GameDriver Pro

GameDriver Pro is a game framework for Unity engine

Getting Start

Overview

This User Manaul was designed to provide GaemDriver Pro users with a basic overview of the features and functionality of the tool.

Install

1. After downloading GameRiver Pro from Unity's Asset Store, go to: "Assets->Import Package->Custom Package…".

2. In the Import Assets window, find and select the GameDriver Pro UnityPackage file.
3. Once Unity appears in the Import Packages window, verify that Import is selected for all projects, then click the Import button at the bottom right of the window.
4. All GameDriver Pro files will be added to Assets/GameDriver.

You can also select the desired section to import.

support

If you want to quickly understand the important features in GameDriver Pro, you can directly refer to the examples in GameDriver/Samples.

If you want to learn more about the design ideas and details of each functional module in GameDriver Pro, you can find more information and FAQs in Online Help .

If you can't find the information you're looking for, please file an issue (or submit a pull request) to describe your experience.

Or contact the author: xuzhuoxi@gmail.com or mailxuzhuoxi@163.com

Features

Source code storage locations in GameDriver Pro are regular.

- The source code in **GameDriver/Runtime/CSharp** only depends on the C# standard library, ** does not depend on the Unity standard library. It is an extension to C#.

- The source code in **GameDriver/Runtime/Actions** depends on CSharp and the Unity standard library. It is a functional extension to the Unity engine.

- The source code in **GameDriver/Runtime/Games** depends on CSharp and Actions. It is a general implementation of system functions commonly used in the game development process.

Functional Overview

GameDriver/Runtime/CSharp

The namespace is **JLGame.GameDriver.CSharp**

- Algs
  Transparent algorithm related functions
  - AStar
    A* general algorithm

- Archive
  File archiving related functions, now including zip decompression function
- Buffer
  The byte data cache function supports reading and writing of big and small endian settings and basic data types.
- DateTimex
  Time related functions. Contains calculation conversions between time units, string formatting of times, timers that support pause and resume, and more.
- Event
  Event related functions. Contains event monitoring, removal and scheduling functions. For details, see 1.Event System(Event)
- Extensions
  Extensions for basic data types in CSharp.
- Imagex
  Image related functions. Now includes RGBA data objects, filter processing functions, and more.
- Mathx
  Mathematical calculation related functions.
- Net
  Socket API definition, message encapsulation and unpacking functions, etc.
- Pool
  Object pool related functions. Contains Key-Value object pool, metadata object pool, reuse object pool, etc.
- Service
  Service framework basic support. For details, see 7. Service Framework(Service)
- TinyJson
  Simple Json reading and writing library. Open source from Github: https://github.com/zanders3/json .
- Utils
  Common tool functions based on CSharp
  - Encrypt
    Encryption and decryption related functions, now including Des, Rijandel, RSA
  - ArrayUtil
    Array related functions, such as array cloning, merging, type conversion, array matrix rotation and other functions

    - BitUtil
      The **bit operation** function of numeric values.
    - ConfusedUtil
      Data Obfuscation related functions.
      +DirectionUtil
      Directory related functions in the file system.
    - FileUtil
      File related functions in the file system.
    - PathUtil
      Path related functions in the file system.
      +PrintUtil
      Print support for object information.
    - ReflexUtil
      Reflection value-related functions.
    - TextUtil
      Character file read and write functions.
- Xml
  XML related function support: serialization and deserialization
- Callback
  Generic Callback Context
- ICloneable
  Generic clone interface

GameDriver/Runtime/Actions

The namespace is **JLGame.GameDriver.Actions**

- Animatorx
  Feature support for Unity's animation system. Contains animation search, animation playback, event management and other functions.
- AssetIndex
  The Editor function provides the ability to generate a "name -> path" index for assets in the project.
- Audio
  Audio Manager, which provides the function of unified management of audio in the project. For details, please see 5. Audio Manager (AudioManager).
- Component
  Some useful components.
- Component2D
  Some useful 2D components.
- DateTimex
  Time-related functions, including timers that rely on the Unity engine, cycle time slice functions, etc.

- ExcelExporter
  Function support for exporting data from the third-party table export tool ExcelExporter.
- Extensions
  Extensions to commonly used data structures in Unity.
- Graphicsx
  Image related functions. Contains color conversion functions, image pixel data structures, texture functions, etc.
- i18n
  Internationalization function. For details, see 3. Internationalization System(i18n).
  +Jsonx
  Related functions for Json processing.
- Layer
  Unity displays the structural layered design capabilities.
- Loaderx
  Resource loading management module. For details, see 2. Loader System (Loader)).
- Localx
  Depends on the local storage capabilities of PlayerPrefs.
- Pool
  The object pool function extends the object pool function in GameDriver/Runtime/CSharp/Pool, and adds parts related to Unity.
- Service
  Service framework Unity support. For details, see 7. Service Framework(Service)
- TheadEvent
  Event related functions. Added Unity multithreading support to GameDriver/Runtime/CSharp/Event.
- UIElements
  Editor function. It is an extension to the UIElements functionality in Unity.
- Utils
  Rely on common functions of Unity.
  - CameraUtil
    Camera related functions.
  - CaptureUtil
    Screenshot related functions.
  - ComponentUtil
    Component related functions.
    +DebugUtil
    Debug printing related functions.
  - GeometryUtil
    Geometric calculation related functions.

    - IdentifyUtil
      Machine Id related functions.
    - ImageUtil
      Image related functions.
    - MathfUtil
      Mathematical calculation related functions
    - PositionUtil
      Unity object **position calculation** related functions.
    - TilemapUtil
      Common public behavior under the **UnityEngine.Tilemaps** component.
    - TransformUtil
      UnityEngine.Transform component related extension functions.
    - UnityPathUtil
      Untiy-related path handling functions.
- Wait
  Unity coroutine related functions.

GameDriver/Runtime/Games

The namespace is **JLGame.GameDriver.Games**

- LocalAccount
  Local account storage function.
- NetManager
  Network connection management, virtual servers. For details, see 4. Network Manager (NetManager).
- PanelManager
  Panel management module. For details, see 6. Panel Manager (PanelManager) +RpgMaterial
  Game data management system. Please see 8.RPG Material Data System(RpgMaterial) for details.
- Service
  Common service instances under the service framework. For details, see 7. Service Framework(Service).
- Tiledx
  Data interpretation capabilities for third-party software Tiled.

Common function modules

1. Event - Efficient event module, supports Unity multithreading

The event adopts the "**monitoring-capture**" mechanism, which supports the settings of the number of captures and capture priority during monitoring.

- **JLGames.GameDriver.CSharp.Event** is responsible for the core logic implementation of the event system.
- **JLGames.GameDriver.Actions.ThreadEvent** provides support for multithreading.

1.1 Core functions - add listeners, remove listeners, dispatch events

1.1.1 Monitoring

The `AddEventListener` series of functions in the interface IEventListener are used to add event listeners.
The logic implementation has been completed in the implementation class

EventDispathver.

```
/// <summary>
/// Add single event listener
/// 添加单次事件监听
/// </summary>
/// <param name="type">Event Type<br/>事件类型</param>
/// <param name="handler">Listener Function<br/>监听函数</param>
/// <param name="weight">Response Weight<br/>响应权重</param>
/// <param name="tag">Function Tag<br/>函数的唯一标签</param>
void OnceEventListener(string type, EventDelegates.EventHandler handler, int weight =
EventConst.DefaultWeight, string tag = null);

/// <summary>
/// Add event listener
/// 添加事件监听
/// </summary>
/// <param name="type">Event Type<br/>事件类型</param>
/// <param name="handler">Listener Function<br/>监听函数</param>
/// <param name="tag">Function Tag<br/>函数的唯一标签</param>
void AddEventListener(string type, EventDelegates.EventHandler handler, string tag = null);

/// <summary>
/// Add event listener
/// 添加事件监听
/// </summary>
/// <param name="type">Event Type<br/>事件类型</param>
/// <param name="handler">Listener Function<br/>监听函数</param>
/// <param name="weight">Response Weight<br/>响应权重</param>
/// <param name="tag">Function Tag<br/>函数的唯一标签</param>
void AddEventListener(string type, EventDelegates.EventHandler handler, int weight, string tag =
null);

/// <summary>
/// Add event listener
/// 添加事件监听
/// </summary>
/// <param name="type">Event Type<br/>事件类型</param>
/// <param name="handler">Listener Function<br/>监听函数</param>
/// <param name="listeningTimes">Times of responses<br/>响应次数</param>
/// <param name="tag">Function Tag<br/>函数的唯一标签</param>
void AddEventListener(string type, EventDelegates.EventHandler handler, uint listeningTimes ,
string tag = null);

/// <summary>
/// Add event listener
/// 添加事件监听
/// </summary>
/// <param name="type">Event Type<br/>事件类型</param>
/// <param name="handler">Listener Function<br/>监听函数</param>
/// <param name="weight">Response Weight<br/>响应权重</param>
/// <param name="listeningTimes">Times of responses<br/>响应次数</param>
/// <param name="tag">Function Tag<br/>函数的唯一标签</param>
void AddEventListener(string type, EventDelegates.EventHandler handler, int weight, uint
listeningTimes , string tag = null);
```

1.1.2 remove monitor

The `RemoveEventListener` series of functions in the interface IEventListener are used to remove event listeners.
The logic implementation has been completed in the implementation class EventDispathver.

```
/// <summary>
/// Delete event listener
/// 删除事件监听
/// </summary>
/// <param name="type">事件类型</param>
/// <param name="handler">监听函数</param>
/// <param name="tag"></param>
void RemoveEventListener(string type, EventDelegates.EventHandler handler, string tag = null);

/// <summary>
/// Delete event listener
/// 删除事件监听
/// </summary>
/// <param name="type">事件类型</param>
/// <param name="tag"></param>
void RemoveEventListener(string type, string tag);

/// <summary>
/// Delete a type of event listeners
/// 删除一类事件监听
/// </summary>
/// <param name="type">事件类型</param>
void RemoveEventListener(string type);

/// <summary>
/// Clear all event listeners
/// 清除全部事件监听
/// </summary>
void RemoveEventListener();
```

1.1.3 Distributing events

The `DispatchEvent` series of functions in the interface IEventDispatcher are used to distribute events.
The logic implementation has been completed in the implementation class EventDispathver.

```
/// <summary>
/// Trigger an event of a certain type and pass data
/// 触发某一类型的事件,并传递数据
/// </summary>
/// <param name="evd">Evd.</param>
void DispatchEvent(EventData evd);

/// <summary>
/// Trigger an event of a certain type and pass data
/// 触发某一类型的事件,并传递数据
/// </summary>
/// <param name="type">事件类型</param>
/// <param name="data">事件的数据(可为null)</param>
void DispatchEvent(string type, object data);
```

1.1.4 Multithreading support

The functions declared in the interface IThreadEventProxy define the multithreading support specification.
The logic implementation has been completed in the implementation class ThreadEventDispatcher.

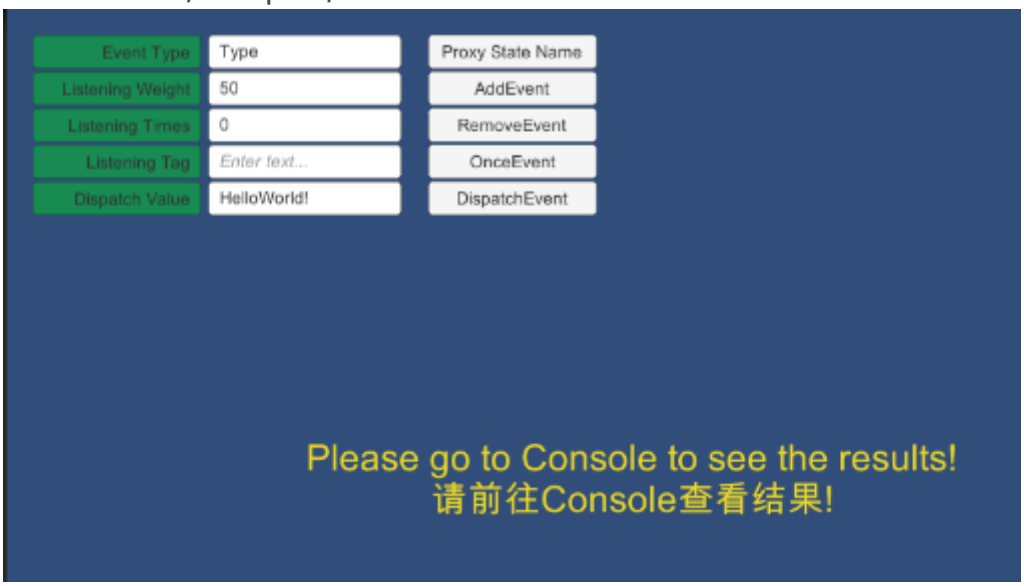
```
/// <summary>
/// Enable main thread proxy mode
/// 开启主线程代理模式
/// </summary>
/// <param name="carrier"></param>
void OpenMainThreadProxy(GameObject carrier);

/// <summary>
/// Turn off main thread proxy mode
/// 关闭主线程代理模式
/// </summary>
void CloseMainThreadProxy();
```

1.1.5 Extensions

Extends EventDispathver or ThreadEventDispatcher and implements a custom interface that can be used to extend event behavior.

1.2 Example

GameDriver/Samples/Event



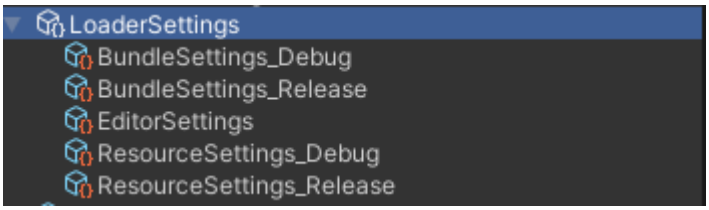
2. Loading management module (Loader) - supports free switching of three modes: Resources, Editor, Assetbundle

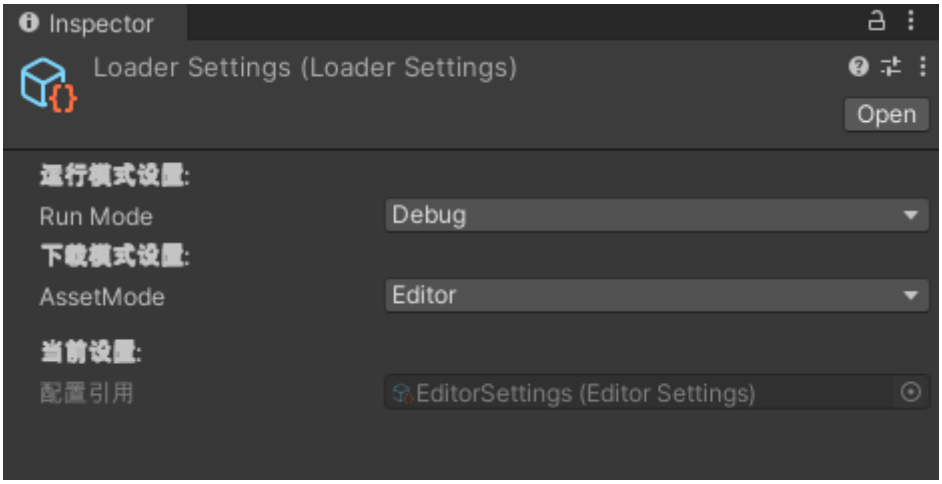
- **JLGames.GameDriver.Actions.Loader** provides full support for the loader.
- **“Tools -> GameDriver -> Project -> Gen LoaderSettings”** provides the generation entry for the loader configuration file.

2.1 Initialization

2.1.1 Generate configuration assets

Execute menu “Tools -> GameDriver -> Project -> Gen LoaderSettings”.
The LoaderSettings.asset (renameable) file will be generated under the project Assets/Resources.

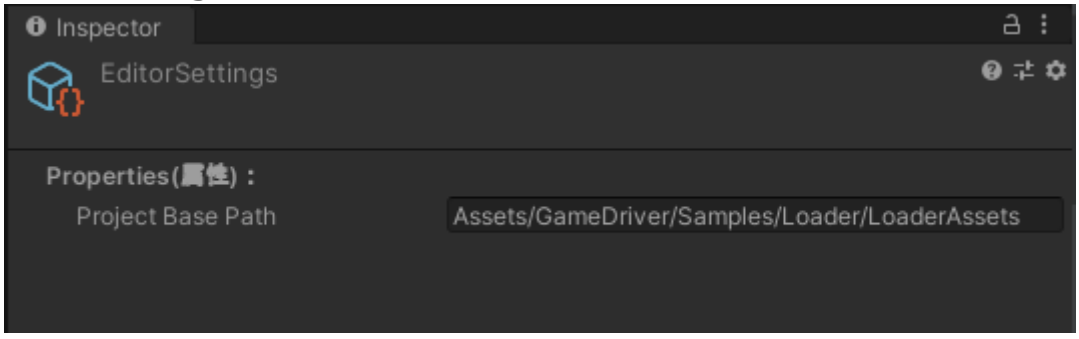




2.1.2 Set configuration according to project requirements

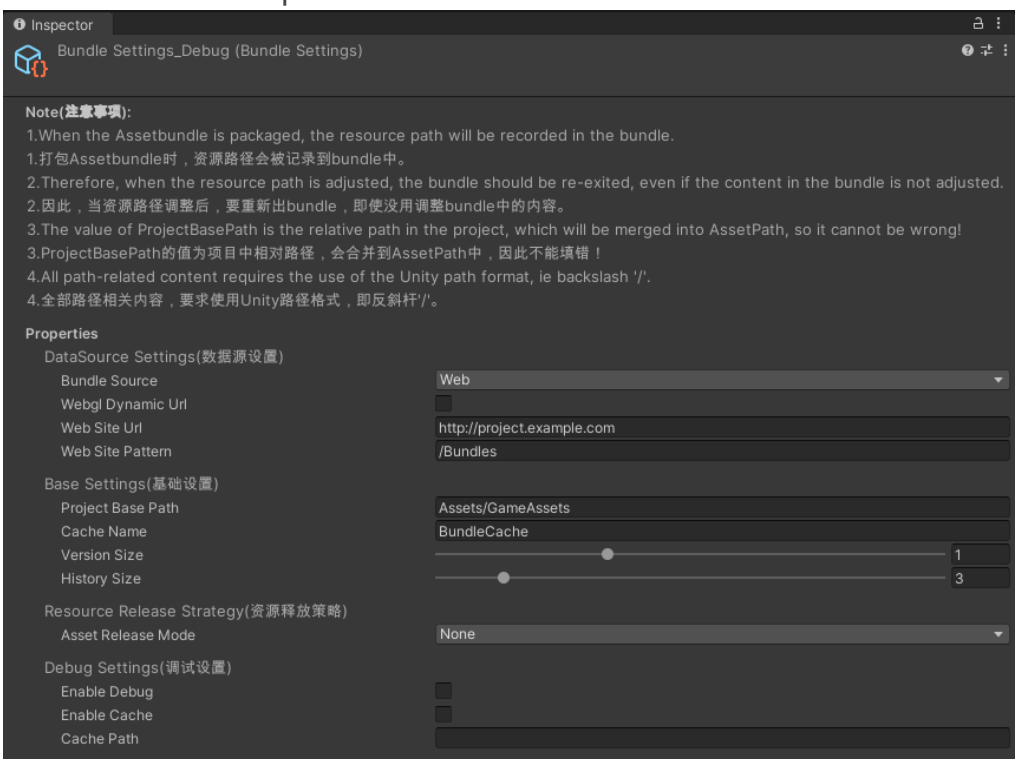
There are 5 available configurations in LoaderSettings, one Editor mode, two Resource modes, and two AssetBundle modes.

- Editor configuration ProjectBasePath can set the base path part of the project’s resource search, which can be spliced before the resource path when loading resources.



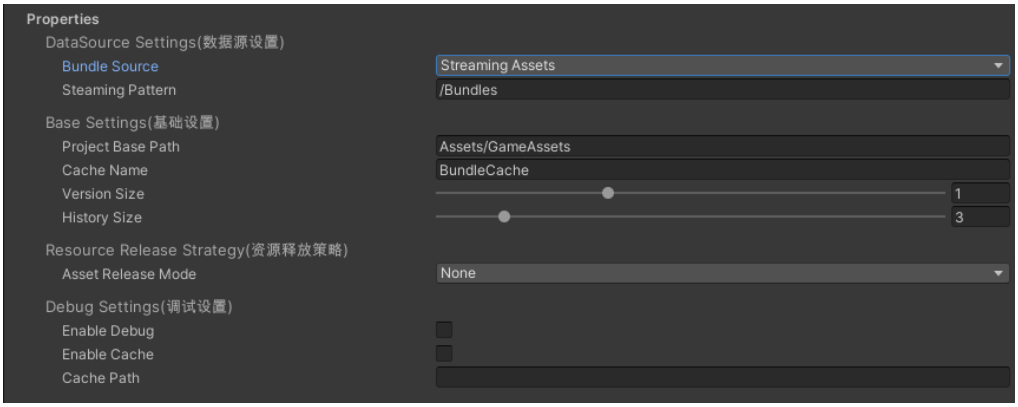
- Resource configuration ProjectBasePath can set the base path part of the project’s resource search, which can be spliced before the resource path when loading resources.

- AssetBundle configuration
    - BundleSource You can choose three modes: Web, Streaming Assets, and File Debug.
        - Web Used to load AssetBundle assets stored directly on the site.
          Fill in the site Url in Web Site Url.
          Fill in the Parttern path in the Web Site Pattern.

