# 目录

目录

[目录 1](#_Toc497)

[接入流程 2](#_Toc31254)

[适配资源表 2](#_Toc19311)

[初始化打包工具 3](#_Toc10938)

[接入运行时代码 3](#_Toc20675)

[使用打包工具 4](#_Toc6421)

[代码目录 4](#_Toc15947)

[编辑器界面 5](#_Toc3034)

[打包配置界面 5](#_Toc1161)

[固定包配置界面 6](#_Toc27544)

[AB检查窗口 7](#_Toc9092)

[使用AssetManager资源管理器 8](#_Toc21288)

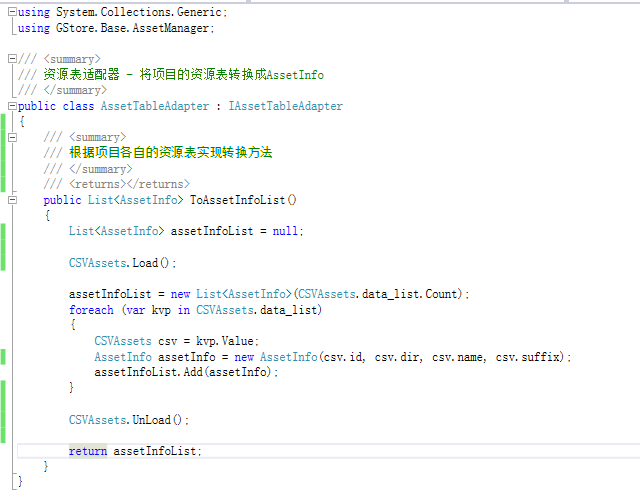
[代码目录 8](#_Toc8690)

[API接口 8](#_Toc12447)

# 接入流程

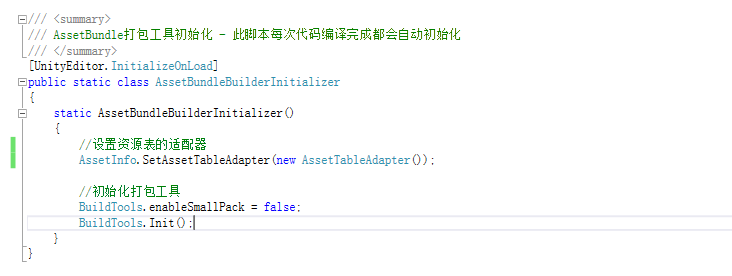
## 适配资源表

实现一个资源表适配器，实现**IAssetTableAdapter**接口，将项目的资源表转换成统一的**AssetInfo**。例如：



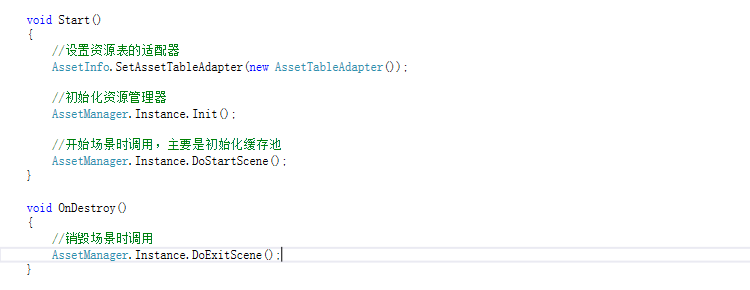
## 初始化打包工具

在**AssetBundleBuilderInitializer**中实现打包工具的初始化代码，包括资源表的适配和打包工具的配置。例如：



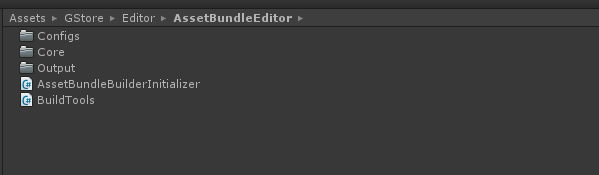
## 接入运行时代码

游戏初始化时，需要先加载资源表，再初始化资源管理器。然后在场景开始的时候，调用**AssetManager.Instance.DoStartScene()**，初始化对象缓冲池；在场景切换的时候，调用**AssetManager.Instance.DoExitScene()**，回收缓冲池资源。



使用打包工具

## 代码目录



目录说明：

**Configs**：存放打包工具的一些配置文件，包括打包工具自身的配置参数和固定包配置信息，调整打包配置后需要保存上传这部分配置到svn。

**Core**：打包工具的核心代码，平台部维护，项目组不得修改。可以使用外链的方式从平台部svn获取。

**Output**：打包操作生成的记录文件，包括BundleTable、DepBundleTable、和missing\_file\_record等。其中BundleTable会复制到AssetBundles文件夹，一起打进应用包体内。其它文件保存在这里用于查询打包信息。

文件说明：

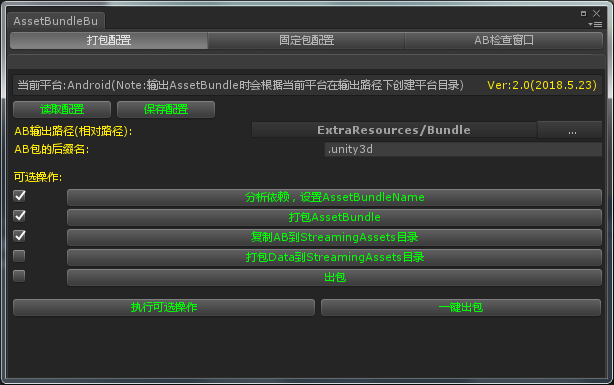
**AssetBundleBuilderInitializer**：打包工具初始化脚本，默认在此处加载资源表和初始话打包工具，项目可根据项目实际需求在这里调整初始化的配置。比如根据setup.xml的配置切换大小包等。

