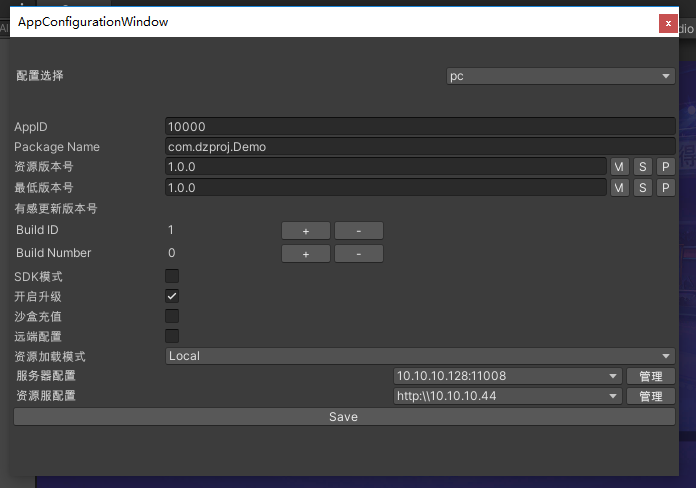
## AppConfiguration



|  |  |
| --- | --- |
| AppID | 应用唯一ID |
| Package Name | 当前Aapp包名 |
| 资源版本号 | 热更新版本号（M:主版本，S:次版本，P:补丁版本） |
| 最低版本号 | 低于该版本号，将会提示大包更新 |
| 有感更新版本号 | 低于该版本号，将会有弹窗提示更新 |
| Build ID | 区分发包编号，每次正式包发布都会递增，相对远端配置也需要新增一份与之对应的配置 |
| Build Number | 当前Build ID下构建了多少次，每一次打包都需要递增 |
| SDK模式 | SDK登录相关 |
| 开启升级 | 是否开启热更新 |
| 沙盒充值 | iOS沙盒充值测试 |
| 远端配置 | 是否启用远端配置。参考远端配置说明 |
| 资源加载模式 | Local：加载编辑器内资源，Asset Bundle：从AB中加载资源。发包后始终为AssetBundle加载。 |
| 服务器配置 | TCP的IP和端口 |
| 资源服配置 | OSS地址，资源存储服务器地址 |

## 资源服务器目录结构

默认资源服务器目录结构都是固定的不可改变。

以OSS上为例：

new-depu 节点

Packs – 为根目录

Config – 为配置相关目录，如Apps远端配置，服务器地址配置

Lang – 按不同平台存放多语言文件与AB

Upgrade – 按不同平台存放对应的热更新资源

## 远端配置

远端配置只有在打包时勾选了远端配置选项后才激活。

它的作用是用于控制当前AppID与BuildID对应的包是处于正式发行？、审核？还是其他状态。不同状态下使用不同的配置与不同的服务器地址。

目前是有四种状态：

1. Production： 生产环境
2. Development： 开发环境
3. Auditing： 审核环境
4. Bate： Bate环境

### Apps.json

分为两块配置：Apps和Servers。

#### Apps配置



如图所示：不同的包会根据自身AppID和BuildID选择对应的配置。

* status：App的状态
* is\_open\_upgrade：是否开启热更新功能
* is\_open\_sdk: 是否开启SDK登录模块
* is\_open\_sandbox: 是否开启iOS沙盒充值测试

#### Servers配置



App会根据从Apps配置中的状态选取对应的服务器配置。

* res\_url: 资源服务器地址。如果需要变更资源服务器地址那就替换一个新的，否则使用当前默认的地址。
* server\_list\_url: 服务器地址文件存放地址

### host\_list.json

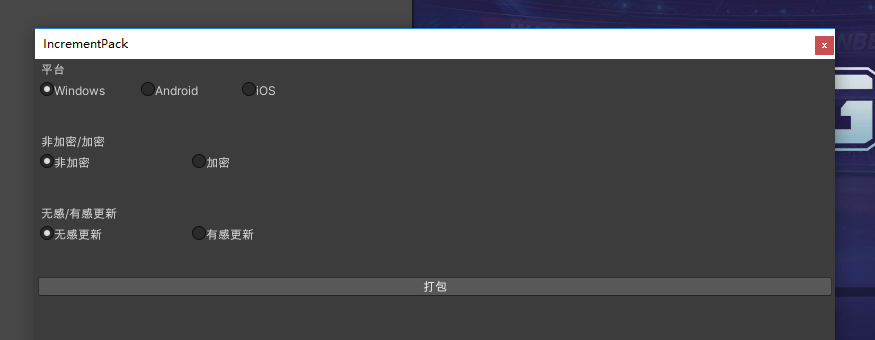
App会根据从Apps配置中的状态选取对应登录服务器和游戏服务器地址，分为国内和海外



