. The Addressable Asset System handles the loading of the bundle and all its dependencies.

If you chose the automatic conversion option or manually added your assets to equivalent Addressables groups, then, depending on your group settings, you end up with the same set of bundles containing the same assets. (The bundle files themselves won't be identical.) You can check for unwanted duplication and other potential issues using the [Analyze tool](https://docs.unity3d.com/Packages/com.unity.addressables@1.19/manual/AnalyzeTool.html). You can make sure that asset loading and unloading behaves as you expect using the [Event viewer] window.

## Files in StreamingAssets

You can continue to load files from the StreamingAssets folder when you use the Addressables system. However, files in this folder cannot be Addressable nor can files reference other assets in your project.

The Addressables system does place its runtime configuration files and local AssetBundles in the StreamingAssets folder during a build. (Addressables removes these files at the conclusion of the build process; you won’t see them in the Editor.)