**BuildTools**：注册打包过程中各个方法，为打包编辑器和云构建提供各种打包接口。默认包含：分析AB依赖、打包AB、导入AB、打包数据、最终出包五个步骤，项目组可根据需要在这个类中添加和注册新的打包步骤。

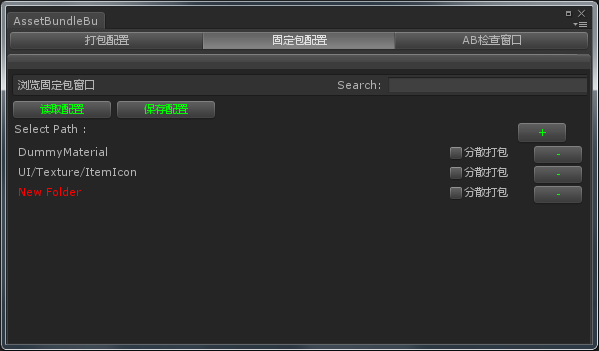
## 编辑器界面

通过点击GStore/Build/AssetBundleBuilder，可以打开打包工具的编辑器界面。

### 打包配置界面

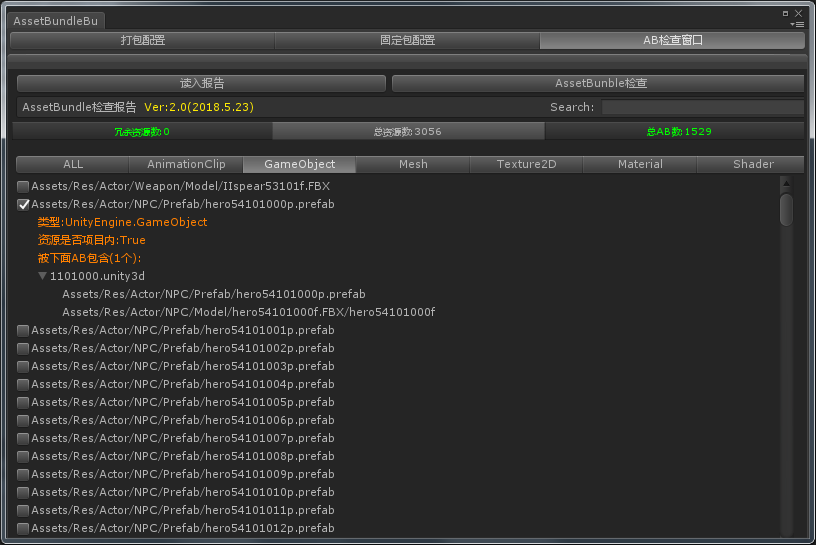
目前只支持修改AB输出路径和AB后缀。下方操作区域可直接执行单步操作，或勾选需要的步骤依次执行。通过此界面可以完成所有的打包相关操作。

### 固定包配置界面



1. 通过在Project窗口选中Res下的子目录，点击“+”号，则可将该目录加入固定包列表。
2. 点击对应条目的“-”号，则将该目录移出固定包列表。
3. 勾选分散打包，则目录下的资源按文件数量单独打包，否则合成同一个AB包。
4. 若固定包配置的目录在项目中已移除，编辑器列表则标红显示，表示改目录已丢失，该条配置自动忽略不生效，若项目确定不再需要该目录，则应删除该条配置。
5. 固定包可用于将没有assetId索引的资源加入打包列表，通过路径加载。合并打包时可用增大AB粒度，并且在发布AB后仍支持LoadAllAsset接口。通常使用合并打包。

### AB检查窗口

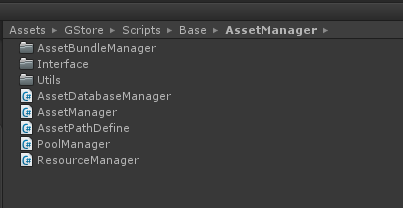


此工具主要用于浏览分析AB包的冗余情况。

1. 点击AssetBundle检查按钮，浏览到要检查的AssetBundle目录，确定后开始分析该目录的所有AssetBundle，并在目录下生成检查报告。
2. 生成报告后会自动打开报告，或者通过点击读入报告可以打包之前生成的旧报告。
3. 打开报告后有三个视图，冗余资源、所有资源和所有AB的浏览界面。选中展开后可以看到详细信息以及和AB之间的关系。

# 使用AssetManager资源管理器

## 代码目录



目录说明：

**AssetBundleManager**：AssetBundle管理器的相关代码。

**Interface**：接口定义。

**Utils**：工具代码和一些辅助类。

文件说明：

AssetManager：提供资源管理器对外统一的加载、缓存、实例化接口。  
AssetDatabaseManager：编辑器下从AssetDatabase加载资源的实现。

ResourceManager：对Resources.Load接口的封装管理。

PoolManager：GameObject对象池的实现和管理。

AssetPathDefine：一些资源相关的路径定义。

## API接口

目前参考Example.cs中的示例代码，API文档待整理。