## 热更新生成与发布

### 生成

首先、打开DZProjMenu/AppConfiguration，点击增量包，打开增量包打包面板



选择对应平台进行打包。

在打包完成后会在当前项目根目录的Export/IncrementPacks/{Platform}/{日期}/中。

该目录中有3个文件：

* increment.zip：所有热更新文件
* modified.log.json：变更日志
* version.json：新的版本信息

### 发布

首先，打开OSS，进入到Packs/Upgrade/{Platform}/。

然后，解压increment.zip，将更新文件全部拖入目录中。

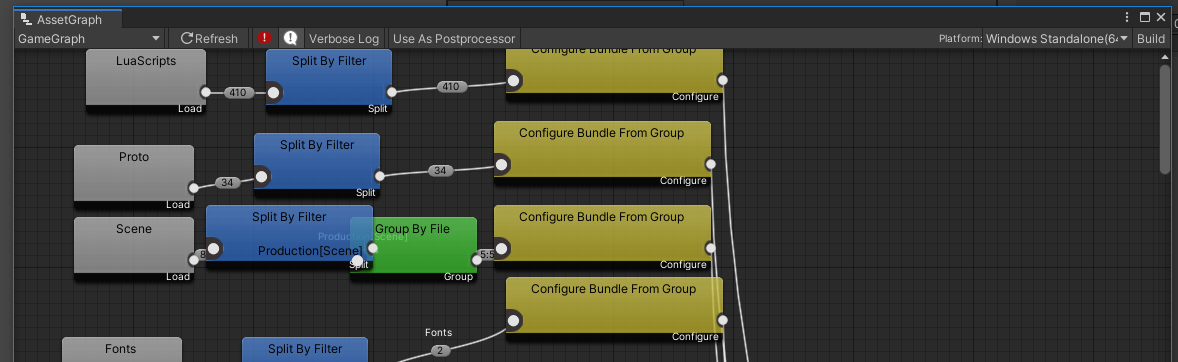
最后，确认版本号后，将新的version.json文件拖入目录中。

## AssetBundle打包与配置

目前项目中使用AssetGraph工具进行AB打包与配置

### 配置AB

首先，在编辑器中双击打开Assets/AssetGraphs/GameGraph配置。

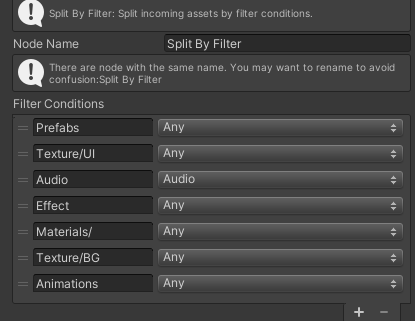
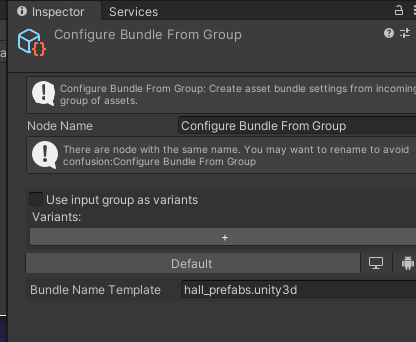
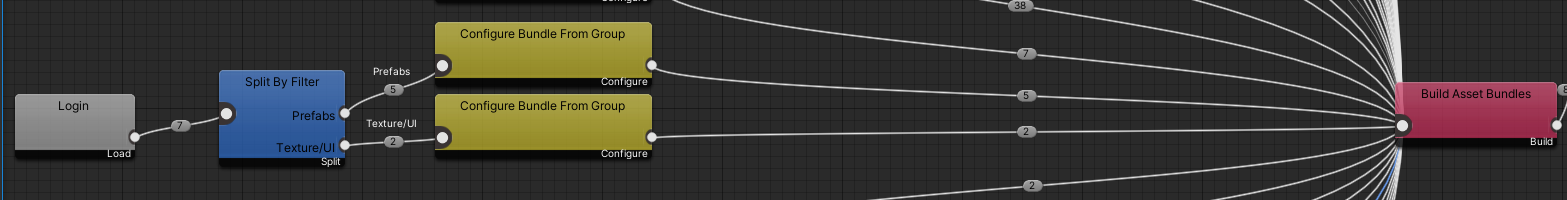


如图所示。

#### 常用节点说明

1. Build Asset Bundles：执行打包AB的节点。
2. AppendExtNode：用于附加后缀名的节点。项目中ab统一使用.unity3d作为后缀。
3. Mirror Directory：将生成好AB从生成目录复制一份到指定目录中。
4. Load From Directory：加载指定目录资源
5. Split By Filter：分类过滤指定资源
6. Configure Bundle From Group：输出的AB配置和输出时的AB名称

#### 创建一个AB打包配置

1. 右击画布空白处
2. 选择LoadAssets/Load From Directory
3. 这个节点用于加载指定目录下资源用于打包AB，也可以直接从Assets中直接拖进来也可以。
4. 选择SplitAssets/Split By Filter
5. 这个节点用于过滤不同类型资源分别打包到不同的AB中去。
6. 选择Configure Bundle/Configure Bundle From Group
7. 这个节点配置最终输出的AB文件。
8. 最后将建立好的节点连接起来并将ConfigureBundle连接到Build Asset Bundles 上

### 执行打包

在编辑菜单中选择**DZProjMenu/AB/打包当前平台AB**。

## CI/CD配置

CI的配置文件放置在项目根目录中**.gitlab-ci.yml**



CI分为2个阶段：test和build。

Test: 目前没有做任何事，未来可以加入代码格式校验，资源格式校验与执行单元测试。

Build：打包阶段

现有3中CI执行配置：

1. apg-test: 只包含Test阶段因此目前只是走个过程。
2. apg-build: 用于开发阶段打包使用，它只在每日凌晨4点才会自动执行打包。
3. apg-publish-build: 用于发行包打包使用，它只能通过发起POST请求手动触发。

\*注：CI高级技巧参考 <https://www.jianshu.com/p/3c0cbb6c2936>

## GIT规范化配置

Git提交规范化配置文件放置在项目根目录**.cz-config.js**。



Types: 定义提交类型

Scopes: 定义受影响的模块