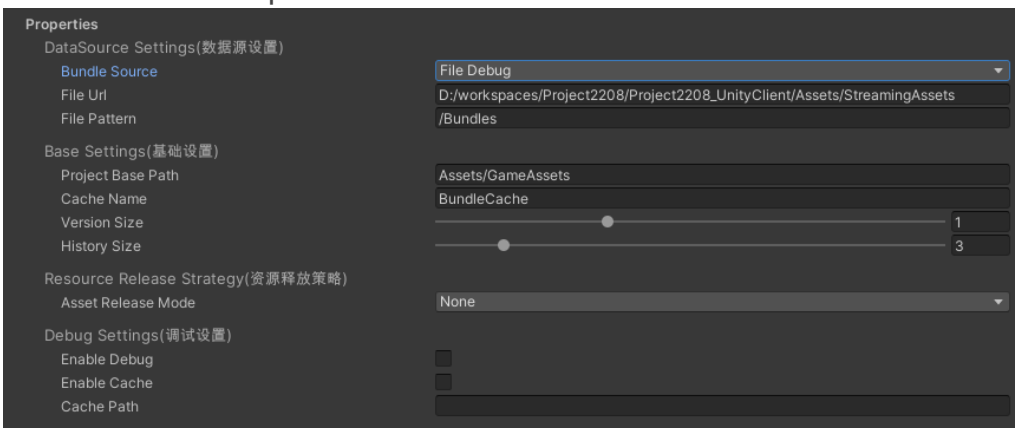


        - Streaming Assets Used to load AssetBundle resources stored in the project StreamingAssets.
          Fill in the relative path relative to StreamingAssets in Streaming

Parttern.

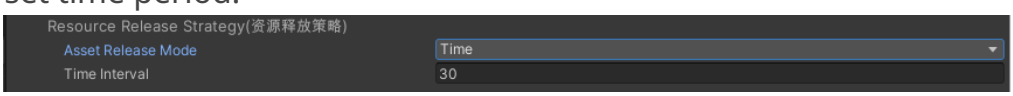


- File Debug Used to load AssetBundle resources in the local file system, mostly used for debugging.
  Fill in the local file path in File Url.
  Fill in the relative path related to File Url in File Pattern.

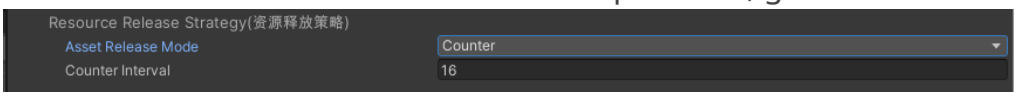


- Base Settings It is mainly about the setting of the Bundle directory information of the project and the setting of the cache information when loading.
  - Project Base Path Set the Bundle directory path of the project
  - CacheName Set the name of the cache area, this can be customized.
  - Version Size Sets the number of versions of each Bundle asset to keep in the cache.
  - History size sets the number of cache areas, each CacheName will generate a cache area.

- Resource Release Strategy There are three options: None, Time, and Counter.
  - None means that the release strategy is not selected. The invalid memory occupation generated after releasing the Bundle and assets needs to be released by calling gc. This strategy is suitable for self-management of memory and gc timing.
  - Time is a timing release strategy, which will call gc according to the set time period.



- Counter is a count release strategy. When the bundle loaded resource reaches the set value and multiple value, gc is called.



2.1.3 Initializing Loaders with Configuration

Initialize the loader with the following API:

- Initialize with the configuration file name generated by first point.

```
/// <summary>
/// Initialize an instance of ILoader
/// 初始化ILoader的实例
/// </summary>
/// <param name="loaderName"></param>
/// <param name="settingsName"></param>
/// <returns></returns>
public static ILoader InitLoader(string loaderName = DefaultName, string settingsName = DefaultName)
{
    if (string.IsNullOrEmpty(settingsName)) return null;
    var settings = LoadLoaderSettings(settingsName);
    return InitLoader(loaderName, settings);
}
```

- Initialize with a configuration instance.

```
/// <summary>
/// Initialize an instance of ILoader
/// 初始化ILoader的实例
/// </summary>
/// <param name="loaderName"></param>
/// <param name="settings"></param>
/// <returns></returns>
public static ILoader InitLoader(string loaderName, LoaderSettings settings)
{
    if (string.IsNullOrEmpty(loaderName) || null == settings) return null;
    if (m_Loaders.ContainsKey(loaderName)) return m_Loaders[loaderName];
    var loader = InitLoader(settings);
    if (null == loader) return null;
    m_Loaders[loaderName] = loader;
    if (DefaultName == loaderName)
    {
        m_DefaultLoader = loader;
    }

    return loader;
}
```

- There is a shortcut initialization function in the Loader static class:

```
/// <summary>
/// Initialize the default loader
/// 初始化默认加载器
/// </summary>
/// <param name="settingsName"></param>
public static void InitLoader(string settingsName = LoaderManager.DefaultName)
{
    LoaderManager.InitLoader(LoaderManager.DefaultName, settingsName);
}
```

2.1.4 Initialize Bundle version information

Call the function in the loader instance:

```
/// <summary>
/// Initialize version information
/// Version information is not allowed to be cached
/// 初始化版本信息
/// 版本信息获取不允许进行缓存
/// </summary>
/// <param name="onVersionAssetBundleLoaded"></param>
IEnumerator InitVersion(LoaderDelegate.OnAssetLoaded<AssetBundleManifest> onVersionAssetBundleLoaded);
```

- The function callback is executed after the initialization version ends, and the initialization result can be judged internally: success or failure.
- The function requires to open the coroutine call, you can use LoaderManager.Mono to open the coroutine:

```
/// <summary>
/// Initialize version information
/// Version information is not allowed to be cached
/// 初始化版本信息
/// 版本信息获取不允许进行缓存
/// </summary>
/// <param name="onVersionLoaded"></param>
public static void InitVersion(LoaderDelegate.OnAssetLoaded<AssetBundleManifest> onVersionLoaded)
{
    DebugUtil.Log("InitVersion:", LoaderManager.Mono);
    LoaderManager.Mono.StartCoroutine(LoaderManager.DefaultLoader.InitVersion(onVersionLoaded));
}
```

2.2 Use

2.2.1 Load Bundle assets

Bundle assets can be loaded only after the Bundle version information is initialized.
The loader implements the IBundleLoader interface and contains functions related to loading Bundle assets.
Loading Bundle assets requires the use of coroutines, which can be enabled using a LoaderManager.Mono instance to load Bundles by coroutines.

```
/// <summary>
/// Asynchronous download bundle
/// 异步下载资源包
/// </summary>
/// <param name="bundleName">资源包名称</param>
/// <param name="onBundleLoaded">回调</param>
/// <param name="autoRelease">是否自动释放资源包</param>
/// <param name="unloadAllLoadedObjects">是否全部释放实例化的资源</param>
/// <returns></returns>
public static void LoadBundleAsync(string bundleName, LoaderDelegate.OnBundleLoaded onBundleLoaded,
    bool autoRelease = true, bool unloadAllLoadedObjects = false)
{
    LoaderManager.Mono.StartCoroutine(LoaderManager.DefaultLoader.LoadBundleAsync(bundleName, onBundleLoaded,
        autoRelease, unloadAllLoadedObjects));
}
```

- bundleName: bundle asset name.
- onBundleLoaded: Execute the result callback, which can be used to determine whether it is successful or not.
- autoRelease: Indicates whether to release the bundle instance after onBundleLoaded execution ends
- unloadAllLoadedObjects: Indicates whether to release the resource assets instantiated from the bundle instance after onBundleLoaded execution ends

- If autoRelease is true, all assets to be used should be instantiated in onoBundleLoaded.

2.2.2 Loading resource assets

In the case of obtaining the bundle instance, the instance of the resource asset can be instantiated from the bundle instance, and then cloned and used.
It is recommended to use synchronous functions for loading resource assets, and asynchronous functions are not recommended. The reason is that Unity does not support opening coroutines within coroutines friendly, and if there are too many layers (like 16 layers), unpredictable errors will appear.

The IAssetLoader interface functions are divided into four categories:

- Single resource asset loading (sync|sync)

```
/// <summary>
/// Load asset from bundle synchronously with relative path
/// 使用相对路径从Bundle中加载资源
/// </summary>
/// <param name="assetPath">资源路径</param>
/// <param name="bundle">资源包</param>
/// <typeparam name="T">资源类型</typeparam>
/// <returns></returns>
T LoadAssetSync<T>(string assetPath, AssetBundle bundle) where T : Object;

/// <summary>
/// Load asset from bundle synchronously with full path
/// 使用完整路径从Bundle中加载资源
/// The full path: refers to the relative path after removing ProjectBasePath
/// 完整路径：指的是去除ProjectBasePath后的相对路径
/// </summary>
/// <param name="fullPath">完整资源路径,忽略设置的ProjectBasePath</param>
/// <param name="bundle">资源包</param>
/// <typeparam name="T">资源类型</typeparam>
/// <returns></returns>
T LoadAssetSyncFull<T>(string fullPath, AssetBundle bundle) where T : Object;

/// <summary>
/// Asynchronously load resources from bundle
/// 异步从资源包中加载资源
/// </summary>
/// <param name="assetPath">资源路径</param>
/// <param name="bundle">资源包</param>
/// <param name="onAssetLoaded">回调</param>
/// <typeparam name="T">资源类型</typeparam>
/// <returns></returns>
IEnumerator LoadAssetAsync<T>(string assetPath, AssetBundle bundle,
LoaderDelegate.OnAssetLoaded<T> onAssetLoaded) where T : Object;
```

- Batch resource asset loading (sync|sync)

```
/// <summary>
/// Synchronized load resources from bundle in batches
/// 同步批量从资源包中加载资源
/// </summary>
/// <param name="assetPaths">资源路径</param>
/// <param name="bundle">资源包</param>
/// <typeparam name="T">资源类型</typeparam>
/// <returns></returns>
T[] LoadAssetsSync<T>(string[] assetPaths, AssetBundle bundle) where T : Object;

/// <summary>
/// Synchronized load resources from bundle in batches
/// 同步批量从资源包中加载资源
/// </summary>
/// <param name="fullPaths">完整资源路径,忽略设置的ProjectBasePath</param>
/// <param name="bundle">资源包</param>
/// <typeparam name="T">资源类型</typeparam>
/// <returns></returns>
T[] LoadAssetsSyncFull<T>(string[] fullPaths, AssetBundle bundle) where T : Object;

/// <summary>
/// Batch asynchronously load resources from bundle
/// 异步从资源包中批量加载资源
/// </summary>
/// <param name="assetPaths">路径数组</param>
/// <param name="bundle">资源包</param>
/// <param name="onMultiAssetLoaded">回调</param>
/// <typeparam name="T"></typeparam>
/// <returns></returns>
IEnumerator LoadMulitAssetAsync<T>(string[] assetPaths, AssetBundle bundle,
LoaderDelegate.OnMultiAssetLoaded<T> onMultiAssetLoaded)
where T : Object;
```

- Single sub-resource asset loading (sync|async)

```
/// <summary>
/// Load sub-resources (sprites, prefabs, etc.)
/// 加载子资源(子图、预置等)
/// </summary>
/// <param name="path"></param>
/// <param name="subName"></param>
/// <param name="ab"></param>
/// <returns></returns>
Object LoadSubAssetSync(string path, string subName, AssetBundle ab);

/// <summary>
/// Load sub-resources (sprites, prefabs, etc.)
/// 加载子资源(子图、预置等)
/// </summary>
/// <param name="path"></param>
/// <param name="subName"></param>
/// <param name="type"></param>
/// <param name="ab"></param>
/// <returns></returns>
Object LoadSubAssetSync(string path, string subName, Type type, AssetBundle ab);

/// <summary>
/// Load sub-resources (sprites, prefabs, etc.)
/// 加载子资源(子图、预置等)
/// </summary>
/// <param name="path"></param>
/// <param name="subName"></param>
/// <param name="ab"></param>
/// <returns></returns>
T LoadSubAssetSync<T>(string path, string subName, AssetBundle ab) where T : Object;

/// <summary>
/// Load sub-resources (sprites, prefabs, etc.) from bundles asynchronously
/// 异步从资源包中加载加载子资源(子图、预置等)
/// </summary>
/// <param name="assetPath">资源路径</param>
/// <param name="subName"></param>
/// <param name="bundle">资源包</param>
/// <param name="onAssetLoaded">Callback(回调)</param>
/// <typeparam name="T">Resource type(资源类型)</typeparam>
/// <returns></returns>
IEnumerator LoadSubAssetAsync<T>(string assetPath, string subName, AssetBundle bundle,
LoaderDelegate.OnAssetLoaded<T> onAssetLoaded)
where T : Object;
```

- Batch sub-resource asset loading (sync|async)

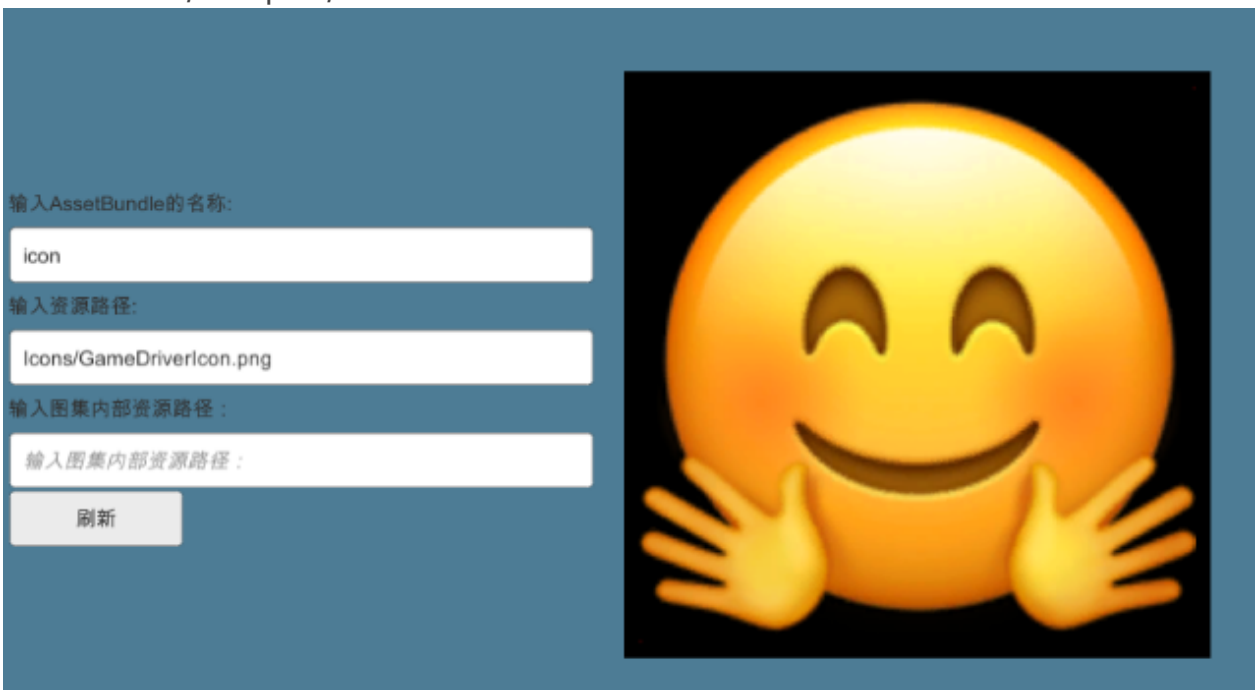
```
/// <summary>
/// Load all sub-resources (sprites, prefabs, etc.)
/// 加载子资源(子图、预置等)
/// </summary>
/// <param name="path"></param>
/// <param name="type"></param>
/// <param name="bundle"></param>
/// <returns></returns>
Object[] LoadSubAssetsSync(string path, Type type, AssetBundle bundle);

/// <summary>
/// Load all sub-resources (sprites, prefabs, etc.)
/// 加载子资源(子图、预置等)
/// </summary>
/// <param name="path"></param>
/// <param name="bundle"></param>
/// <typeparam name="T"></typeparam>
/// <returns></returns>
T[] LoadSubAssetsSync<T>(string path, AssetBundle bundle) where T : Object;

/// <summary>
/// Load sub-resources (sprites, prefabs, etc.)
/// 加载子资源(子图、预置等)
/// </summary>
/// <param name="path"></param>
/// <param name="subNames"></param>
/// <param name="bundle"></param>
/// <returns></returns>
Object[] LoadSubAssetsSync(string path, string[] subNames, AssetBundle bundle);

/// <summary>
/// Load sub-resources (sprites, prefabs, etc.)
/// 加载子资源(子图、预置等)
/// </summary>
/// <param name="path"></param>
/// <param name="subNames"></param>
/// <param name="type"></param>
/// <param name="ab"></param>
/// <returns></returns>
Object[] LoadSubAssetsSync(string path, string[] subNames, Type type, AssetBundle ab);

/// <summary>
/// Load sub-resources (sprites, prefabs, etc.)
/// 加载子资源(子图、预置等)
/// </summary>
/// <param name="path"></param>
/// <param name="subNames"></param>
/// <param name="bundle"></param>
/// <typeparam name="T"></typeparam>
/// <returns></returns>
T[] LoadSubAssetsSync<T>(string path, string[] subNames, AssetBundle bundle) where T : Object;

/// <summary>
/// Load sub-resources (sprites, prefabs, etc.) from bundles asynchronously
/// 异步从资源包中加载加载子资源(子图、预置等)
/// </summary>
/// <param name="assetPath">资源路径</param>
/// <param name="buundle">资源包</param>
/// <param name="onAssetsLoaded">Callback(回调)</param>
/// <typeparam name="T">Resource type(资源类型)</typeparam>
```

```
/// <returns></returns>
IEnumerator LoadSubAssetsAsync<T>(string assetPath, AssetBundle buundle,
LoaderDelegate.OnMultiAssetLoaded<T> onAssetsLoaded)
where T : Object;
```

- For more usage, please refer to the example, API or source code.

2.3 Example

GameDriver/Samples/Loader



3. Internationalization (i18n) - Lightweight Internationalization Solution

- **JLGames.GameDriver.Actions.i18n** provides full functional support for the internationalization module.
- BabelEdit is recommended for authoring management datasets.
- Use internationalization (i18n) in three steps:
    1. Prepare the data.
    2. Initialize (**Initialize by registering information** or **Initialize by configuring assets**).
    3. Use.

3.1 Preparation

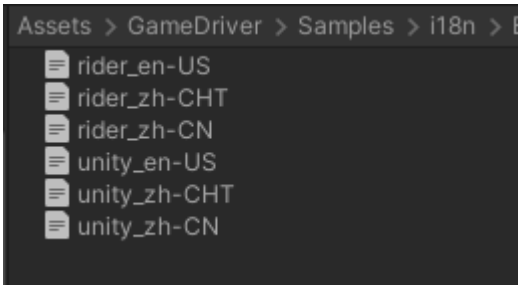
Understand the internationalization requirements of the project and prepare data files.

3.1.1 Clarify the types of international languages that the project needs to support

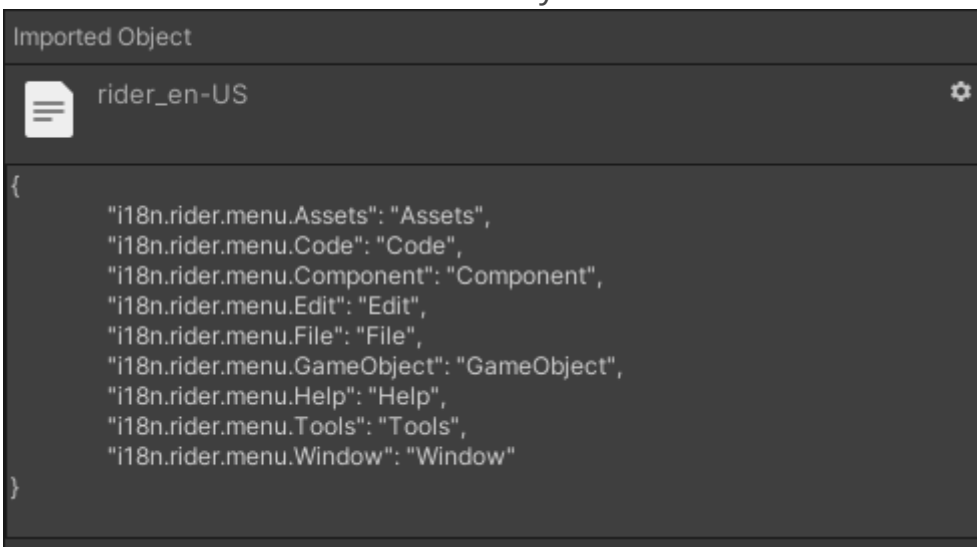
Here are **English[English]**, **Chinese[Simplified]**, **Chinese[Traditional]** as examples.

3.1.2 Prepare internationalization data files

Multiple file sets are supported, here are two file sets **unity**, **rider** for example.



The data format is in the form of Key-Value:



The current test dataset is authored using Jetbrains' [BabelEdit](https://www.codeandweb.com/babeledit), which is a nice tool.

3.2 Initialize by registering internationalization informations

3.2.1 Register languages with the registry.

Call the RegisterLang function in the I18NRegister instance:

```
/// <summary>
/// Register language
/// 注册语言
/// </summary>
/// <param name="langName">Language name 语言名称</param>
/// <param name="langSuffix">Language file suffix 语言文件后缀</param>
/// <param name="default">Set as default? 是否设置为默认</param>
public void RegisterLang(string langName, string langSuffix, bool @default = false)
{
    m_LangSuffixMap[langName] = langSuffix;
    if (@default) m_DefaultLang = langName;
}
```

- langName as langSuffix suggests.
- default=true sets the current language as the default language.

3.2.1 Loading information to the registry registration file set

Call the RegisterFile function in the I18NRegister instance:

```
/// <summary>
/// Register loading info of the file set.
/// 注册文件集加载信息
/// </summary>
/// <param name="fileKey">file set key name 文件映射Key</param>
/// <param name="fileBundle">bundle of the file set 文件集所在bundle</param>
/// <param name="filePath"></param>
/// <param name="default"></param>
public void RegisterFile(string fileKey, string fileBundle, string filePath, bool @default = false)
{
    m_FilePathMap[fileKey] = new RegisterFileInfo(fileKey, fileBundle, filePath);
    if (@default) m_DefaultFileKey = fileKey;
}
```

- fileKey is used to identify the currently set fileset.
- default=truej Set the current fileset as the default fileset.

