Addressable Assets development cycle

Asset group schemas

Schemas define a set of data. You can attach schemas to Asset groups in the Inspector. The set of schemas attached to a group defines how the build processes its contents. For example, when building in packed mode, groups with the [BundledAssetGroupSchema](https://docs.unity3d.com/Packages/com.unity.addressables@1.16/api/UnityEditor.AddressableAssets.Settings.GroupSchemas.BundledAssetGroupSchema.html) schema attached to them act as sources for asset bundles. You can combine sets of schemas into templates that you use to define new groups. You can add schema templates via the [AddressableAssetSettings](https://docs.unity3d.com/Packages/com.unity.addressables@1.16/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.html) Inspector.

Build scripts

Build scripts are represented as [ScriptableObject](https://docs.unity3d.com/Manual/class-ScriptableObject.html) Assets in the Project that implement the [IDataBuilder](https://docs.unity3d.com/Packages/com.unity.addressables@1.16/api/UnityEditor.AddressableAssets.Build.IDataBuilder.html) interface. Users can create their own build scripts and add them to the [AddressableAssetSettings](https://docs.unity3d.com/Packages/com.unity.addressables@1.16/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.html) object through its Inspector. To apply a build script in the **Addressables Groups** window (**Window** > **Asset Management** > **Addressables** > **Groups**), select **Play Mode Script**, and choose a dropdown option. Currently, there are three scripts implemented to support the full application build, and three Play mode scripts for iterating in the Editor.

Play mode scripts

The Addressable Assets package has three build scripts that create Play mode data to help you accelerate app development.

Use Asset Database (faster)

Use Asset Database mode ([BuildScriptFastMode](https://docs.unity3d.com/Packages/com.unity.addressables@1.16/api/UnityEditor.AddressableAssets.Build.DataBuilders.BuildScriptFastMode.html)) allows you to run the game quickly as you work through the flow of your game. It loads Assets directly through the Asset database for quick iteration with no analysis or AssetBundle creation.

Simulate Groups (advanced)

Simulate Groups mode ([BuildScriptVirtualMode](https://docs.unity3d.com/Packages/com.unity.addressables@1.16/api/UnityEditor.AddressableAssets.Build.DataBuilders.BuildScriptVirtualMode.html)) analyzes content for layout and dependencies without creating AssetBundles. Assets load from the Asset database though the [ResourceManager](https://docs.unity3d.com/Packages/com.unity.addressables@1.16/api/UnityEngine.ResourceManagement.ResourceManager.html), as if they were loaded through bundles. To see when bundles load or unload during game play, view the Asset usage in the [**Addressables Event Viewer** window](https://docs.unity3d.com/Packages/com.unity.addressables@1.16/manual/MemoryManagement.html#the-addressables-event-viewer) (**Window** > **Asset Management** > **Addressables** > **Event Viewer**).

Simulate Groups mode helps you simulate load strategies and tweak your content groups to find the right balance for a production release.

Use Existing Build (requires built groups)

Use Existing Build mode most closely matches a deployed application build, but it requires you to build the data as a separate step. If you aren't modifying Assets, this mode is the fastest since it does not process any data when entering Play mode. You must either build the content for this mode in the **Addressables Groups** window (**Window** > **Asset Management** > **Addressables** > **Groups**) by selecting **Build** > **New Build** > **Default Build Script**, or using the [AddressableAssetSettings.BuildPlayerContent()](https://docs.unity3d.com/Packages/com.unity.addressables@1.16/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.html#UnityEditor_AddressableAssets_Settings_AddressableAssetSettings_BuildPlayerContent) method in your game script.

If under **New Build** there is an unclickable **No Build Script Available**, check [AddressableAssetSettings](https://docs.unity3d.com/Packages/com.unity.addressables@1.16/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.html) in the Inspector and see **Build and Play Mode Scripts** section. In order to show up under **New Build** in **Addressables Groups** window , there must be a build script [ScriptableObject](https://docs.unity3d.com/Manual/class-ScriptableObject.html) that is able to build type [AddressablesPlayerBuildResult](https://docs.unity3d.com/Packages/com.unity.addressables@1.16/api/UnityEditor.AddressableAssets.Build.AddressablesPlayerBuildResult.html) paired with an entry in the **Build and Play Mode Scripts** section of the Inspector window for AddressableAssetSettings.