3.2.3 According to the registry, load data into the manager

Call the LoadData function in the II18NManager instance:
`I18NManagerShared.Manager.LoadData();`

```
/// <summary>
/// Reload data according to the settings in Register
/// 根据Register中设置重新加载数据
/// </summary>
void LoadData();
```

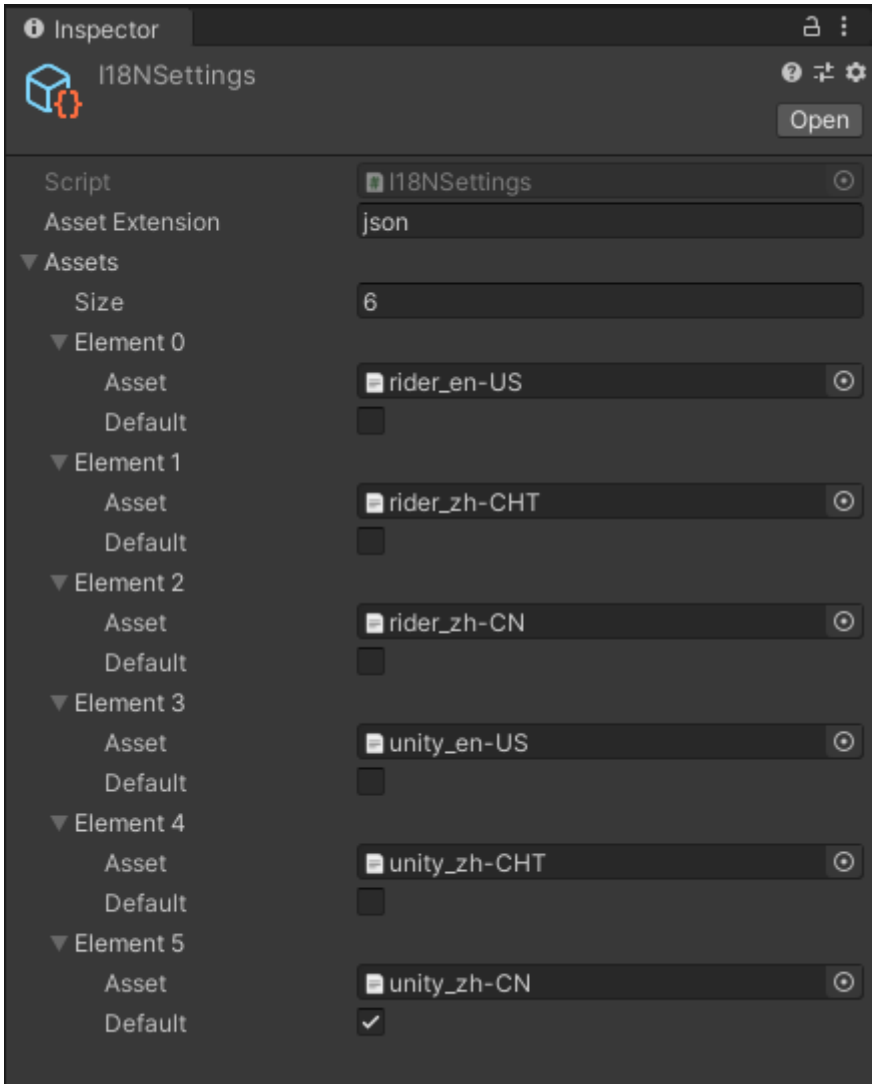
NOTE: If you have performed load data in the past, you should clear the old data first:
`I18NManagerShared.Manager.ClearData();`

```
/// <summary>
/// Clear all
/// 清除全部
/// </summary>
void ClearData();
```

3.3 Initialize by configuring asset

3.3.1 Generate configuration assets and configure information

1. Select the generation directory of the configuration assets, click the menu “Tools -> GameDriver -> Project -> Gen I18NSettings”.
2. The generated asset file is named “I18NSettings.asset”.
3. Set the extension name of the fileset: Asset Extension. “json” is suggested here.
4. Add the fileset to the Assets configuration list. Or set one of them as default (**only the last default takes effect**).



3.3.3 Initializing data by configuring assets

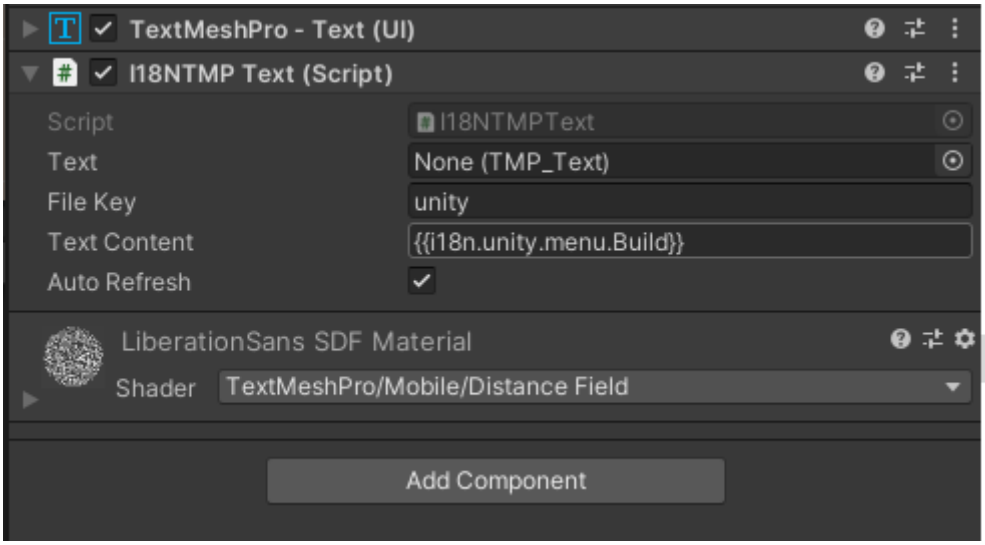
Call the LoadDataFromSettings function in the II18NManager instance:
`I18NManagerShared.Manager.LoadDataFromSettings(settings);`

```
/// <summary>
/// Load data from configuration assets
/// 从配置资产中加载数据
/// </summary>
/// <param name="settings"></param>
void LoadDataFromSettings(I18NSettings settings);
```

3.4 Use

3.4.1 Use the I18NTMPText component to add internationalization support to TMP_Text.

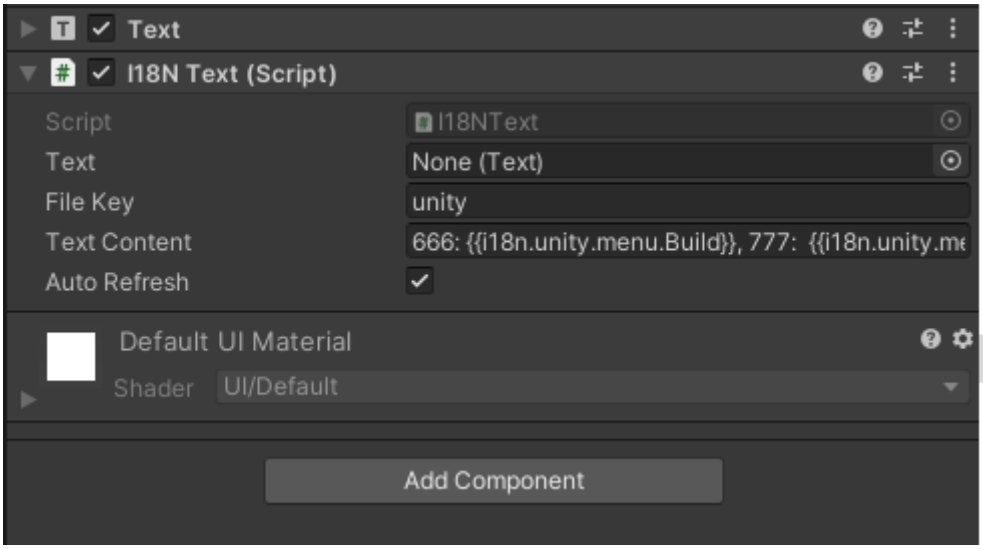
Add I18NTMPText component to TMP_Text node



- File Key is the fileKey parameter when registering a file set.
- Auto Refresh When checked, the manager will refresh itself after reloading the data.

3.4.2 Use the I18NText component to add internationalization support to Text.

Add I18NText component to Text node



- File Key is the fileKey parameter when registering a file set.
- Auto Refresh When checked, the manager will refresh itself after reloading the data.

3.4.3 Reading internationalized data in code.

I18NManagerShared.Manager.GetValue(id, fileKey, lang);

```
/// <summary>
/// Get data, use default language and default file
/// 取数据，使用默认语言，默认文件
/// </summary>
/// <param name="id"></param>
/// <returns></returns>
string GetValue(string id);

/// <summary>
/// Get data, use default language
/// 取数据，使用默认语言
/// </summary>
/// <param name="id"></param>
/// <param name="fileKey"></param>
/// <returns></returns>
string GetValue(string id, string fileKey);

/// <summary>
/// Get data
/// 取数据
/// </summary>
/// <param name="id"></param>
/// <param name="fileKey"></param>
/// <param name="langName"></param>
/// <returns></returns>
string GetValue(string id, string fileKey, string langName);
```

3.4.4 Internationalize the text content in the code.

Support script format in the file content: `
I18NManagerShared.Manager.GetContent(content, fileKey, lang);

```
/// <summary>
/// Get Internationalized Content, use default language and default file
/// 获取国际化后的内容，使用默认语言，默认文件
/// </summary>
/// <param name="content"></param>
/// <returns></returns>
string GetContent(string content);

/// <summary>
/// Get Internationalized Content, use default language
/// 获取国际化后的内容，使用默认语言
/// </summary>
/// <param name="content"></param>
/// <param name="fileKey"></param>
/// <returns></returns>
string GetContent(string content, string fileKey);

/// <summary>
/// Get Internationalized Content
/// 获取国际化后的内容
/// </summary>
/// <param name="content"></param>
/// <param name="fileKey"></param>
/// <param name="langName"></param>
/// <returns></returns>
string GetContent(string content, string fileKey, string langName);
```

3.5 Example

GameDriver/Samples/i18n

4. Network management extension module (NetManager)

Easy-to-use network link management module

- **JLGames.GameDriver.CSharp.Net** provides a C# package for Socket API, which enables developers to develop protocols such as TCP, UDP,

HTTP(HTTPS), WebSocket(WebSockets) on the same set of APIs.

- **JLGames.GameDriver.CSharp.Net** also provides support and extension for data packing and unpacking.
- Management of link references is available in **JLGames.GameDriver.Games.NetManager**.
- Extended support for stand-alone servers in **JLGames.GameDriver.Games.NetManager.Virtual**, which can simulate the behavior of server protocol capture and logic processing without an actual network connection.

4.1 Use of VirtualServer

4.1.1 Instantiate and add to NetManager management

```
var server = new VirtualServer("server", "none");
NetManager.Shared.Register(server, true);
```

```
/// <summary>
/// Add the instance of INetEntity to the management
/// Currently, instance management of INetClient and INetServer is supported.
/// 把INetEntity的实例加入到管理中
/// 当前支持INetClient与INetServer的实例管理
/// </summary>
/// <param name="entity"></param>
/// <param name="default"></param>
public void Register(INetEntity entity, bool @default = false)
{
    if (null == entity) return;
    if (entity is INetClient)
        m_ClientCache.RegisterEntity(entity as INetClient, @default);
    if (entity is INetServer)
        m_ServerCache.RegisterEntity(entity as INetServer, @default);
}
```

Objects added to NetManager can be obtained by name.

4.1.2 Registering extensions

```
server.RegisterExtension(VirtualReqIds.Id0,
  OnRequestHandler);
```

```
/// <summary>
/// Register extension
/// 注册扩展
/// </summary>
/// <param name="protoId"></param>
/// <param name="onProtoRequestHandler"></param>
void RegisterExtension(string protoId, VirtualServerDelegate.OnProtoRequestHandler onProtoRequestHandler);
```

- protoId: Protocol ID, defined by the developer.
- onProtoRequestHandler: Response function, corresponding to the response management logic of the protocol number.

4.1.3 Protocol response

Response delegate declaration:
`VirtualServerDelegate.OnProtoRequestHandler`

```
/// <summary>
/// Delegate declaration for protocol handle function
/// 协议响应函数的委托声明
/// </summary>
/// <param name="protoId"></param>
/// <param name="data"></param>
public delegate void OnProtoRequestHandler(IVirtualServer server, string protoId, object data);
```

- server: IVirtualServer instance, which is mainly used in the function to send broadcast notifications of protocol results and data.
- protoId: protocol number
- data: the data sent when the protocol requests

4.1.4 Message Push

- The INotifyServer interface contains function declarations for message push.

```
public interface INotifyServer
{
    /// <summary>
    /// Message notify to client
    /// 向客户端发送消息推送
    /// </summary>
    /// <param name="protoId"></param>
    /// <param name="data"></param>
    void NotifyToClient(string protoId, object data = null);
}
```

- The IMaterialNotifyServer interface contains function declarations for pushing messages related to material data.

```
public interface IMaterialNotifyServer
{
    /// <summary>
    /// Notify Material Update - Offset
    /// 通知材料更新 - 偏移
    /// </summary>
    /// <param name="offset"></param>
    void NotifyMaterialEntryUpdate(DataOffset offset);

    /// <summary>
    /// Notify Material Update - Number
    /// 通知材料更新 - 数量
    /// </summary>
    /// <param name="num"></param>
    void NotifyMaterialEntryUpdate(DataNum num);

    /// <summary>
    /// Notify Material Update - Number and Offset
    /// 通知材料更新 - 偏移+数量
    /// </summary>
    /// <param name="notify"></param>
    void NotifyMaterialEntryUpdate(UserNotifyData notify);

    /// <summary>
    /// Notify Multiple Material Update - Offset
    /// 通知材料更新 - 偏移
    /// </summary>
    /// <param name="offsetArr"></param>
    void NotifyMaterialEntryUpdate(DataOffset[] offsetArr);

    /// <summary>
    /// Notify Multiple Material Update - Number
    /// 通知材料更新 - 数量
    /// </summary>
    /// <param name="numArr"></param>
    void NotifyMaterialEntryUpdate(DataNum[] numArr);

    /// <summary>
    /// Notify Multiple Material Update - Number and Offset
    /// 通知材料更新 - 偏移+数量
    /// </summary>
    /// <param name="notifyArr"></param>
    void NotifyMaterialEntryUpdate(UserNotifyData[] notifyArr);
}
```

- The server instance in the response delegate `VirtualServerDelegate.OnProtoRequestHandler` has implemented the INotifyServer and IMaterialNotifyServer interfaces, and realizes the data broadcasting function by calling the message push function.

4.2 Use of VirtualClient

4.2.1 Instantiate and add to NetManager management.

```
var client = new VirtualClient("client", "none");
NetManager.Shared.Register(client, true);
```

```
/// <summary>
/// Add the instance of INetEntity to the management
/// Currently, instance management of INetClient and INetServer is supported.
/// 把INetEntity的实例加入到管理中
/// 当前支持INetClient与INetServer的实例管理
/// </summary>
/// <param name="entity"></param>
/// <param name="default"></param>
public void Register(INetEntity entity, bool @default = false)
{
    if (null == entity) return;
    if (entity is INetClient)
        m_ClientCache.RegisterEntity(entity as INetClient, @default);
    if (entity is INetServer)
        m_ServerCache.RegisterEntity(entity as INetServer, @default);
}
```

Objects added to NetManager can be obtained by name.

4.2.2 Connecting to the server

```
var serverProxy = NetManager.Shared.
  GetServer<IVirtualServer>("server") as IVirtualServerProxy;
client.Connect(serverProxy);
```

```
/// <summary>
/// Connect to server
/// 连接Server
/// </summary>
void Connect(IVirtualServerProxy server);

/// <summary>
/// Disconnect from server
/// 断开Server连接
/// </summary>
void Disconnect();

/// <summary>
/// Reconnect to server
/// 重连
/// </summary>
void Reconnect(IVirtualServerProxy server);
```

- Get the server proxy (actually the VirtualServer object) from the manager.
- The connection proxy function is Connect.
- The function to cancel the proxy connection is Disconnect.

4.2.3 Monitor request response, monitor message push

The monitoring function is built under the event mechanism:

```
client.AddEventListener(VirtualClientEvents.EventResponse,
  OnClientResponse);
client.AddEventListener(VirtualClientEvents.EventNotify,
  OnClientNotify);
```

```
/// <summary>
/// Received protocol response event
/// 收到协议响应事件
/// Data Foramt: VirtualClientResponse
/// 数据格式：VirtualServerResponse
/// </summary>
public const string EventResponse = "VirtualClientEvents.EventResponse";

/// <summary>
/// Received protocol notify event
/// 收到协议通知事件
/// Data Foramt: VirtualClientResponse
/// 数据格式：VirtualClientResponse
/// </summary>
public const string EventNotify = "VirtualClientEvents.EventNotify";
```

4.2.4 Send message request

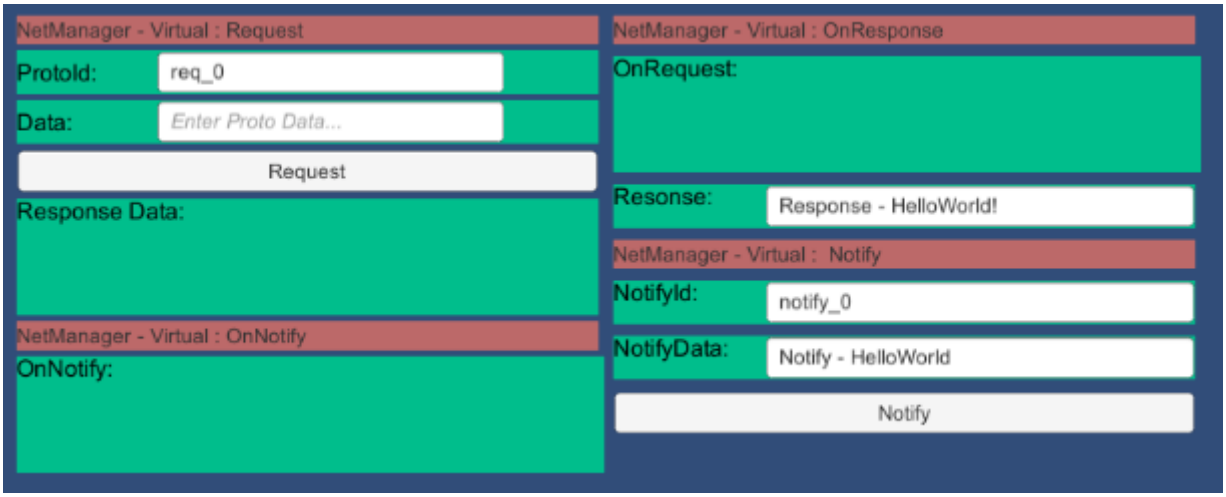
Use the Request function under the interface IVirtualClient to send a message request to the server.

```
NetManager.Shared.GetClient<IVirtualClient>("client")
  .Request(protoId, m_InputRequestData.text.Trim());
```

```
/// <summary>
/// Send a protocol request to the server
/// 向Server发送协议请求
/// </summary>
/// <param name="protoId"></param>
/// <param name="data"></param>
void Request(string protoId, object data);
```

4.3 Example

GameDriver/Samples/NetManager



5. Audio Manager (AudioManager)

The perfect audio management module supports scene and UI music sound effects, and removes the dependency of Assetbundle.

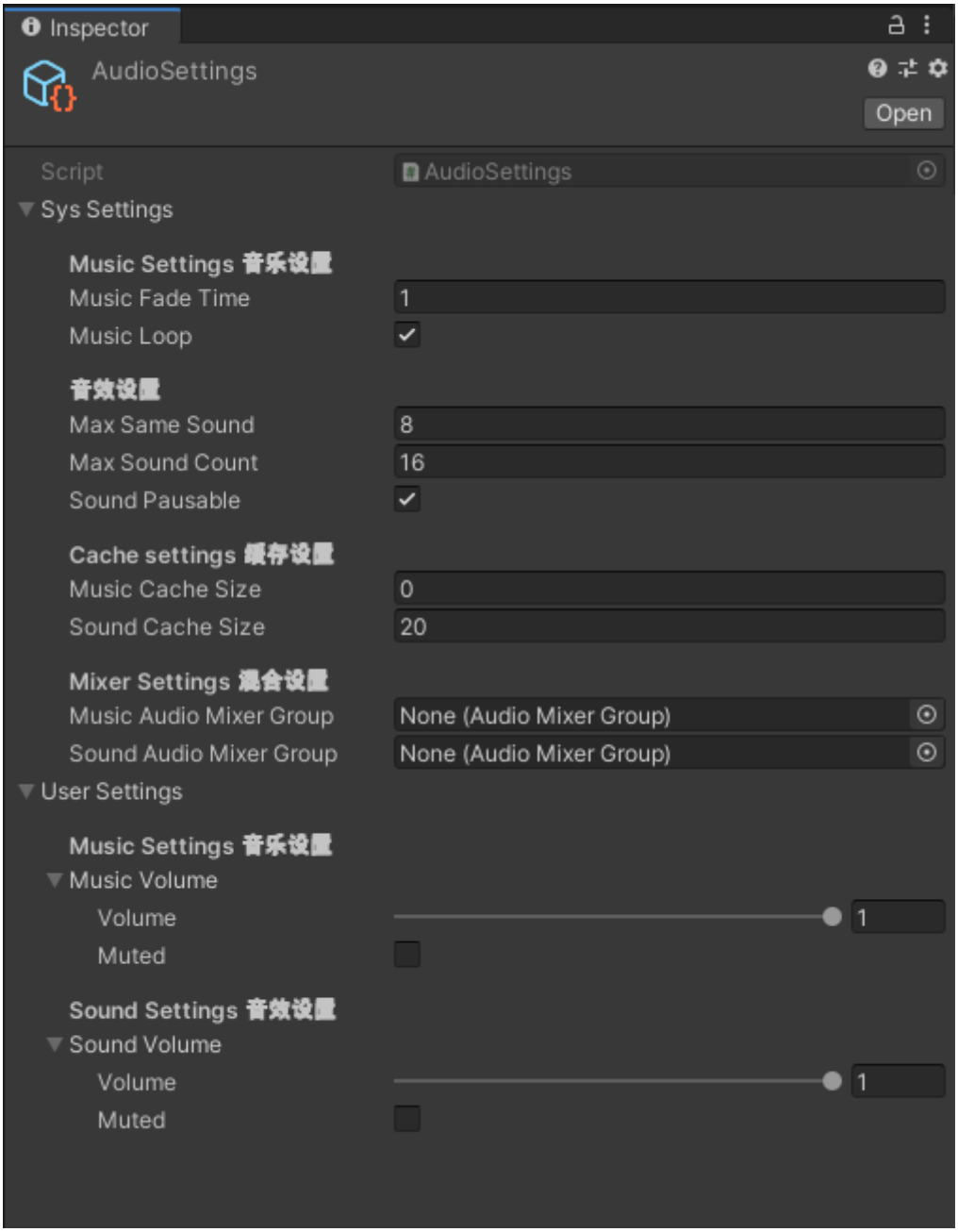
- **JLGames.GameDriver.Actions.Audio** provides full functional support for audio management.
- Audio loading uses the default loader (Loader), but also supports custom loader.
- Use Ico method to inject audio resource information.
- There is no need to directly reference audio resources, which realizes the separation of resource usage and resource packaging.
- Support cache settings to reduce loading consumption.

5.1 Initialization

5.1.1 Generate configuration assets

Execute menu “Tools -> GameDriver -> Project -> Gen AudioSettings”.
The AudioSettings.asset (renameable) file will be generated under the project

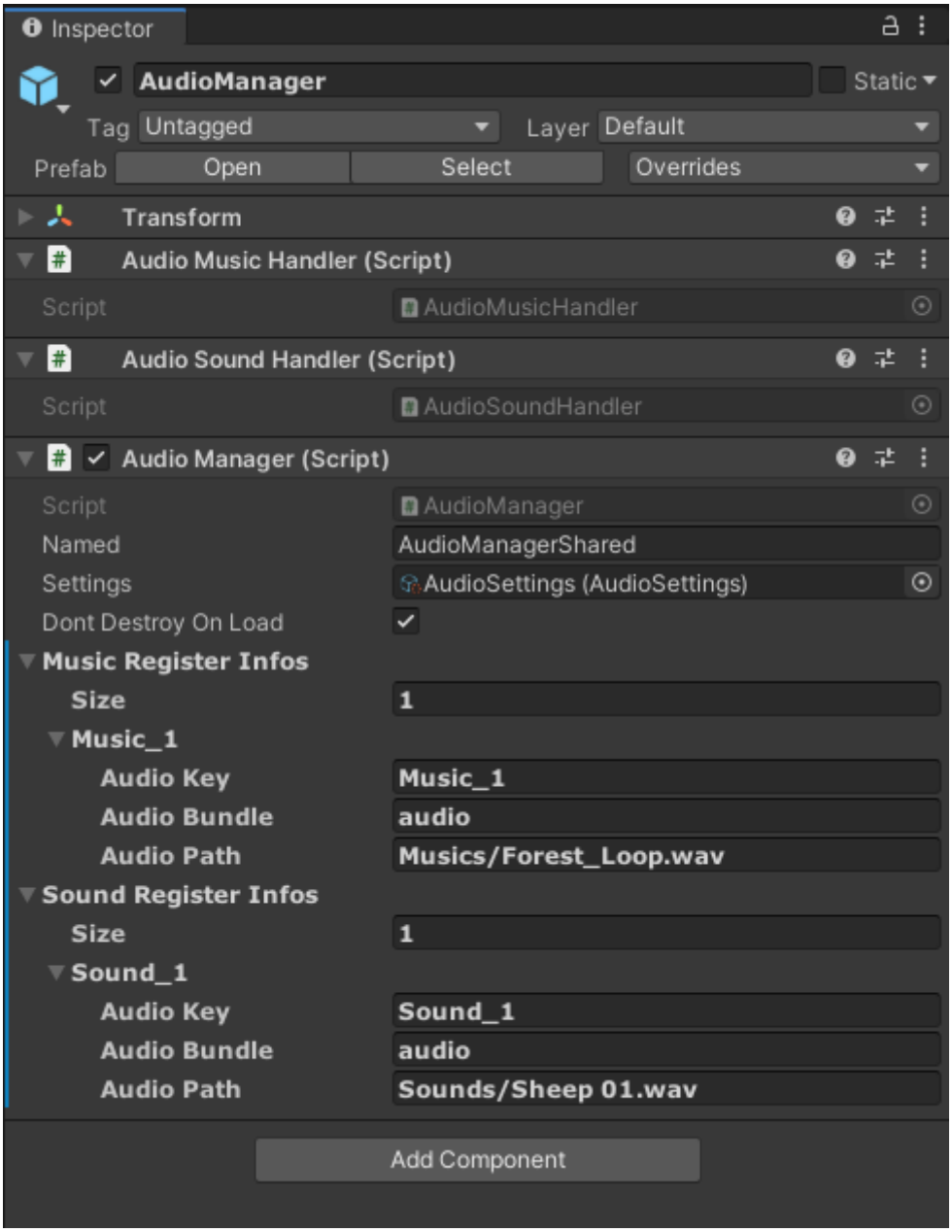
Assets/Resources.



- SysSettings are system-level settings and cannot be modified by users.
- UserSettings are user-level settings, open to AP modification. I

5.1.2 Setting up management objects

Copy GameDriver/Assets/AudioManager/Prefabs/AudioManager.prefab to the project initialization scene, and re-associate AudioSettings.asset



- Dont Destroy OnLoad whether to save the object when the scene is destroyed
- Music Register Infos Music information preprocessing registry
- Sound Register Infos Sound effect information preprocessing registry

5.1.3 Set the audio loader:

```
AudioManagerPool.Shared.SetLoaderAdapter(new AudioLoader());
```

If you need to customize the loading, you can implement the IAudioLoader interface.

5.2 Use

5.2.1 Register audio information (optional)

Audio information can be registered in the script using the AudioManager function

```
/// <summary>
/// Register music info
/// 注册音乐信息
/// </summary>
/// <param name="key"></param>
/// <param name="bundleName"></param>
/// <param name="path"></param>
void RegisterMusic(string key, string bundleName, string path);

/// <summary>
/// Clear music register
/// 清理音乐注册表
/// </summary>
void ClearMusicRegister();
```

```
/// <summary>
/// Register sound info.
/// 注册音效信息
/// </summary>
/// <param name="key"></param>
/// <param name="bundleName"></param>
/// <param name="path"></param>
void RegisterSound(string key, string bundleName, string path);

/// <summary>
/// Clear sound register
/// 清理音效注册表
/// </summary>
void ClearSoundRegister();
```

5.2.2 Play music, play sound effects

- Functions starting with SwitchMusic in AudioManger provide support for playing music.
- Functions starting with PlaySound in AudioManager provide support for playing sound effects.
- The playback of music sound effects supports direct playback without registration.

5.2.3 Audio Settings

- AudioManager supports music mute and volume settings.

```
/// <summary>
/// The name of the music playing
/// 播放中的音乐名
/// </summary>
string PlayingMusicName { get; }

/// <summary>
/// Set music volume
/// 设置音乐音量
/// </summary>
/// <param name="volume"></param>
/// <param name="refresh"></param>
void SetMusicVolume(float volume, bool refresh = true);
```

- AudioManager supports audio mute and volume settings.

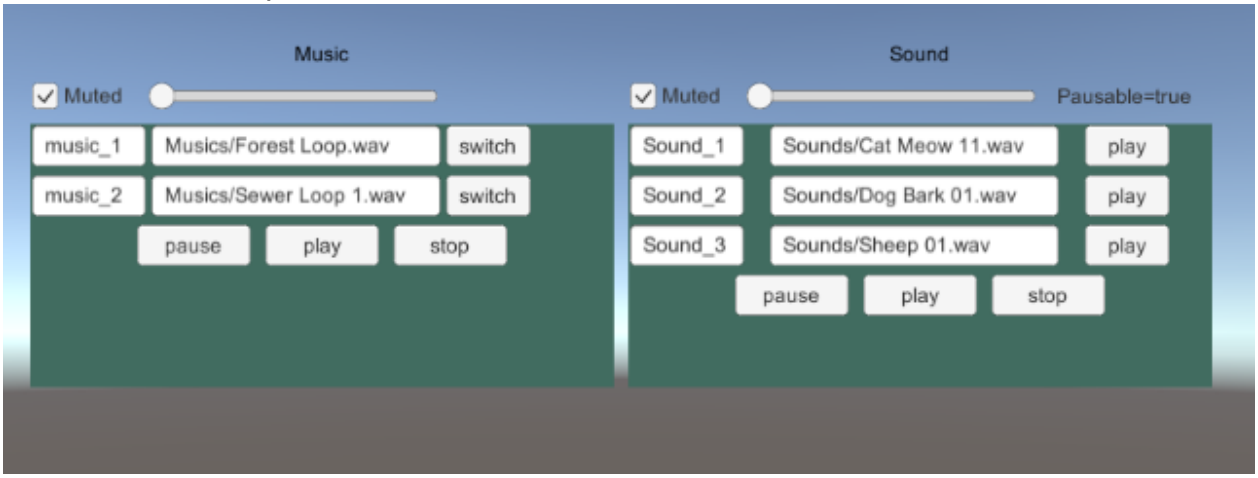
```
/// <summary>
/// Set sound volume
/// 设置音效音量
/// </summary>
/// <param name="volume"></param>
/// <param name="refresh"></param>
void SetSoundVolume(float volume, bool refresh = true);

/// <summary>
/// Set sound mute
/// 设置音效静音
/// </summary>
/// <param name="muted"></param>
/// <param name="refresh"></param>
void SetSoundMuted(bool muted, bool refresh = true);
```

5.2.4 It is recommended to use packaged components, such as:

AudioDemoMusicNode and AudioDemoSoundNode components in the example.

5.3 Example

GameDriver/Samples/Audio



6. Panel Manager (PanelManager)

- **JLGames.GameDriver.Actions.Layer** provides container level management.
- **JLGames.GameDriver.Games.PanelManager** provides panel management functions.
- The usage process of the panel management module: registration -> display -> close
- If the cached panel is not destroyed, it is recommended to move the panel off the screen.

6.1 Registration Information

6.1.1 Registration layer container information

```
PanelManagerShared.Manager.Register.RegisterLayer(containerName
  tranContainer, isDefault);
```

```
/// <summary>
/// Register panel container layer
/// 注册面板容器
/// </summary>
/// <param name="layerName"></param>
/// <param name="layer"></param>
/// <param name="default"></param>
void RegisterLayer(string layerName, Transform layer, bool @default = false);
```

- layerName: Specify a name for the registered layer information, regardless of the layer's name attribute.
- layer: The Transform component reference of the layer.
- default: Whether to set to default, if true, when opening the panel without specifying a layer name, the default layer is selected.

6.1.2 Registering background processing information

```
PanelManagerShared.Manager.Register.RegisterBackground(backgrou
  backgroundOrigin, backgroundScript);
```

```
/// <summary>
/// Register background meta object
/// 注册背景元预制件
/// </summary>
/// <param name="key"></param>
/// <param name="origin"></param>
/// <param name="script"></param>
void RegisterBackground(string key, GameObject origin, string script);
```

- key: Specify a key for the registered background information to identify the uniqueness of the information.
- origin: The prefab for the background, when the panel is shown, the cloned prefab is added to the bottommost layer of the panel container.
- script: The class name (including namespace) of the script component that handles the background logic.

6.1.3 Register animation information

```
PanelManagerShared.Manager.Register.RegisterAnimator(animKey, a
```

```
/// <summary>
/// Register RuntimeAnimatorController
/// 注册动画控制器
/// </summary>
/// <param name="key"></param>
/// <param name="animator"></param>
void RegisterAnimator(string key, RuntimeAnimatorController animator);
```

- key: Specify a Key for the registered animation information, which is used to identify the uniqueness of the information.
- animator: Animate the RuntimeAnimatorController component.

6.1.4 Setting the base container for the panel

```
PanelManagerShared.Manager.Register.RegisterPanelContainer(cont
```

```
/// <summary>
/// Register panel instance meta container
/// 注册实例容器
/// </summary>
/// <param name="container"></param>
void RegisterPanelContainer(GameObject container);
```

- container: The prefab of the panel container. When the panel is displayed, the cloned prefab is added to the display layer as the panel root node, and the actual content task child node of the panel is added to the panel container.

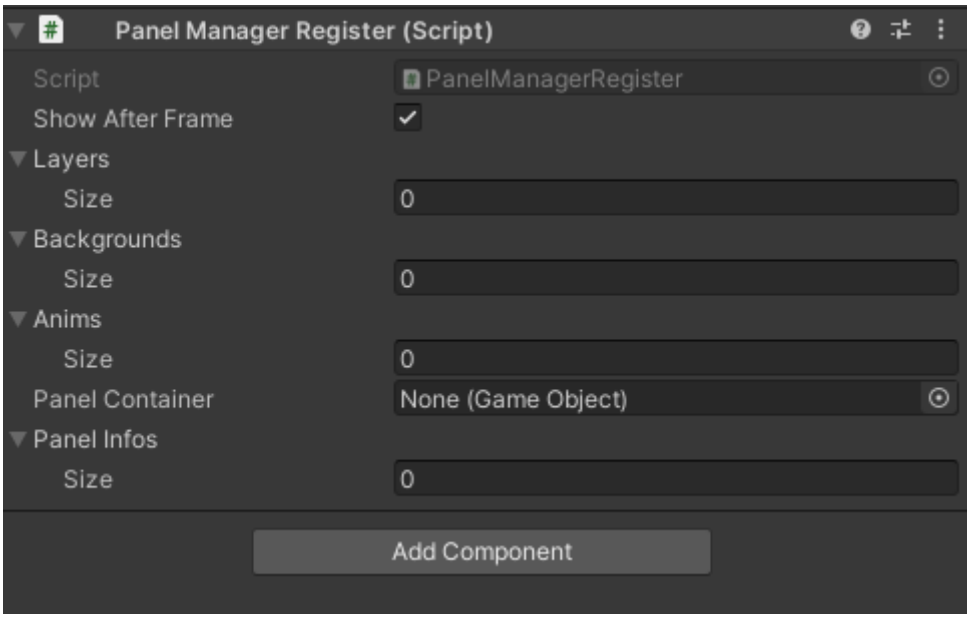
6.1.5 Registration panel information

```
PanelManagerShared.Manager.Register.RegisterPanelInfo(panelId,
  settings, maxDisplayNum, extendType);
```

```
/// <summary>
/// Register panel info
/// 注册面板信息
/// </summary>
/// <param name="info"></param>
void RegisterPanelInfo(IPanelInfo info);

/// <summary>
/// Register panel info
/// 注册面板信息
/// </summary>
/// <param name="id"></param>
/// <param name="bundleName"></param>
/// <param name="assetPath"></param>
void RegisterPanelInfo(string id, string bundleName, string assetPath);

/// <summary>
/// Register panel info
/// 注册面板信息
/// </summary>
/// <param name="id"></param>
/// <param name="settings"></param>
/// <param name="maxDisplayNum"></param>
/// <param name="extendType">业务扩展用面板类型</param>
void RegisterPanelInfo(string id, IPanelSettings settings, int maxDisplayNum, int extendType);
```

- Panel information is finally saved as an instance of IPanelInfo for search and display.
- When the project is used, the configuration information of the project panel should be read and registered.

6.1.6 Using the PanelManagerRegister component, you can handle the registration function of layer container, background, animation and panel information at the same time



6.1.7 Note

Background registration and **Animation registration optional**, only register when there are relevant requirements in the panel information.

6.2 Display Panel

6.2.1 To pop up a panel by panel id, you can select the specified panel layer container.

```
PanelManagerShared.Manager.ShowPanel(panelId);
```

```
/// <summary>
/// Show a panel.
/// 打开面板
/// </summary>
/// <param name="panelId">Panel Id<br/>面板的配置Id</param>
/// <param name="layerName">The container layer name corresponding to the panel to be displayed<br/>展示时指定的容器名称</param>
IPanelInstance ShowPanel(string panelId, string layerName = null);
```

6.2.2 Pop up the panel through the panel id, and pass in the parameter object, you can choose to specify the panel layer container.

```
PanelManagerShared.Manager.ShowPanel(panelId, panelParams);
```

```
/// <summary>
/// Show a panel.
/// 打开面板
/// </summary>
/// <param name="panelId">Panel Id<br/>面板的配置Id</param>
/// <param name="params">Parameters injected into IParamsPanel<br/>展示时IParamsPanel注入的参数对象</param>
/// <param name="layerName">The container layer name corresponding to the panel to be displayed<br/>展示时指定的容器名称</param>
IPanelInstance ShowPanel(string panelId, object @params, string layerName = null);
```

6.3 Close Panel

6.3.1 Close the panel by the panel id.

You can choose to close the default, close the first display, and close the last display through the panel id:

```
/// <summary>
/// Close panel by panel id
/// 关闭面板
/// </summary>
/// <param name="panelId">面板的配置Id</param>
void ClosePanel(string panelId);

/// <summary>
/// Close the first panel by panel id
/// 关闭面板
/// </summary>
/// <param name="panelId">面板的配置Id</param>
void CloseFirstPanel(string panelId);

/// <summary>
/// Close the last panel by panel id
/// 关闭面板
/// </summary>
/// <param name="panelId">面板的配置Id</param>
void CloseLastPanel(string panelId);
```

6.3.2 Close the panel through instance information.

Close the panel accurately by providing the panel instance Id (instanceId), display view (view), and panel instance (IPanelInstance).

```
/// <summary>
/// Close the panel by instance id
/// 关闭面板
/// </summary>
/// <param name="instanceId">面板的实例Id</param>
void CloseInstacnePanel(string instanceId);

/// <summary>
/// Close the panel by view instance
/// 关闭面板
/// </summary>
/// <param name="view">面板显示视图</param>
void CloseInstacnePanel(GameObject view);

/// <summary>
/// Close the panel by instance
/// 关闭面板
/// </summary>
/// <param name="instance">面板的实例对象</param>
void CloseInstacnePanel(IPanelInstance instance);
```

6.3.3 Close panels in batches.

You can choose to close all displayed panels, close all panels with a specified panel id, or close panels that match the matching function.

```
/// <summary>
/// Close all displayed panels
/// 关闭全部显示的面板
/// </summary>
void ClosePanels();

/// <summary>
/// Close all displayed panels by panel id
/// </summary>
/// <param name="panelId">面板的配置Id</param>
void ClosePanels(string panelId);

/// <summary>
/// Close multi displayed panels match function
/// </summary>
/// <param name="match"></param>
void ClosePanels(Predicate<IPanelInstance> match);
```

6.4 Advanced Applications

6.4.1 Custom Registrar

The registrar in IPanelManager can be customized, as long as the IPanelRegister interface is implemented.

```
/// <summary>
/// Set panel register.
/// 设置注册表
/// </summary>
/// <param name="register"></param>
void SetRegister(IPanelRegister register);
```

6.4.2 Custom Loaders

The loader in IPanelManager can be customized, as long as the IIPanelLoaderAdapter interface is implemented.

```
/// <summary>
/// Set asset loader adapter.
/// 设置加载器
/// </summary>
/// <param name="loader"></param>
void SetLoader(IPanelLoaderAdapter loader);
```

6.4.3 Set panel display timing

SetShowMoment in IPanelManger can set the panel display timing (immediate display | frame end display)

```
/// <summary>
/// Set show monent
/// 设置显示时机
/// </summary>
/// <param name="showAfterFrame"></param>
void SetShowMoment(bool showAfterFrame);
```

- show immediately When the code is executed, the loading process is performed immediately. Add panels to the display node as soon as the asset is ready.
- end of frame display When the code is executed, the loading process is performed immediately. When the resource preparation is complete, start the coroutine, wait for WaitForEndOfFrame, and add the panel to the display node.

6.4.4 IPanelSettings Description

The IPanelSettings instance is the interface of the panel configuration information saved when the panel is registered, including resource configuration, background configuration and animation configuration.

6.4.4.1 Resource Configuration

IPanelAssetSettings instance
Main configuration properties: BundleName, AssetPath, MainScriptName,

MainScriptParams

```
/// <summary>
/// Assetbundle name
/// Assetbundle名称
/// </summary>
string BundleName { get; }

/// <summary>
/// Asset path
/// 资源路径
/// </summary>
string AssetPath { get; }

/// <summary>
/// Main script full name
/// 主脚本完整类名
/// </summary>
string MainScriptName { get; }

/// <summary>
/// Json string params of script
/// 主脚本注入的参数字符串
/// </summary>
string MainScriptParams { get; }
```

- BundleName The name of the Assetbundle where the panel assets are located
- AssetPath The path of the panel asset in the Assetbundle
- MainScriptName If the panel has a main script, the full class name (including namespace) of the main script is returned. Returns null or an empty string without a main script. +MainScriptParams It takes effect when MainScriptName exists and implements the IParamsPanel interface, and is the incoming parameter of the function SetPanelStringParams.