To add a new Build or Play Mode script, click the + under the **Build and Play Mode Scripts** section and find your build mode asset. Once it is added, if the script is a Play Mode script it will show up under **Window** > **Asset Management** > **Addressables** > **Groups** > **Play Mode Script**. If the script is able to build [AddressablesPlayerBuildResult](https://docs.unity3d.com/Packages/com.unity.addressables@1.16/api/UnityEditor.AddressableAssets.Build.AddressablesPlayerBuildResult.html) it will show up under **Window** > **Asset Management** > **Addressables** > **Groups** > **Build** > **New Build**. Build and Play Mode scripts provided by default, including BuildSciptPackedMode, are located under Assets/AddressableAssetsData/DataBuilders. See earlier section "Build scripts" for more information on custom build scripts.

Analysis and debugging

By default, Addressable Assets only logs warnings and errors. You can enable detailed logging by opening the **Player** settings window (**Edit** > **Project Settings...** > **Player**), navigating to the **Other Settings** > **Configuration** section, and adding "ADDRESSABLES\_LOG\_ALL" to the **Scripting Define Symbols** field.

You can also disable exceptions by unchecking the **Log Runtime Exceptions** option in the [AddressableAssetSettings](https://docs.unity3d.com/Packages/com.unity.addressables@1.16/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.html) object Inspector. You can implement the [ResourceManager.ExceptionHandler](https://docs.unity3d.com/Packages/com.unity.addressables@1.16/api/UnityEngine.ResourceManagement.ResourceManager.html#UnityEngine_ResourceManagement_ResourceManager_ExceptionHandler) property with your own exception handler if desired, but this should be done after Addressables finishes runtime initialization (see below).

Enable the [build layout report](https://docs.unity3d.com/Packages/com.unity.addressables@1.16/manual/DiagnosticTools.html#build-layout-report) to get information and statistics about your content builds. You can use this report to help verify that your builds are creating your bundles as you expect.

Initialization objects

You can attach objects to the Addressable Assets settings and pass them to the initialization process at runtime. The [CacheInitializationSettings](https://docs.unity3d.com/Packages/com.unity.addressables@1.16/api/UnityEditor.AddressableAssets.Settings.CacheInitializationSettings.html) object controls Unity's caching API at runtime. To create your own initialization object, create a ScriptableObject that implements the [IObjectInitializationDataProvider](https://docs.unity3d.com/Packages/com.unity.addressables@1.16/api/UnityEngine.ResourceManagement.Util.IObjectInitializationDataProvider.html) interface. This is the Editor component of the system responsible for creating the [ObjectInitializationData](https://docs.unity3d.com/Packages/com.unity.addressables@1.16/api/UnityEngine.ResourceManagement.Util.ObjectInitializationData.html) that is serialized with the runtime data.

Customizing URL Evaluation

There are several scenarios where you will need to customize the path or URL of an Asset (an AssetBundle generally) at runtime. The most common example is creating signed URLs. Another might be dynamic host determination.

The code below is an example of appending a query string to all URLs:

//Implement a method to transform the internal ids of locations

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;

}

\****Please Note***\*: When bundling video files into Addressables with the intent of loading them on the Android platform, you must create a [CacheInitializationSettings](https://docs.unity3d.com/Packages/com.unity.addressables@1.16/api/UnityEditor.AddressableAssets.Settings.CacheInitializationSettings.html) object, disable Compress Bundles on that object, then add it to the list of Initialization Objects on the [AddressableAssetSettings](https://docs.unity3d.com/Packages/com.unity.addressables@1.16/api/UnityEditor.AddressableAssets.Settings.AddressableAssetSettings.html) object if it has not been already.

Content update workflow

Update workflow moved to a new page: [Content Update Workflow](https://docs.unity3d.com/Packages/com.unity.addressables@1.16/manual/ContentUpdateWorkflow.html)

Multiple Projects

Some users find it beneficial to split their project into multiple Unity projects, such as isolating the artwork from the code to lessen import times.

In order to take advantage of this multiple project setup you'll need to utilize [Addressables.LoadContentCatalogAsync(...)](https://docs.unity3d.com/Packages/com.unity.addressables@1.16/manual/LoadContentCatalogAsync.html) to load the content catalogs of your separate projects in your main project.

A general multi-project workflow is as follows:

1. Create your main project (Project A) and ancillary project(s) (Project(s) B, C, etc.).
2. Add Addressables in each project and setup the desired Addressable Asset Entries.
3. Build your Addressable Player Content for each project.
4. In your main project, before attempting to load assets from the other projects, load the desired content catalogs from the other projects using Addressables.LoadContentCatalogAsync(...)
5. Use Addressables as normal.

Note: Ensure that the content catalogs and Asset Bundles of the other projects are reachable by the main project. Setup any required hosting services beforehand.