6.4.4.2 Background configuration

IPanelBackgroundSettings instance

- basic parameters
  - OriginKey The key specified when the background information is registered
  - OriginInfo Find out the registration information through the key specified during the registration of the background information, and take the origin prefab.
  - Mode Four background modes are supported: None, Color, Image, Screenshot
    - None Dependency coefficient: none
    - Color Dependency coefficient: Color

        - Image Dependencies: ImageSprite, ImageBundle, ImagePath, BlurFactor
        - Screenshot Dependency factor: ScreenshotFactor
- Mode related parameters
    - Color [Valid when Mode=Color] background color value, used to set the background color value
    - ImageSprite [Valid when Mode=Image] Two cases: ImageSprite is not empty, ImageSprite is empty.
        - ImageSprite is not empty Ignore ImageBundle and ImagePath parameters, use ImageSprite to fill.
        - ImageSprite is empty Through the loader in Register, use ImageBundle and ImagePath to load Sprite fills.

    - ImageBundle and ImagePath [effective when Mode=Image] Used when the ImageSprite is empty, to load the Sprite resource.
    - BlurFactor [Valid when Mode=Image] Blur factor: [0,1]
    - ScreenshotFactor [Valid when Mode=Screenshot] Screenshot image blur factor

6.4.4.3 Animation Configuration

IPanelAnimSettings instance

```
/// <summary>
/// Open animator mapping Key
/// 展示时动画控制器映射Key
/// </summary>
string OpenKey { get; }

/// <summary>
/// Animation state in open animator mapping Key
/// 展示时指定动画State
/// </summary>
string OpenState { get; }

/// <summary>
/// Close animator mapping Key
/// 关闭时动画控制器映射Key
/// </summary>
string CloseKey { get; }

/// <summary>
/// Animation state in open animator mapping Key
/// 关闭时指定动画State
/// </summary>
string CloseState { get; }
```

- OpenKey The key used when animation information is registered, used to find animations

- OpenState The state name in the animation Animator, used to play the specified animation
- CloseKey The key used when animation information is registered, used to find animations
- CloseState The state name in the animation Animator, used to play the specified animation

6.4.5 Panel function extension

At this stage, there are 4 interfaces related to panel function expansion: IInitPanel, IParamsPanel, IRefreshPanel, IDisposePanel
NOTE: IShowPanel, IClosePanel are **deprecated**.

- IInitPanel Used for panel initialization, the call timing is after OnEnable

```
/// <summary>
/// Show panel execution timing: Awake > OnEnable > IShowPanel > IInitPanel > IParamsPanel > IRefreshPanel > Start
/// 展示面板执行时机：Awake > OnEnable > IShowPanel > IInitPanel > IParamsPanel > IRefreshPanel > Start
/// </summary>
public interface IInitPanel
{
    /// <summary>
    /// 初始化
    /// </summary>
    /// <param name="instance"></param>
    void InitPanel(IPanelInstance instance);
}
```

  +IParamsPanel Used for panel injection parameters.

```
/// <summary>
/// Show panel execution timing: Awake > OnEnable > IShowPanel > IInitPanel > IParamsPanel > IRefreshPanel > Start
/// 展示面板执行时机：Awake > OnEnable > IShowPanel > IInitPanel > IParamsPanel > IRefreshPanel > Start
/// </summary>
public interface IParamsPanel
{
    /// <summary>
    /// 设置面板参数
    /// </summary>
    /// <param name="instance"></param>
    /// <param name="str"></param>
    void SetPanelStringParams(IPanelInstance instance, string str);

    /// <summary>
    /// 设置面板参数
    /// </summary>
    /// <param name="instance"></param>
    /// <param name="params"></param>
    void SetPanelObjectParams(IPanelInstance instance, object @params);
}
```

    - SetPanelObjectParams Triggered when calling ShowPanel with parameters passed in.
    - SetPanelStringParams Triggering requires both of the following conditions:
        1. No parameters were passed when calling ShowPanel.
        2. When registering panel information, MainScriptParams in IPanelAssetSettings configuration is not empty. The incoming parameter is the value of MainScriptParams.

- IRefreshPanel Used when the panel refresh logic is not suitable for use in Start or Awake.

```
/// <summary>
/// Show panel execution timing: Awake > OnEnable > IShowPanel > IInitPanel > IParamsPanel > IRefreshPanel > Start
/// 展示面板执行时机：Awake > OnEnable > IShowPanel > IInitPanel > IParamsPanel > IRefreshPanel > Start
/// </summary>
public interface IRefreshPanel
{
    /// <summary>
    /// 刷新面板
    /// </summary>
    /// <param name="instance"></param>
    void RefreshPanel(IPanelInstance instance);
}
```

- IDisposePanel Called when the panel is about to be destroyed to perform the release behavior.

```
/// <summary>
/// Remove panel execution timing: IDisposePanel > IClosePanel > OnDisable > OnDestroy
/// 移除面板执行时机：IDisposePanel > IClosePanel > OnDisable > OnDestroy
/// </summary>
public interface IDisposePanel
{
    /// <summary>
    /// 销亡处理
    /// </summary>
    /// <param name="instance"></param>
    void DisposePanel(IPanelInstance instance);
}
```

6.5 Example

GameDriver/Samples/Panel

7. Service Framework (Service)

- **JLGames.GameDriver.CSharp.Service** provides the basic interface and behavior specification of the service framework.

- **JLGames.GameDriver.Actions.Service** provides service configuration assets for Unity (**not enabled**).
- **JLGames.GameDriver.Games.Service** provides common service functions.

7.1 Design Purpose

Game projects have some obvious characteristics: high degree of data coupling and strong system dependence.
So the entire game project can easily evolve into a **monolithic software**.
As there are more modules in the system, the monolithic architecture becomes more complex. In addition, the interleaving of new functions and modification of old functions affects each other and is ultimately difficult to manage.
In order to solve or reduce the impact of the above problems, combined with the idea of **service-oriented architecture(SOA)**, here is a solution and solve the following problems::

- Realize logical separation of business modules
  - Logical separation between different businesses
  - Logical separation of data and display under the same business

- Extended support for normalized business modules
- Unify the specification of cross-business behavior calls

7.2 Design Ideas

- How to divide the service and how to grasp the granularity
  1. All businesses in the range from “game login complete” to “game logout start” are regarded as service division objects.
  2. Divide service units on a regular basis. The division rules are as follows:
     - Priority is given to dividing the game business and obtaining a functional system.
     - The functional system is further divided according to data, display and management to obtain service units.
     - Data reading and writing are isolated through the interface.

  3. Design a functional open interface (read-only interface) for each service unit.
  4. The communication between service units adopts the event mechanism.

- According to the idea of IOC, extended business services are added to the service framework management in the form of injection.
- According to the design idea of the game engine, an interface function called at a fixed timing is added to the extended business service, which is used for initializing data, releasing data or special logic implementation inside the service.

- All extension services are managed by the framework, which is convenient for unified processing of initialization, update and release.

7.3 Interface description

7.3.1 Service Basics

- **IService** Only the class that implements the IService interface can be identified as a service and added to the framework management.
  The ServiceName property returns the service name, which is required to be **unique**.
- **IEventDispatcher** The timing management of the service depends on the event module, so the service implementation class is required to implement the IEventDispatcher interface.
- **ServiceBase** Implement some necessary functions of the service class, which can be used for inheritance.
- **IProgressingService** Provides a service interface for **progress update**.
  Triggers a progress update when the service instance throws the **ServiceEvents.OnServiceProcessing** event.
  The service instance requires the attribute functions of **total progress** and **current progress**.

```
/// <summary>
/// Progress metering interface
/// 进度计量接口
/// </summary>
public interface IProgressingService : IEventDispatcher
{
    /// <summary>
    /// Total progress
    /// 进度总量
    /// [0,int.Max)
    /// </summary>
    uint ProgressingLen { get; }

    /// <summary>
    /// Current amount of progress
    /// 进度当前量
    /// [0,Total]
    /// </summary>
    uint ProgressingCurrent { get; }
}
```

7.3.2 Service initialization related

Execution timing: IArgumentService -> IAWakableService -> IInitService

- **IArgumentService** Provides a service interface for parameter injection
  The completion of the function execution represents the end of the process

of **injecting parameters**.

```
/// <summary>
/// Data injectioin interface
/// 注入数据接口
/// </summary>
public interface IArgumentService
{
    /// <summary>
    /// Inject data.
    /// 注入数据
    /// </summary>
    /// <param name="args"></param>
    void InjectArgument(object[] args);
}
```

- **IAWakableService** A service interface that provides **activation** logic. The completion of the function execution represents the end of the **activation** process.

```
/// <summary>
/// Service Awaka interface
/// 服务激活接口
/// </summary>
public interface IAwakableService
{
    /// <summary>
    /// Awake service
    /// 激活Service
    /// Async is not allowed
    /// 不允许使用异步
    /// </summary>
    void Awake();
}
```

- **IInitService** A service interface that provides **initialization** logic. When the service instance throws the **ServiceEvents.OnServiceInited** event, it means the **initialization** process ends.

```
/// <summary>
/// Service initialization interface
/// 服务初始化接口
/// Only when the current interface is implemented and configured into ServiceConfig,
/// the init method will be executed during the initialization process
/// 只有实现了当前接口，并配置到ServiceConfig中时，在初始化过程中才会执行init方法
/// </summary>
public interface IInitService : IService, IEventDispatcher
{
    /// <summary>
    /// Whether the initialization has been completed
    /// 是否已经完成初始化
    /// </summary>
    bool IsInited { get; }

    /// <summary>
    /// Initialize base data
    /// 初始化基础数据
    /// </summary>
    void Init();
}
```

- **IInitDataService** A service interface that provides **data initialization** logic. When the service instance throws the **ServiceEvents.OnServiceDataInited** event, it means the **data initialization** process ends.

```
/// <summary>
/// Progress metering interface
/// 进度计量接口
/// </summary>
public interface IProgressingService : IEventDispatcher
{
    /// <summary>
    /// Total progress
    /// 进度总量
    /// [0,int.Max)
    /// </summary>
    uint ProgressingLen { get; }

    /// <summary>
    /// Current amount of progress
    /// 进度当前量
    /// [0,Total]
    /// </summary>
    uint ProgressingCurrent { get; }
}
```

7.3.3 Management scheduling related

- **IClearService** A service interface that provides **cleanup** logic.
  Mostly used for data reset, event removal and other logic.
  The completion of the function execution means the end of the cleanup process.

```
/// <summary>
/// Reset service interface, providing reset service to initialized state
/// 重置服务接口，提供把服务重置到初始化状态
/// </summary>
public interface IClearService
{
    /// <summary>
    /// reset
    /// 重置
    /// Clear events, clear timers, etc.
    /// 清除事件、清除计时器等
    /// </summary>
    void Clear();
}
```

- **ILoadDataService** Provides a service interface for data loading logic. When the service instance throws the **ServiceEvents.OnServiceLoaded**

event, it means the **data loading** process ends.

```
/// <summary>
/// Load data processing interface
/// 加载数据处理接口
/// </summary>
public interface ILoadDataService : IEventDispatcher
{
    /// <summary>
    /// Load Data
    /// 加载数据
    /// </summary>
    void LoadData();
}
```

- **ISaveDataService** Provides a service interface for **data storage** logic. When the service instance throws the **ServiceEvents.OnServiceSaved** event, it means the **data saving** process ends.

```
/// <summary>
/// Save data processing interface
/// 保存数据处理接口
/// </summary>
public interface ISaveDataService : IEventDispatcher
{
    /// <summary>
    /// Save data
    /// 保存数据
    /// </summary>
    void SaveData();
}
```

7.3.4 Event Description

7.3.4.1 Progress update related events

```
/// <summary>
/// A single service initializes the progress update event, which is dispatched by the service instance.
/// The service instance must be an implementation class of the IProgressingService interface
/// 单个服务初始化进度更新事件，由服务实例调度。服务实例必须为IProgressingService接口的实现类
/// </summary>
public const string OnServiceProcessing = "ServiceEvents.OnServiceProcessing";

/// <summary>
/// Service initialization process progress update
/// 服务初始化进程进度更新
/// </summary>
public const string OnInitializationProcessing = "ServiceEvents:OnInitializationProcessing";

/// <summary>
/// Service initialization process completed
/// 服务初始化进程完成
/// </summary>
public const string OnInitializationFinish = "ServiceEvents:OnInitializationFinish";
```

- **ServiceEvents.OnServiceProcessing**
  - Scheduling body: a service instance that implements the IProgressingService interface

    - Scheduling timing: when it is necessary to inform the listener of the update progress.
- **ServiceEvents.OnInitializationProcessing**
    - Scheduling subject: ServiceManager
    - Scheduling timing: During the initial execution period from StartInitalization to endCall, it is scheduled when three events of OnServiceProcessing, OnServiceInited, and OnServiceDataInited are captured.

- **ServiceEvents.OnInitializationFinish**
    - Scheduling subject: ServiceManager
    - Scheduling timing: when the call to StartInitalization ends, after the execution of endCall.

7.3.4.2 Service preparation related events

```
/// <summary>
/// Service argument injection result event, dispatched by ServiceManager.
/// Succ=true when the service implements the IArgumentService interface.
/// 服务参数注入结果事件，由ServiceManager调度。当服务实现IArgumentService接口时，Succ=true。
/// Event data format: ServiceResultEventData
/// 事件数据格式：ServiceResultData
/// </summary>
public const string OnServiceInjected = "ServiceEvents:OnServiceInjected";

/// <summary>
/// All service argument injection completion event
/// 全部服务参数注入完成事件
/// Event data format: null
/// 事件数据格式：null
/// </summary>
public const string OnServiceAllInjected = "ServiceEvents:OnServiceAllInjected";

/// <summary>
/// Service activation result event, dispatched by ServiceManager.
/// Succ=true when the service implements the IWakableService interface.
/// 服务激活结果事件，由ServiceManager调度。当服务实现IAwakableService接口时，Succ=true。
/// Event data format: ServiceResultEventData
/// 事件数据格式：ServiceResultData
/// </summary>
public const string OnServiceAwaked = "ServiceEvents:OnServiceAwaked";

/// <summary>
/// All service activation completion event
/// 全部服务激活完成事件
/// Event data format: null
/// 事件数据格式：null
/// </summary>
public const string OnServiceAllAwaked = "ServiceEvents:OnServiceAllAwaked";
```

Dispatching order: OnServiceInjected > OnServiceAllInjected > OnServiceAwaked > OnServiceAllAwaked

- **ServiceEvents.OnServiceInjected**
    - Scheduling subject: ServiceManager
    - Scheduling timing: during the call to StartInitalization

- **ServiceEvents.OnServiceAllInjected**

    - Scheduling subject: ServiceManager
    - Scheduling timing: During the call to StartInitalization, all services that implement IArgumentService are processed.
- **ServiceEvents.OnServiceAwaked**
    - Scheduling subject: ServiceManager
    - Scheduling timing: during the call to StartInitalization

- **ServiceEvents.OnServiceAllAwaked**
    - Scheduling subject: ServiceManager
    - Scheduling timing: During the call to StartInitalization, after all the services that implement IAWakableService are processed.

7.3.4.3 Service initialization related events

```
/// <summary>
/// Single service initialization start event, dispatched by ServiceManager
/// 单个服务初始化开始事件，由ServiceManager调度
/// Event data format: ServiceResultData
/// 事件数据格式：ServiceResultData
/// </summary>
public const string OnServiceInitStart = "ServiceEvents:OnServiceInitStart";

/// <summary>
/// A single service initialization complete event, which are dispatched by the service instance.
/// Re-dispatched after being captured by ServiceManager.
/// 单个服务初始化完成事件，由服务实例调度。被ServiceManager捕获后重新调度。
/// Event data format: {named:string}
/// 事件数据格式：{named:string}
/// </summary>
public const string OnServiceInited = "ServiceEvents:OnServiceInited";

/// <summary>
/// All service initialization complete event
/// 全部服务初始化完成事件
/// Event data format: len:int
/// 事件数据格式：len:int
/// </summary>
public const string OnServiceAllInited = "ServiceEvents:OnServiceAllInited";
```

```
/// <summary>
/// Single service data initialization start event, dispatched by ServiceManager
/// 单个服务数据初始化开始事件，由ServiceManager调度
/// Event data format: ServiceResultData
/// 事件数据格式：ServiceResultData
/// </summary>
public const string OnServiceDataInitStart = "ServiceEvents:OnServiceDataInitStart";

/// <summary>
/// A single service data initialization complete event, which are dispatched by the service instance.
/// Re-dispatched after being captured by ServiceManager.
/// 单个服务数据初始化完成事件，由服务实例调度。被ServiceManager捕获后重新调度。
/// Event data format: {named:string}
/// 事件数据格式：{named:string}
/// </summary>
public const string OnServiceDataInited = "ServiceEvents:OnServiceDataInited";

/// <summary>
/// All service data initialization complete event
/// 全部服务数据初始化完成事件
/// Event data format: len:int
/// 事件数据格式：len:int
/// </summary>
public const string OnServiceDataAllInited = "ServiceEvents:OnServiceDataAllInited";
```

Dispatching order: OnServiceInitStart > OnServiceInited > OnServiceAllInited > OnServiceDataInitStart > OnServiceDataInited > OnServiceDataAllInited

- **ServiceEvents.OnServiceInitStart**
  - Scheduling subject: ServiceManager
  - Scheduling timing: During the call to StartInitalization, process each service before the behavior of the IInitService interface.

- **ServiceEvents.OnServiceInited**
  - Scheduling subject: service instance, ServiceManager that implements IInitService
  - Scheduling timing: After the service instance that implements IInitService handles the Init behavior; ServiceManager captures the former event and reschedules it.

- **ServiceEvents.OnServiceAllInited**
  - Scheduling subject: ServiceManager
  - Scheduling timing: During the call to StartInitalization, after all the services that implement IInitService are processed.

- **ServiceEvents.OnServiceDataInitStart**
  - Scheduling subject: ServiceManager
  - Scheduling timing: During the call to StartInitalization, process each service before the behavior of the IInitDataService interface.

- **ServiceEvents.OnServiceDataInited**
  - Scheduling subject: service instance, ServiceManager that implements IInitDataService
  - Scheduling timing: After the service instance that implements IInitDataService handles the InitData behavior; ServiceManager captures the former event and reschedules it.

- **ServiceEvents.OnServiceDataAllInited**
  - Scheduling subject: ServiceManager
  - Scheduling timing: During the call to StartInitalization, after all the services that implement IInitDataService are processed.

7.3.4.4 Data loading related events

```
/// <summary>
/// A single  data service load data start event, dispatched by ServiceManager
/// Succ=true when the service implements the ILoadDataService interface.
/// 单个服务数据加载数据开始事件，由ServiceManager调度。当服务实现ILoadDataService接口时，Succ=true。
/// Event data format: ServiceResultData
/// 事件数据格式：ServiceResultData
/// </summary>
public const string OnServiceDataLoadStart = "ServiceEvents:OnServiceDataLoadStart";

/// <summary>
/// A single data service load data completion event, which are dispatched by the service instance.
/// Re-dispatched after being captured by ServiceManager.
/// 单个数据服务加载数据完成事件，由服务实例调度。被ServiceManager捕获后重新调度。
/// Event data format: {named:string}
/// 事件数据格式：{named:string}
/// </summary>
public const string OnServiceDataLoaded = "ServiceEvents:OnServiceDataLoaded";

/// <summary>
/// All data service loading data completion event
/// 全部数据服务加载数据完成事件
/// Event data format: len:int
/// 事件数据格式：len:int
/// </summary>
public const string OnServiceDataAllLoaded = "ServiceEvents:OnServiceDataAllLoaded";
```

- **ServiceEvents.OnServiceDataLoadStart**
  - Scheduling subject: ServiceManager
  - Scheduling timing: During the call to LoadServicesData, process each service before the behavior of the ILoadDataService interface.

- **ServiceEvents.OnServiceDataLoaded**
  - Scheduling subject: service instance, ServiceManager that implements ILoadDataService
  - Scheduling timing: After the service instance that implements ILoadDataService handles the LoadData behavior; ServiceManager catches the former event and reschedules it.

- **ServiceEvents.OnServiceDataAllLoaded**
  - Scheduling subject: ServiceManager
  - Scheduling timing: During the call to LoadServicesData, all services that implement ILoadDataService are processed.

7.3.4.5 Data save related events

```
/// <summary>
/// A single data service save data start event, dispatched by ServiceManager
/// Succ=true when the service implements the ISaveDataService interface.
/// 单个服务数据保存数据开始事件，由ServiceManager调度。当服务实现ISaveDataService接口时，Succ=true。
/// Event data format: ServiceResultData
/// 事件数据格式：ServiceResultData
/// </summary>
public const string OnServiceDataSaveStart = "ServiceEvents:OnServiceDataSaveStart";

/// <summary>
/// A single data service saves data completion events, which are dispatched by the service instance.
/// Re-dispatched after being captured by ServiceManager.
/// 单个数据服务保存数据完成事件，由服务实例调度。被ServiceManager捕获后重新调度。
/// Event data format: {named:string}
/// 事件数据格式：{named:string}
/// </summary>
public const string OnServiceDataSaved = "ServiceEvents:OnServiceDataSaved";

/// <summary>
/// All data services save data complete event
/// 全部数据服务保存数据完成事件
/// Event data format: len:int
/// 事件数据格式：len:int
/// </summary>
public const string OnServiceDataAllSaved = "ServiceEvents:OnServiceDataAllSaved";
```

- **ServiceEvents.OnServiceDataSaveStart**
  - Scheduling subject: ServiceManager
  - Scheduling timing: During the call to SaveServicesData, process each service before the behavior of the ISaveDataService interface.

- **ServiceEvents.OnServiceDataSaved**
  - Scheduling subject: service instance, ServiceManager that implements ISaveDataService
  - Scheduling timing: After the service instance that implements ISaveDataService handles the SaveData behavior; ServiceManager captures the former event and reschedules it.

- **ServiceEvents.OnServiceDataAllSaved**
  - Scheduling subject: ServiceManager
  - Scheduling timing: During the call to SaveServicesData, after all the services that implement ISaveDataService are processed.

7.4 How to use the framework

Before getting the service to use, it must be registered to the framework management, and then can be used after initialization. This is to ensure that the data has been processed completely.

7.4.1 Register the service to the framework management

By calling the AddConfig function in the ServiceConfig instance, the service can be registered with the framework management.
`ServiceConfig.Shared.AddConfig(new ServiceInfo(serviceName, serviceImpl, args));`

```
/// <summary>
/// Add a service to the end of the configuration list
/// 添加服务到配置列表尾部
/// </summary>
/// <param name="sc"></param>
/// <param name="ignoreSame"></param>
/// <exception cref="Exception"></exception>
public void AddConfig(ServiceInfo sc, bool ignoreSame = true)
{
    if (!ignoreSame && ContainsService(sc.ServiceName))
    {
        throw new Exception($"重复的ServiceName:{sc.ServiceName}");
    }

    m_Config.Add(sc);
    sc.ServiceImpl.ServiceName = sc.ServiceName;
}
```

Suggest:

1. Manage the registration logic with a class dedicated to handling registration services. Such as ServiceRegister in the example.
2. Registration logic calls should be guaranteed to be unique.

7.4.2 Service initialization

Start initialization by calling the **StartInitalization** function in the **ServiceManager** instance.
Note: It should not be called repeatedly after initialization has started.

```
/// <summary>
/// Initialize the configured service
/// 初始化配置好的服务
/// If it has been initialized once, try to execute the callback directly
/// 如果已经初始化一次，就尝试直接执行回调
/// </summary>
/// <param name="endCall"></param>
public void StartInitalization(Callback endCall)
{
    m_Services = ServiceConfig.Shared.ServiceInfos;
    m_FinishCall = endCall;

    InjectToServices();
    ActiveServices();
    m_ProcessingLen = GetServiceTotalLen();
    AddProcessingEvents();

    StartInitServices();
}
```

7.4.3 Service call

After the initial completion of all services, the service instance can be obtained by calling the **GetService** function in **ServiceConfig**.
There are multiple functions for obtaining service instances in ServiceConfig, but the GetService function is ultimately called.

```
/// <summary>
/// Get Service instance through ServiceName
/// 通过ServiceName取得Service实例
/// </summary>
/// <param name="serviceNamed"></param>
/// <returns></returns>
public static object GetService(string serviceNamed)
{
    return ServiceConfig.Shared.GetServiceImpl(serviceNamed);
}

/// <summary>
/// Get Service instance through ServiceName
/// 通过ServiceName取得Service实例
/// </summary>
/// <param name="named"></param>
/// <typeparam name="T">转换为指定接口</typeparam>
/// <returns></returns>
public static T GetService<T>(string named)
{
    var service = GetService(named);
    if (service is T)
    {
        return (T) service;
    }

    return (T) (object) null;
}
```

Suggest:

1. Manage the acquisition logic using a class that specifically manages acquiring service instances. Such as ServiceCenter in the example.

7.4.4 Service cleanup

When the game is to achieve a soft restart, it is necessary to reset all services and then reinitialize.
All services can be reset by calling **ClearServices** in **ServiceManager**.
Only services that implement the IClearService interface will perform reset logic.

```
/// <summary>
/// Service cleanup
/// 服务清理
/// </summary>
public void ClearServices()
{
    RemoveEventListener();
    for (var i = m_Services.Length - 1; i >= 0; i--)
    {
        (m_Services[i].ServiceImpl as IClearService)?.Clear();
    }
}
```

```
/// <summary>
/// Reset service interface, providing reset service to initialized state
/// 重置服务接口，提供把服务重置到初始化状态
/// </summary>
public interface IClearService
{
    /// <summary>
    /// reset
    /// 重置
    /// Clear events, clear timers, etc.
    /// 清除事件、清除计时器等
    /// </summary>
    void Clear();
}
```

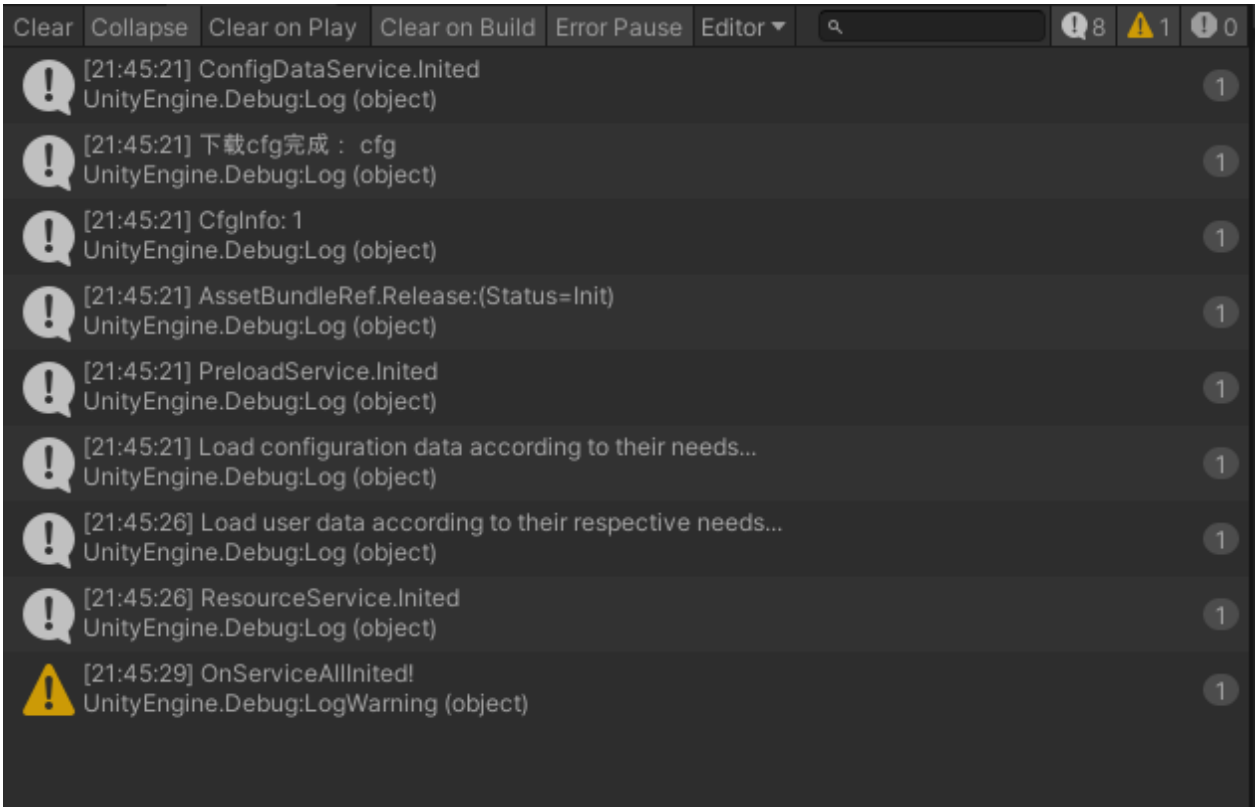
Suggest:

1. Each service is recommended to implement the IClearService interface.
2. In the Clear function in the IClearService interface, also clear all event listeners.

7.5 Example

GameDriver/Samples/Service

Name	IArgumentService	IAwakableService	IInitService	IInitDataService
ServiceNames.ConfigData	Disable	Finish	Finish	Waiting
ServiceNames.Preload	Disable	Finish	Finish	Waiting
ServiceNames.Material_1	Disable	Disable	Finish	Waiting
ServiceNames.DemoService1	Finish	Disable	Finish	Waiting
ServiceNames.DemoService2	Finish	Disable	Doing	Waiting
ServiceNames.DemoService3	Finish	Disable	Waiting	Waiting
ServiceNames.DemoService4	Finish	Disable	Waiting	Waiting



Service description in example:

- ConfigDataService
    1. Use ExcelExport to export the Excel header and data.
    2. cfg.asset is the DirFilesIndex asset manager, which is used to manage the path of Excel's Json data.
    3. Use the built-in Loader to load the cfg.asset and Json files and cache them.
    4. Init processing is complete.

- PreloadService
    1. Use PreloadSettings to manage the configuration information of preloaded resources.
    2. Use the built-in Loader to load the PreloadSettings asset.
    3. Load resources according to the PreloadSettings configuration information.
    4. Init processing is complete.

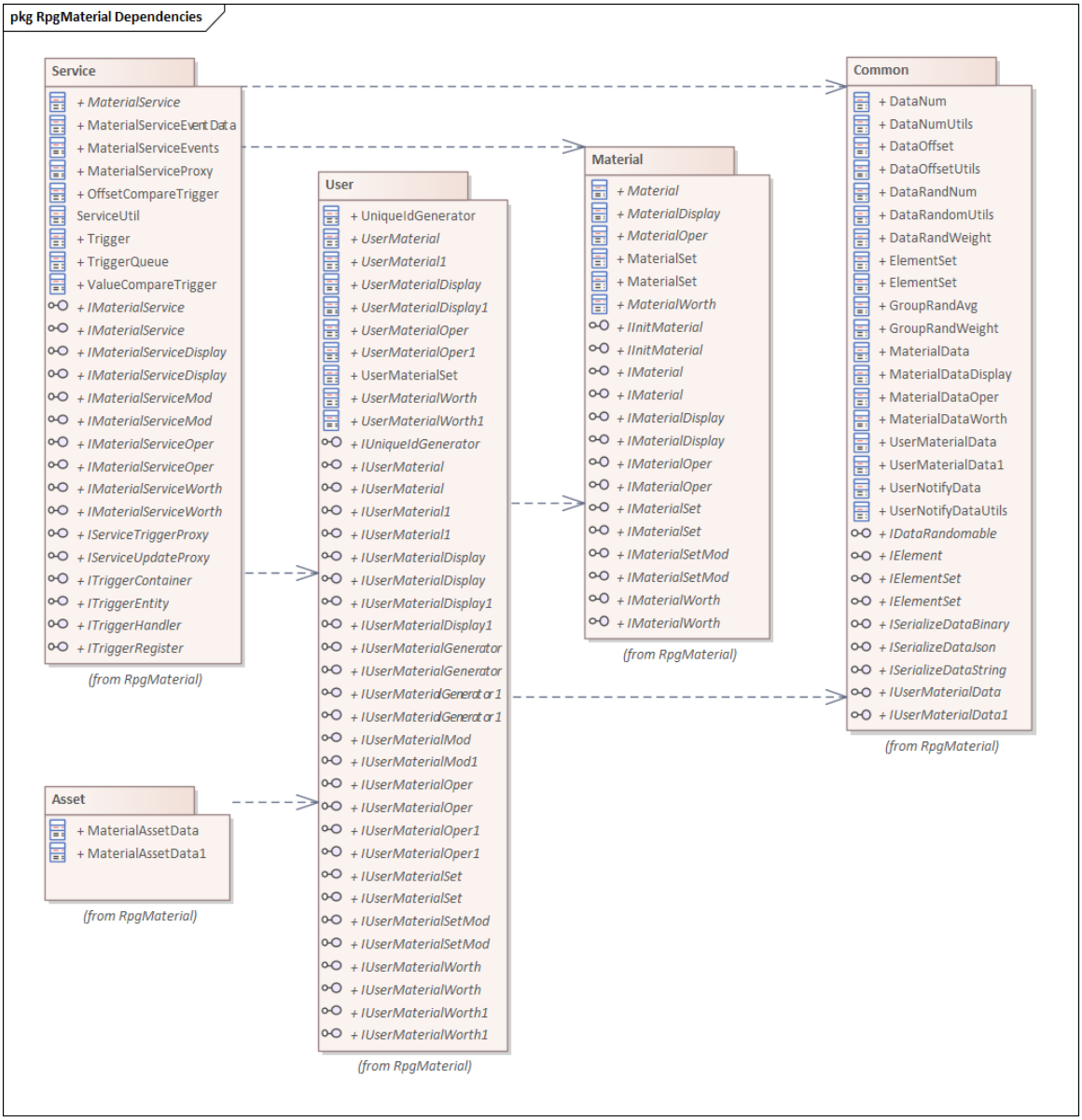
- ResourceService Service implementation under the Material Data Framework.
    1. Read the configuration from the ConfigDataService service.
    2. Initialize by configuration.
    3. Init processing is complete.
    4. Simulate the generation of user data.
    5. InitData processing is complete.

- DemoService The boilerplate service simulates different functions under the same service logic by injecting parameters.

8. Rpg Material Data System

This is a general game data management system.
The player’s numerical data is managed in the form of KTV (Key-Type-Value).

- **JLGames.GameDriver.Games.RpgMaterial** Provides functional support for the Rpg material data system. The dependencies are as follows:



- **JLGames.GameDriver.Games.RpgMaterial.Common** The basic data structure of the system.
- **JLGames.GameDriver.Games.RpgMaterial.Material** A data structure in the system about material definitions.
- **JLGames.GameDriver.Games.RpgMaterial.User** The data structure in the system about user data storage.
- **JLGames.GameDriver.Games.RpgMaterial.Service** The data structure in the system that provides external interface support.
- **JLGames.GameDriver.Games.RpgMaterial.Asset** The data structure in the system that is serialized for Unity.

8.1 Design Ideas

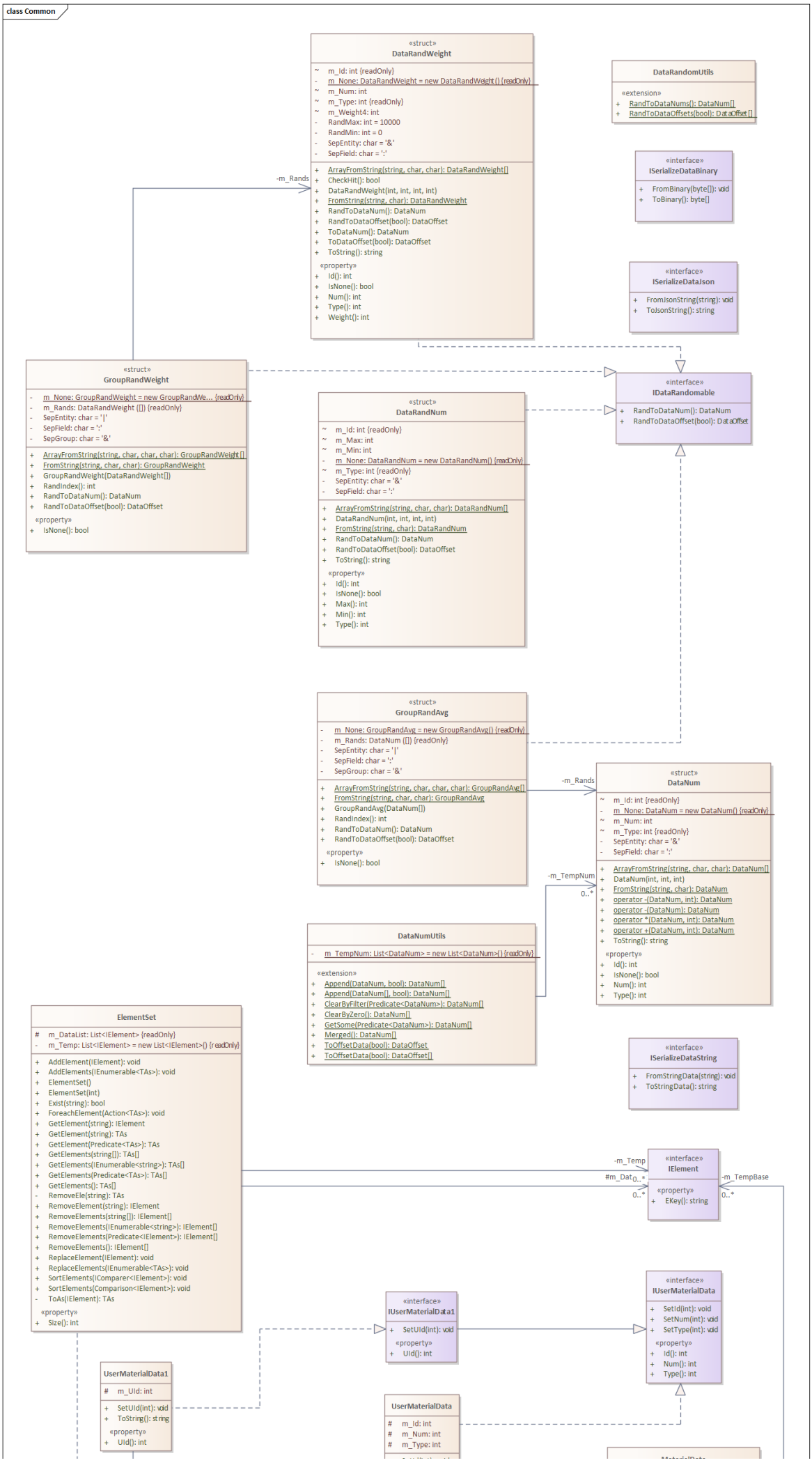
- Metadata (configuration data) of user data is managed in **KTD(Key-Type-Define)** format.
- The source of metadata (configuration data) can be derived from value table export, server acquisition, asset configuration, etc.

- Through **KT(Key&Type)**, the metadata (configuration data) definition (Define) can be obtained, and the validity can be checked when the user data is set.
- User data is managed in **KTV(Key-Type-Value)** format (memory, local file, server, etc.).
- Under the same metadata definition, if multiple pieces of user data are required, they will be saved in the form of **KTUV(Key-Type-UId-Value)**, which is compatible with both common data formats and multiple pieces of data. Require.
- User data (Value) usually uses basic numerical types to meet the needs, of course, game designers can expand the storage structure of data.
- Through **KT(Key&Type)** or **KTU(Key&Type&UId)**, the player's specific data (Value) can be obtained for calculation and display.
- Changes to user data are issued in the form of notifications, supporting both incremental changes and direct updates.

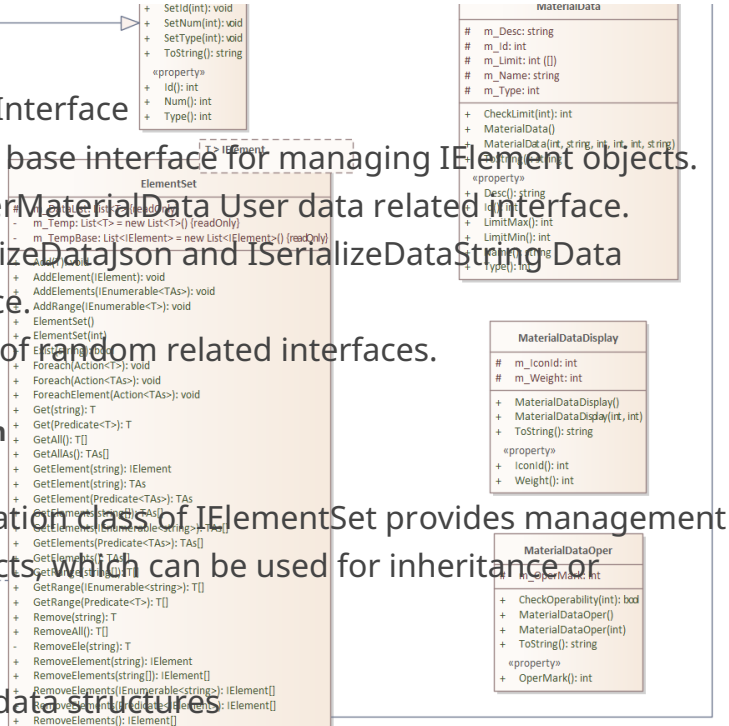
8.1.1 Common module design instructions

- Provides basic management functions of data objects and defines the most basic interface of data objects.
- Provide the most basic configuration data interface definition and related data structure implementation.
- Provide the most basic user data interface definition and related data structure implementation.
- Provides optional extension interface definition.

- Provides data structure implementations for basic computations.


class Common
«struct»
DataRandWeight
~ m_Id: int {readOnly}
- m_None: DataRandWeight = new DataRandWeight() {readOnly}
~ m_Num: int
~ m_Type: int {readOnly}
~ m_Weight4: int
- RandMax: int = 10000
- RandMin: int = 0
- SepEntity: char = '&'
- SepField: char = ':'
+ ArrayFromString(string, char, char): DataRandWeight[]
+ CheckHit(): bool
+ DataRandWeight(int, int, int, int)
+ FromString(string, char): DataRandWeight
+ RandToDataNum(): DataNum
+ RandToDataOffset(bool): DataOffset
+ ToDataNum(): DataNum
+ ToDataOffset(bool): DataOffset
+ ToString(): string
«property»
+ Id(): int
+ IsNone(): bool
+ Num(): int
+ Type(): int
+ Weight(): int
-m_Rands
DataRandomUtils
«extension»
+ RandToDataNums(): DataNum[]
+ RandToDataOffsets(bool): DataOffset[]
«interface»
ISerializeDataBinary
+ FromBinary(byte[]): void
+ ToBinary(): byte[]
«interface»
ISerializeDataJson
+ FromJsonString(string): void
+ ToJsonString(): string
«struct»
GroupRandWeight
- m_None: GroupRandWeight = new GroupRandWe... {readOnly}
- m_Rands: DataRandWeight ([]) {readOnly}
- SepEntity: char = '|'
- SepField: char = ':'
- SepGroup: char = '&'
+ ArrayFromString(string, char, char, char): GroupRandWeight[]
+ FromString(string, char, char): GroupRandWeight
+ GroupRandWeight(DataRandWeight[])
+ RandIndex(): int
+ RandToDataNum(): DataNum
+ RandToDataOffset(bool): DataOffset
«property»
+ IsNone(): bool
«interface»
IDataRandomable
+ RandToDataNum(): DataNum
+ RandToDataOffset(bool): DataOffset
«struct»
DataRandNum
~ m_Id: int {readOnly}
~ m_Max: int
~ m_Min: int
- m_None: DataRandNum = new DataRandNum() {readOnly}
~ m_Type: int {readOnly}
- SepEntity: char = '&'
- SepField: char = ':'
+ ArrayFromString(string, char, char): DataRandNum[]
+ DataRandNum(int, int, int, int)
+ FromString(string, char): DataRandNum
+ RandToDataNum(): DataNum
+ RandToDataOffset(bool): DataOffset
+ ToString(): string
«property»
+ Id(): int
+ IsNone(): bool
+ Max(): int
+ Min(): int
+ Type(): int
«struct»
GroupRandAvg
- m_None: GroupRandAvg = new GroupRandAvg() {readOnly}
- m_Rands: DataNum ([]) {readOnly}
- SepEntity: char = '|'
- SepField: char = ':'
- SepGroup: char = '&'
+ ArrayFromString(string, char, char, char): GroupRandAvg[]
+ FromString(string, char, char): GroupRandAvg
+ GroupRandAvg(DataNum[])
+ RandIndex(): int
+ RandToDataNum(): DataNum
+ RandToDataOffset(bool): DataOffset
«property»
+ IsNone(): bool
-m_Rands
«struct»
DataNum
~ m_Id: int {readOnly}
- m_None: DataNum = new DataNum() {readOnly}
~ m_Num: int
~ m_Type: int {readOnly}
- SepEntity: char = '&'
- SepField: char = ':'
+ ArrayFromString(string, char, char): DataNum[]
+ DataNum(int, int, int)
+ FromString(string, char): DataNum
+ operator -(DataNum, int): DataNum
+ operator -(DataNum): DataNum
+ operator *(DataNum, int): DataNum
+ operator +(DataNum, int): DataNum
+ ToString(): string
«property»
+ Id(): int
+ IsNone(): bool
+ Num(): int
+ Type(): int
-m_TempNum
0..*
DataNumUtils
- m_TempNum: List<DataNum> = new List<DataNum>() {readOnly}
«extension»
+ Append(DataNum, bool): DataNum[]
+ Append(DataNum[], bool): DataNum[]
+ ClearByFilter(Predicate<DataNum>): DataNum[]
+ ClearByZero(): DataNum[]
+ GetSome(Predicate<DataNum>): DataNum[]
+ Merged(): DataNum[]
+ ToOffsetData(bool): DataOffset
+ ToOffsetData(bool): DataOffset[]
«interface»
ISerializeDataString
+ FromStringData(string): void
+ ToStringData(): string
ElementSet
m_DataList: List<IElement> {readOnly}
- m_Temp: List<IElement> = new List<IElement>() {readOnly}
+ AddElement(IElement): void
+ AddElements(IEnumerable<TAs>): void
+ ElementSet()
+ ElementSet(int)
+ Exist(string): bool
+ ForeachElement(Action<TAs>): void
+ GetElement(string): IElement
+ GetElement(string): TAs
+ GetElement(Predicate<TAs>): TAs
+ GetElements(string[]): TAs[]
+ GetElements(IEnumerable<string>): TAs[]
+ GetElements(Predicate<TAs>): TAs[]
+ GetElements(): TAs[]
- RemoveEle(string): TAs
+ RemoveElement(string): IElement
+ RemoveElements(string[]): IElement[]
+ RemoveElements(IEnumerable<string>): IElement[]
+ RemoveElements(Predicate<IElement>): IElement[]
+ RemoveElements(): IElement[]
+ ReplaceElement(IElement): void
+ ReplaceElements(IEnumerable<TAs>): void
+ SortElements(IComparer<IElement>): void
+ SortElements(Comparison<IElement>): void
- ToAs(IElement): TAs
«property»
+ Size(): int
-m_Temp
#m_Dat
0..*
«interface»
IElement
«property»
+ EKey(): string
-m_TempBase
0..*
«interface»
IUserMaterialData1
+ SetUId(int): void
«property»
+ UId(): int
«interface»
IUserMaterialData
+ SetId(int): void
+ SetNum(int): void
+ SetType(int): void
«property»
+ Id(): int
+ Num(): int
+ Type(): int
UserMaterialData1
m_UId: int
+ SetUId(int): void
+ ToString(): string
«property»
+ UId(): int
UserMaterialData
m_Id: int
m_Num: int
m_Type: int

8.1.1.1 Common interface design



8.1.1.2 Common data structure design

- ElementSet The implementation class of IElementSet provides management functions for IElement objects, which can be used for inheritance or combination.

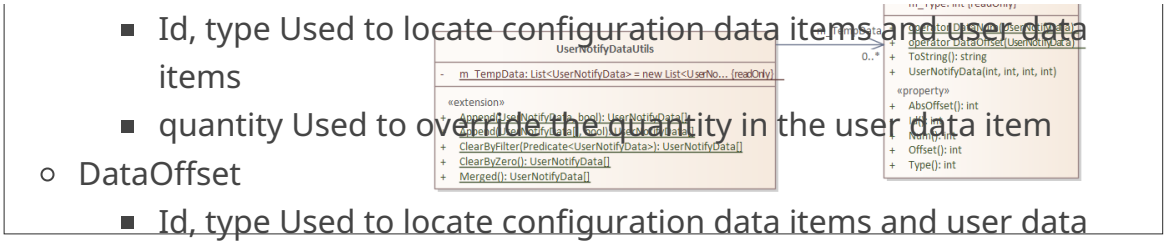
- Configuration data related data structures
  - MaterialData Configuration data base, including Id, name, type, size constraint, description.
  - MaterialDataDisplay Configuration data displays related data, including icon ID and display weight.
  - MaterialDataOper Configuration data manipulation behavior settings, including bit-wise defined values.

    Operation behavior settings:
    - Assume that the 1st bit of the binary is defined as usable and the 2nd bit is defined as discardable.
    - When the value is binary 11, it means that the data definition can be used or discarded.
  - MaterialDataWorth Configure data value to define relevant data, including value and quality.

    Value Definition:
    - The marked unit price of the type market item.
    - For quantity comparison and calculation between different configurations.

- User data related data structures
  - UserMaterialData User data base, including Id (corresponding to the Id of MaterialData), type (corresponding to the type of MaterialData), quantity
  - UserMaterialData1 Inherited from UserMaterialData, add the unique Id attribute

- Calculate and notify related data structures
  - DataNum



    - Id, type Used to locate configuration data items and user data items
    - quantity Used to override the quantity in the user data item
  - DataOffset
    - Id, type Used to locate configuration data items and user data items
    - offset value Used to add to the quantity in the user data item, the result overwrites the quantity in the user data item

  - UserNotifyData After updating the user data through the DataNum and DataOffset data, the UserNotifyData result can be calculated to notify the front-end display effect.
- random correlation data structure
  - DataRandNum Randomize the mean between the minimum and maximum values.
  - DataRandWeight There are only two results, true and false, for the random weights.
  - GroupRandAvg Random one of multiple values, with average probability.
  - GroupRandWeight Random one of multiple values, the probability depends on the weight.

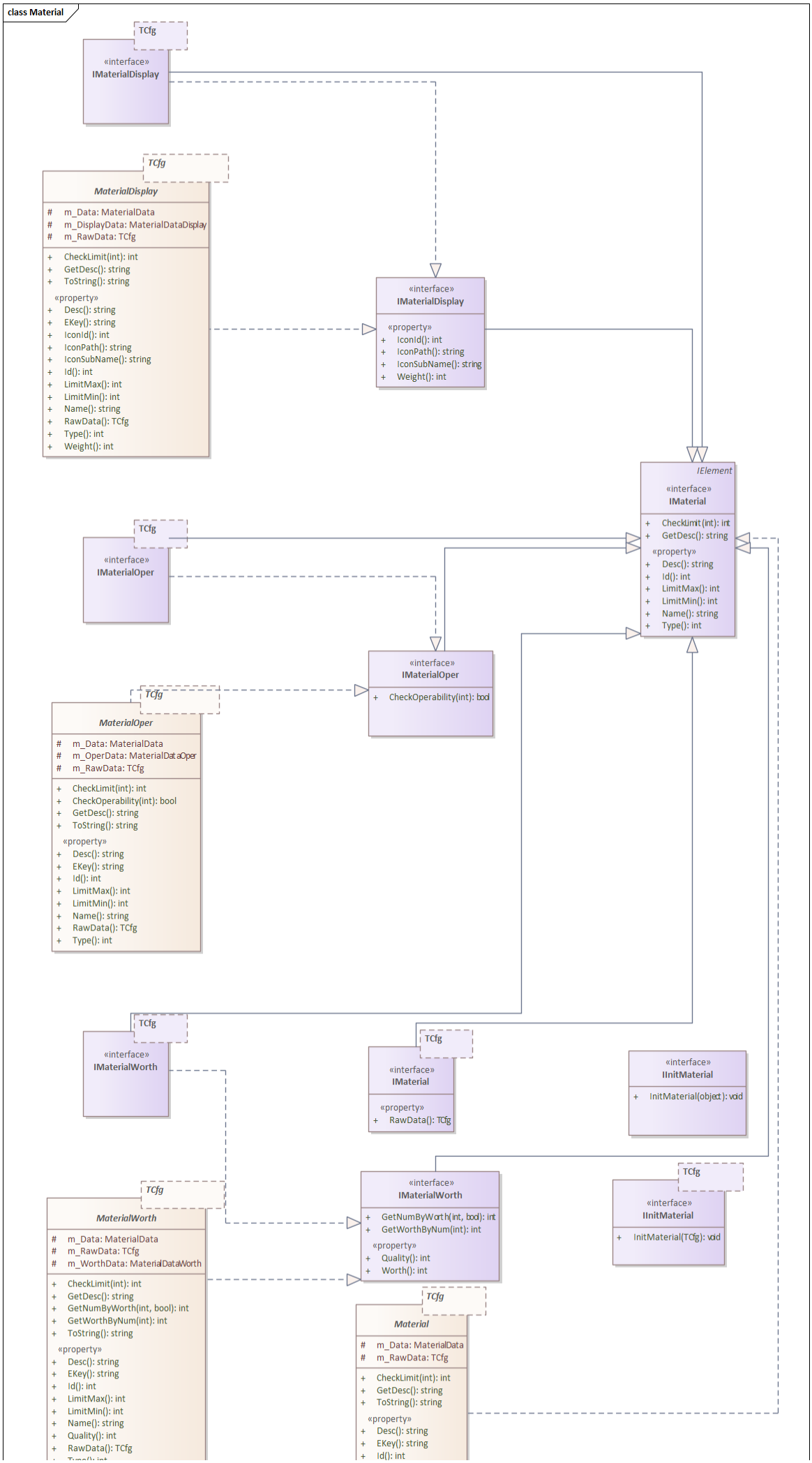
8.1.1.3 Common tool class design

- DataNumUtils Behavior logic related to DataNum
- DataOffsetUtils Behavior logic related to DataOffset
- UserNotifyDataUtils Behavior logic related to UserNotifyData
- DataRandomUtils Random logic for the class implementing the IDataRandomable interface

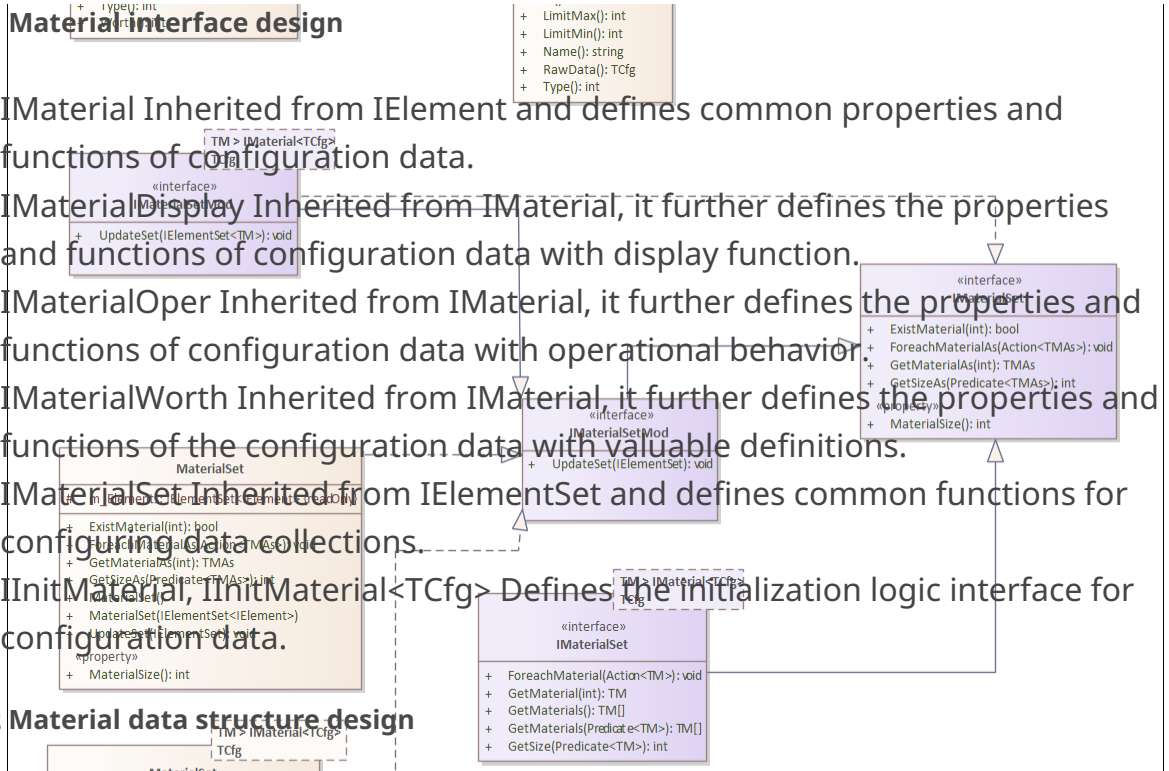
8.1.2 Material Module Design Instructions

- Provides common interface definitions and related implementations for configuration data.

- Combination of interfaces and data structures in Common.

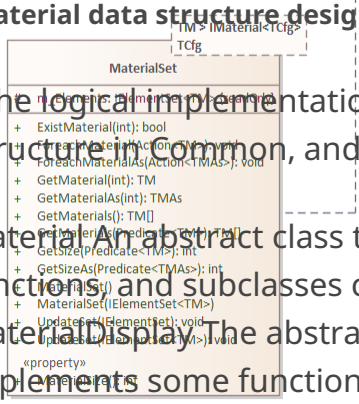

class Material
TCfg
«interface»
IMaterialDisplay
TCfg
MaterialDisplay
m_Data: MaterialData
m_DisplayData: MaterialDataDisplay
m_RawData: TCfg
+ CheckLimit(int): int
+ GetDesc(): string
+ ToString(): string
«property»
+ Desc(): string
+ EKey(): string
+ IconId(): int
+ IconPath(): string
+ IconSubName(): string
+ Id(): int
+ LimitMax(): int
+ LimitMin(): int
+ Name(): string
+ RawData(): TCfg
+ Type(): int
+ Weight(): int
«interface»
IMaterialDisplay
«property»
+ IconId(): int
+ IconPath(): string
+ IconSubName(): string
+ Weight(): int
IElement
«interface»
IMaterial
+ CheckLimit(int): int
+ GetDesc(): string
«property»
+ Desc(): string
+ Id(): int
+ LimitMax(): int
+ LimitMin(): int
+ Name(): string
+ Type(): int
TCfg
«interface»
IMaterialOper
«interface»
IMaterialOper
+ CheckOperability(int): bool
TCfg
MaterialOper
m_Data: MaterialData
m_OperData: MaterialDataOper
m_RawData: TCfg
+ CheckLimit(int): int
+ CheckOperability(int): bool
+ GetDesc(): string
+ ToString(): string
«property»
+ Desc(): string
+ EKey(): string
+ Id(): int
+ LimitMax(): int
+ LimitMin(): int
+ Name(): string
+ RawData(): TCfg
+ Type(): int
TCfg
«interface»
IMaterialWorth
TCfg
«interface»
IMaterial
«property»
+ RawData(): TCfg
«interface»
IInitMaterial
+ InitMaterial(object): void
TCfg
MaterialWorth
m_Data: MaterialData
m_RawData: TCfg
m_WorthData: MaterialDataWorth
+ CheckLimit(int): int
+ GetDesc(): string
+ GetNumByWorth(int, bool): int
+ GetWorthByNum(int): int
+ ToString(): string
«property»
+ Desc(): string
+ EKey(): string
+ Id(): int
+ LimitMax(): int
+ LimitMin(): int
+ Name(): string
+ Quality(): int
+ RawData(): TCfg
«interface»
IMaterialWorth
+ GetNumByWorth(int, bool): int
+ GetWorthByNum(int): int
«property»
+ Quality(): int
+ Worth(): int
TCfg
«interface»
IInitMaterial
+ InitMaterial(TCfg): void
TCfg
Material
m_Data: MaterialData
m_RawData: TCfg
+ CheckLimit(int): int
+ GetDesc(): string
+ ToString(): string
«property»
+ Desc(): string
+ EKey(): string
+ Id(): int

8.1.2.1 Material interface design



- IMaterial Inherited from IElement and defines common properties and functions of configuration data.
- IMaterialDisplay Inherited from IMaterial, it further defines the properties and functions of configuration data with display function.
- IMaterialOper Inherited from IMaterial, it further defines the properties and functions of configuration data with operational behavior.
- IMaterialWorth Inherited from IMaterial, it further defines the properties and functions of the configuration data with valuable definitions.
- IMaterialSet Inherited from IElementSet and defines common functions for configuring data collections.
- IInitMaterial, IInitMaterial<TCfg> Defines the initialization logic interface for configuration data.

8.1.2.2 Material data structure design



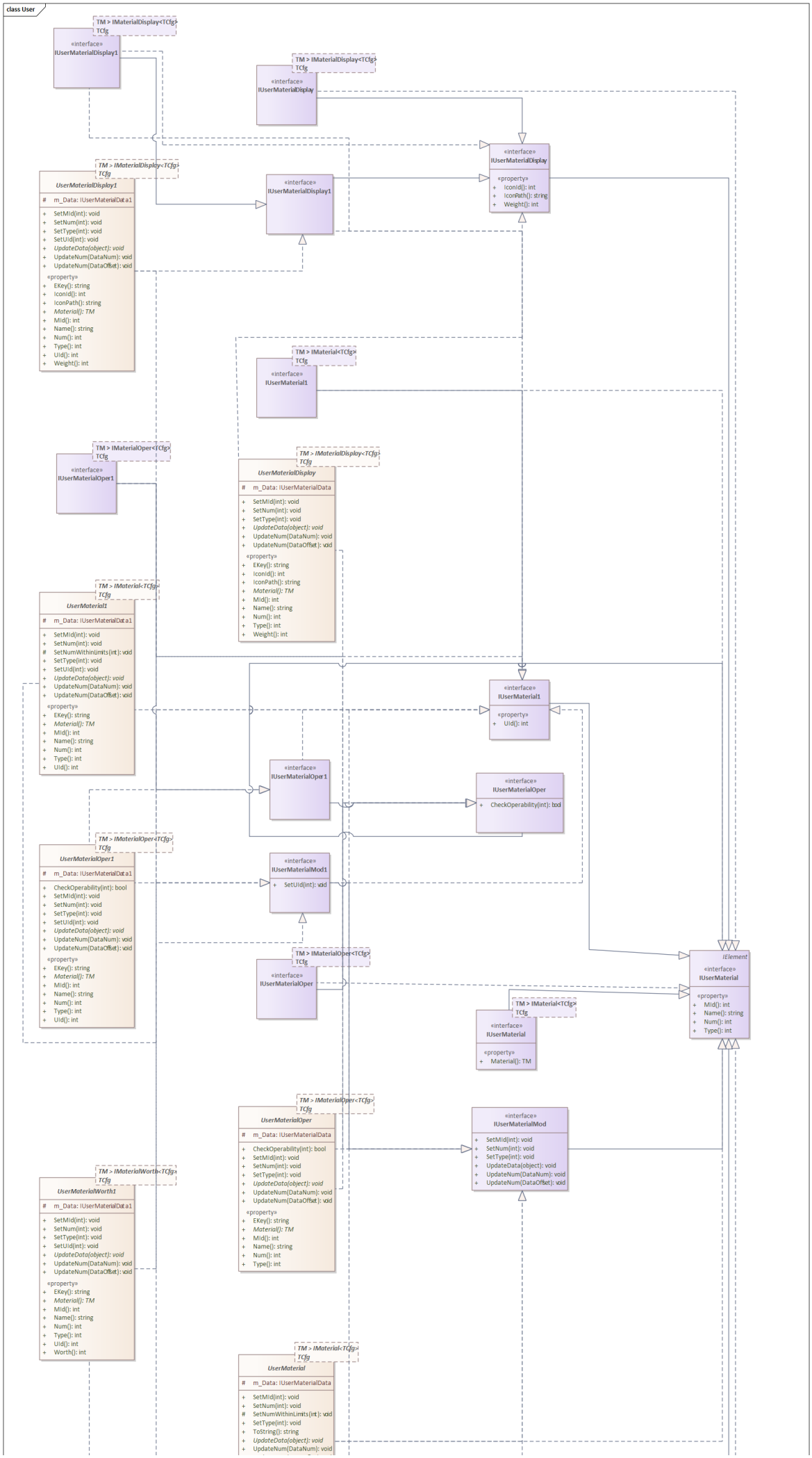
Note: The logical implementation of the following data structure depends on the array structure in common, and the usage method is **combination**.

- Material An abstract class that implements IMaterial, implements some functions, and subclasses can override behavior.
- MaterialDisplay The abstract class that implements IMaterialDisplay implements some functions, and subclasses can override the behavior.
- MaterialOper The abstract class that implements IMaterialOper, implements some functions, and subclasses can override the behavior.
- MaterialWorth An abstract class that implements IMaterialWorth, implements some functions, and subclasses can override the behavior.

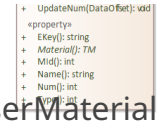
8.1.3 User module design description

- Provides normal user data functions.
- Provides user data function with unique Id.

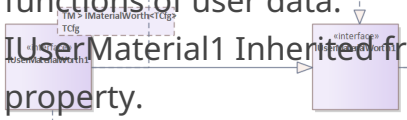
- Provides functionality for managing user data collections.

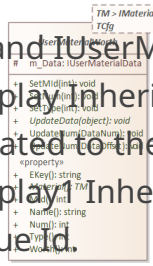

class User
TM > IMaterialDisplay<TCfg>
TCfg
«interface»
IUserMaterialDisplay1
TM > IMaterialDisplay<TCfg>
TCfg
«interface»
IUserMaterialDisplay
«interface»
IUserMaterialDisplay
«property»
+ IconId(): int
+ IconPath(): string
+ Weight(): int
TM > IMaterialDisplay<TCfg>
TCfg
UserMaterialDisplay1
m_Data: IUserMaterialData
+ SetMId(int): void
+ SetNum(int): void
+ SetType(int): void
+ SetUId(int): void
+ UpdateData(object): void
+ UpdateNum(DataNum): void
+ UpdateNum(DataOffset): void
«property»
+ EKey(): string
+ IconId(): int
+ IconPath(): string
+ Material(): TM
+ MId(): int
+ Name(): string
+ Num(): int
+ Type(): int
+ UId(): int
+ Weight(): int
«interface»
IUserMaterialDisplay1
TM > IMaterial<TCfg>
TCfg
«interface»
IUserMaterial1
TM > IMaterialOper<TCfg>
TCfg
«interface»
IUserMaterialOper1
TM > IMaterialDisplay<TCfg>
TCfg
UserMaterialDisplay
m_Data: IUserMaterialData
+ SetMId(int): void
+ SetNum(int): void
+ SetType(int): void
+ UpdateData(object): void
+ UpdateNum(DataNum): void
+ UpdateNum(DataOffset): void
«property»
+ EKey(): string
+ IconId(): int
+ IconPath(): string
+ Material(): TM
+ MId(): int
+ Name(): string
+ Num(): int
+ Type(): int
+ Weight(): int
TM > IMaterial<TCfg>
TCfg
UserMaterial1
m_Data: IUserMaterialData
+ SetMId(int): void
+ SetNum(int): void
# SetNumWithinLimits(int): void
+ SetType(int): void
+ SetUId(int): void
+ UpdateData(object): void
+ UpdateNum(DataNum): void
+ UpdateNum(DataOffset): void
«property»
+ EKey(): string
+ Material(): TM
+ MId(): int
+ Name(): string
+ Num(): int
+ Type(): int
+ UId(): int
«interface»
IUserMaterial1
«property»
+ UId(): int
«interface»
IUserMaterialOper1
«interface»
IUserMaterialOper
+ CheckOperability(int): bool
TM > IMaterialOper<TCfg>
TCfg
UserMaterialOper1
m_Data: IUserMaterialData
+ CheckOperability(int): bool
+ SetMId(int): void
+ SetNum(int): void
+ SetType(int): void
+ SetUId(int): void
+ UpdateData(object): void
+ UpdateNum(DataNum): void
+ UpdateNum(DataOffset): void
«property»
+ EKey(): string
+ Material(): TM
+ MId(): int
+ Name(): string
+ Num(): int
+ Type(): int
+ UId(): int
«interface»
IUserMaterialMod1
+ SetUId(int): void
TM > IMaterialOper<TCfg>
TCfg
«interface»
IUserMaterialOper
IElement
«interface»
IUserMaterial
«property»
+ MId(): int
+ Name(): string
+ Num(): int
+ Type(): int
TM > IMaterial<TCfg>
TCfg
«interface»
IUserMaterial
«property»
+ Material(): TM
TM > IMaterialOper<TCfg>
TCfg
UserMaterialOper
m_Data: IUserMaterialData
+ CheckOperability(int): bool
+ SetMId(int): void
+ SetNum(int): void
+ SetType(int): void
+ UpdateData(object): void
+ UpdateNum(DataNum): void
+ UpdateNum(DataOffset): void
«property»
+ EKey(): string
+ Material(): TM
+ MId(): int
+ Name(): string
+ Num(): int
+ Type(): int
«interface»
IUserMaterialMod
+ SetMId(int): void
+ SetNum(int): void
+ SetType(int): void
+ UpdateData(object): void
+ UpdateNum(DataNum): void
+ UpdateNum(DataOffset): void
TM > IMaterialWorth<TCfg>
TCfg
UserMaterialWorth1
m_Data: IUserMaterialData
+ SetMId(int): void
+ SetNum(int): void
+ SetType(int): void
+ SetUId(int): void
+ UpdateData(object): void
+ UpdateNum(DataNum): void
+ UpdateNum(DataOffset): void
«property»
+ EKey(): string
+ Material(): TM
+ MId(): int
+ Name(): string
+ Num(): int
+ Type(): int
+ UId(): int
+ Worth(): int
TM > IMaterial<TCfg>
TCfg
UserMaterial
m_Data: IUserMaterialData
+ SetMId(int): void
+ SetNum(int): void
# SetNumWithinLimits(int): void
+ SetType(int): void
+ ToString(): string
+ UpdateData(object): void
+ UpdateNum(DataNum): void

8.1.3.1 User interface design

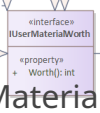


- IUserMaterial and IUserMaterial1
  - IUserMaterial Inherited from IElement, defines the properties and functions of user data.
  - IUserMaterial1 Inherited from IUserMaterial, adding a unique Id property.

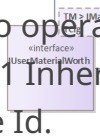


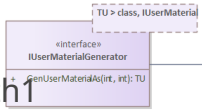


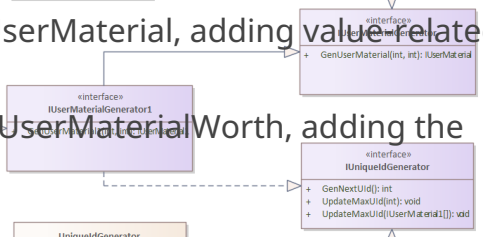
- IUserMaterialDisplay and IUserMaterialDisplay1
  - IUserMaterialDisplay Inherited from IUserMaterial, adding properties and functions related to the display function.
  - IUserMaterialDisplay1 Inherited from IUserMaterialDisplay, adding the property of Unique Id.



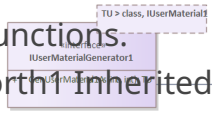
- IUserMaterialOper and IUserMaterialOper1
  - IUserMaterialOper Inherited from IUserMaterial, adding properties and functions related to operation behavior.
  - IUserMaterialOper1 Inherited from IUserMaterialDisplay, adding the property of Unique Id.



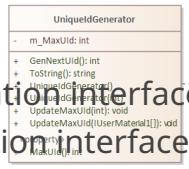


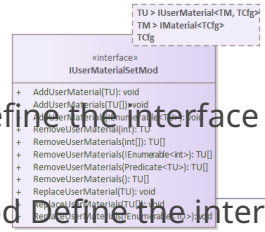


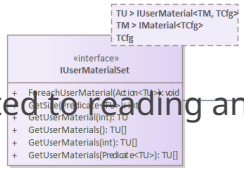
- IUserMaterialWorth and IUserMaterialWorth1
  - IUserMaterialWorth Inherited from IUserMaterial, adding value-related properties and functions.
  - IUserMaterialWorth1 Inherited from IUserMaterialWorth, adding the unique Id attribute.



- IUserMaterialMod and IUserMaterialMod1
  - IUserMaterialMod Define the modification interface of user data
  - IUserMaterialMod1 Define a modification interface for user data with a unique Id

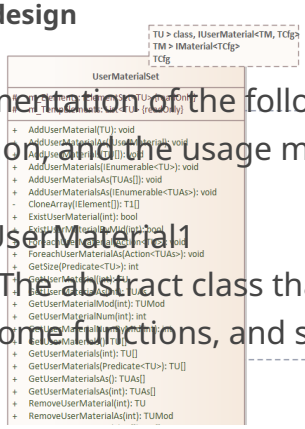


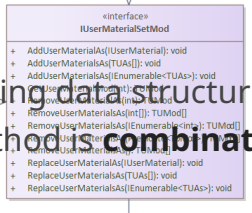


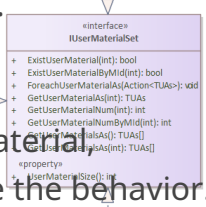


- IUserMaterialSet Define the interface related to reading and searching in the user data collection
- IUserMaterialSetMod Define the interface related to modification in the user data collection

8.1.3.2 User data structure design

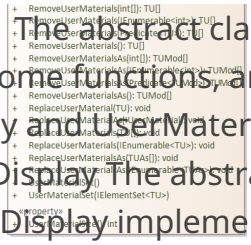






Note: The logical implementation of the following data structure depends on the array structure in Common, and the usage method is **combination**.

- UserMaterial and UserMaterial1
  - UserMaterial The abstract class that implements IUserMaterial, implements some functions, and subclasses can override the behavior.



  - UserMaterial1 The abstract class that implements IUserMaterial1, implements some functions, and subclasses can override the behavior.
- UserMaterialDisplay and UserMaterialDisplay1
  - UserMaterialDisplay The abstract class that implements IUserMaterialDisplay, implements some functions, and subclasses can override the behavior.
  - UserMaterialDisplay1 The abstract class that implements IUserMaterialDisplay1, implements some functions, and subclasses can override the behavior.

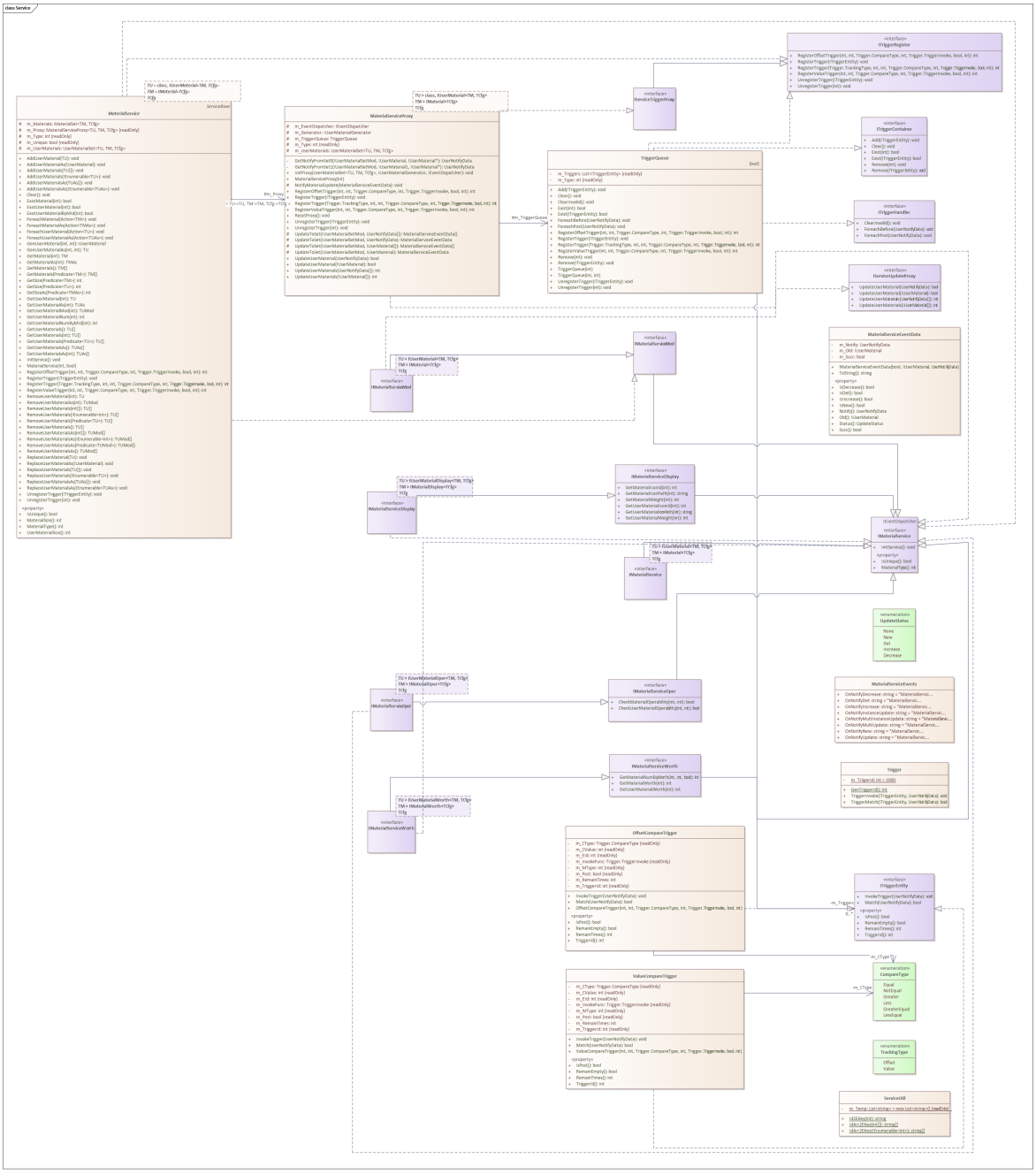
- UserMaterialOper and UserMaterialOper1
  - UserMaterialOper The abstract class that implements IUserMaterialOper, implements some functions, and subclasses can override the behavior.
  - UserMaterialOper1 The abstract class that implements IUserMaterialWorth1, implements some functions, and subclasses can override the behavior.

- UserMaterialWorth and UserMaterialWorth1
  - UserMaterialWorth The abstract class that implements IUserMaterialWorth1, implements some functions, and subclasses can override the behavior.
  - UserMaterialWorth1 The abstract class that implements IUserMaterialWorth1, implements some functions, and subclasses can override the behavior.

- UserMaterialSet The abstract class that implements IUserMaterialSet, implements some functions, and subclasses can override the behavior.

8.1.4 Service Module Design Instructions

- The design of material Service extends the function of Service Framework.
- The classification of the material Service should correspond to the type (referring to the type in the configuration data) one-to-one, that is, a type creates a Service.
- Provides the basic interface definition of Material Service.
- Provides basic interface definitions for displayable materials, actionable materials, and valuable materials.
- Provides a default implementation class of material service, which can be used as needed. Direct modification and inheritance are not recommended.



8.1.4.1 Service interface design

- IMaterialService Basic interface definition for Material Service.
- IMaterialServiceDisplay Displays the basic interface definition of the Material Service.
- IMaterialServiceOper Basic interface definition for the Actionable Material Service.
- IMaterialServiceWorth Basic interface definition for Value Material Service.
- IMaterialServiceMod Modification-related interface definitions for material services.

8.1.4.2 Service Logic Design

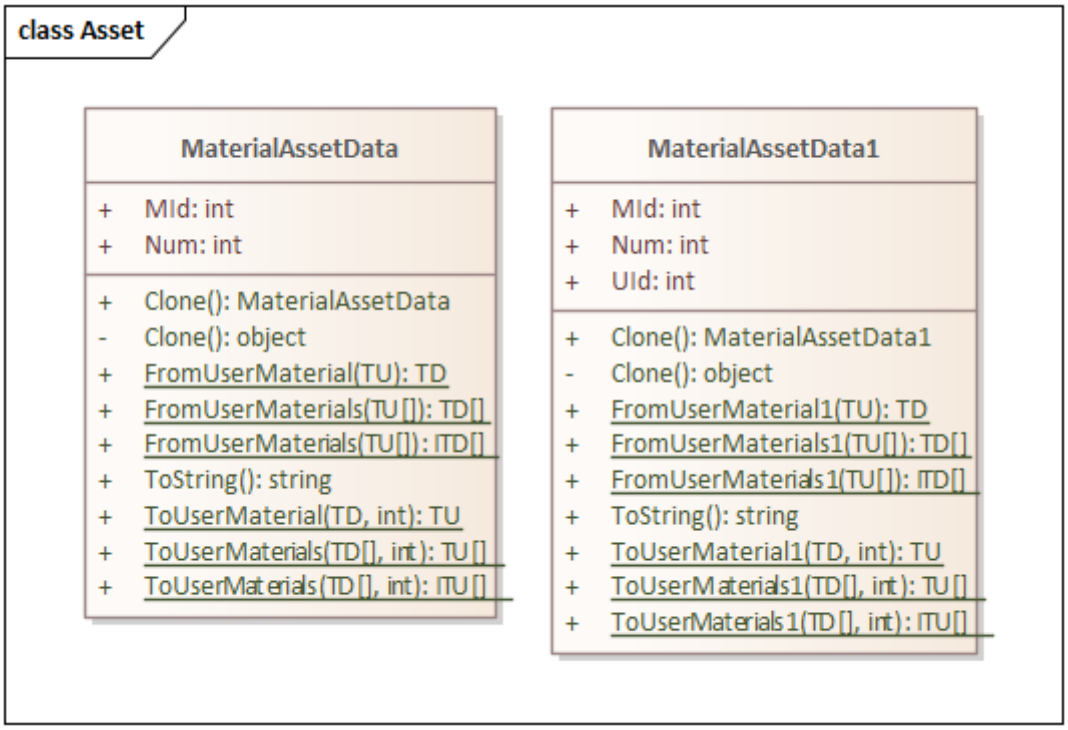
- MaterialService A default material service implementation class that can be used on demand. Direct modification and inheritance are not recommended.

Internal functions are composed of functional units in User, Material, and Common.

- ServiceUtil Provide some basic logic functions related to service.

8.1.5 Asset Module Design Instructions

- MaterialAssetData Serialized structure of material user data, used in Unity panels.
- MaterialAssetData1 Serialized structure of material user data with unique Id for use in Unity panels.



8.2 Use

Depends on the service framework(JLGames.GameDriver.Games.Services), and the specific usage process is consistent with the general service.

8.2.1 Classification of material data

Classify material configuration table data according to business needs

8.2.2 Prepare the interface

Create a service interface for each type of material, at least inherit IInitService, IInitDataService, IMaterialService

- IInitService Used to initialize material configuration data
- IInitDataService Used to initialize material user data.
  User data source:
  1. Server request

2. Local cache file
3. Other sources

- IMaterialService Provides some material-related properties and functions.

8.2.3 Implementing the interface

Implement the interface created in the previous step.

8.2.4 Configure and initialize according to the service framework process

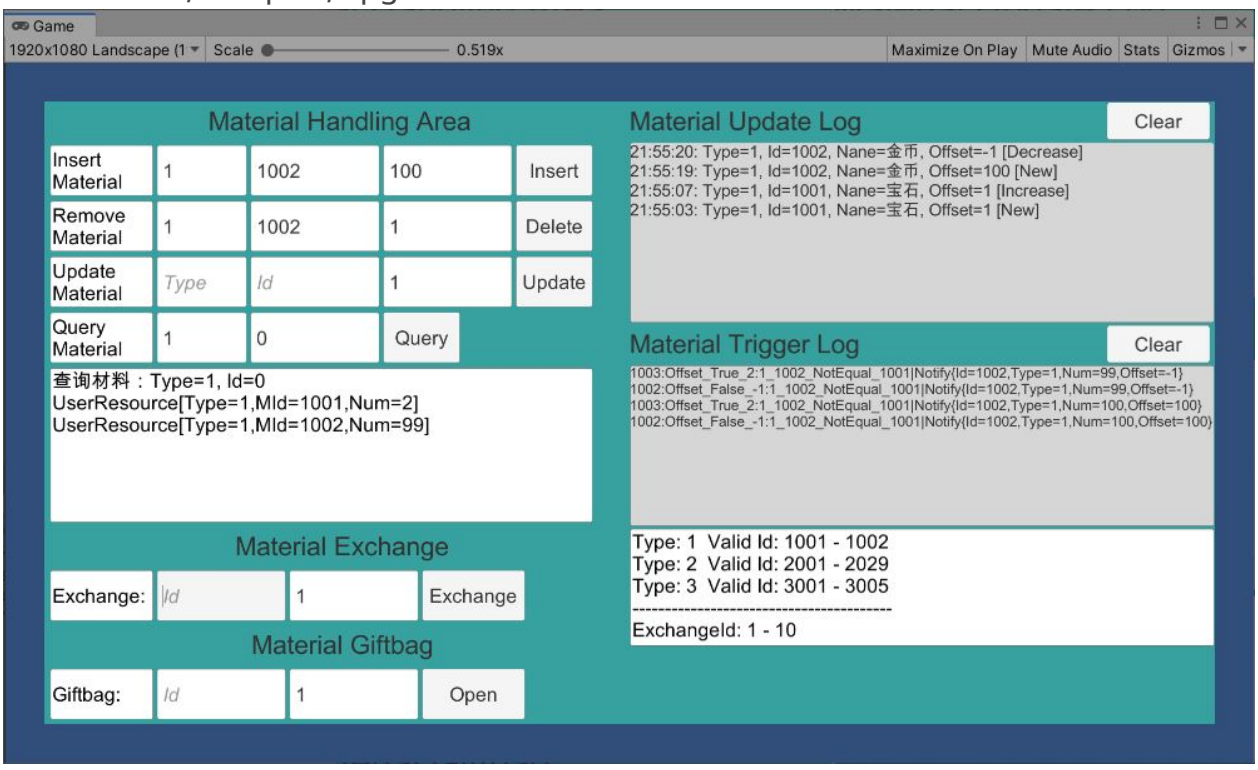
According to the [Service Framework Description] (Manual_7.Service_en.html#7.4), register, initialize, and call.

8.2.5 Other functions

- By default, an event is thrown when the user data changes. Perform logic processing after data change by listening to corresponding events.
- Supports trigger function by default, and supports trigger execution when user data changes.

8.3 Example

GameDriver/Samples/RpgMaterial



- Example Explanation
  1. Provides examples of three material types: Resource, Item, and Giftbag.
  2. An example of the material exchange function is provided.

3. Provided an example of opening function of gift bag (Giftbag).