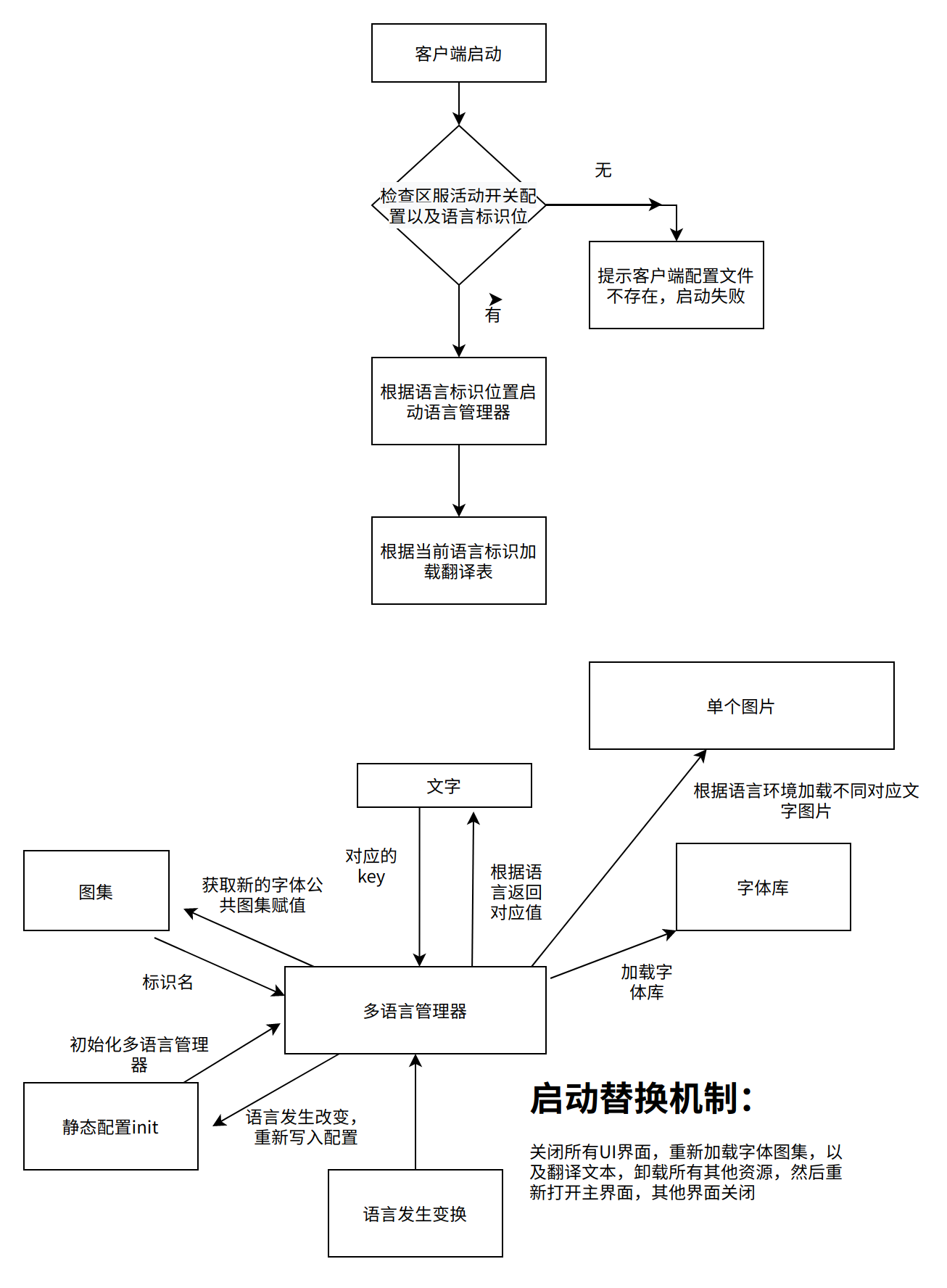
**多语言方案**

**设计目标**

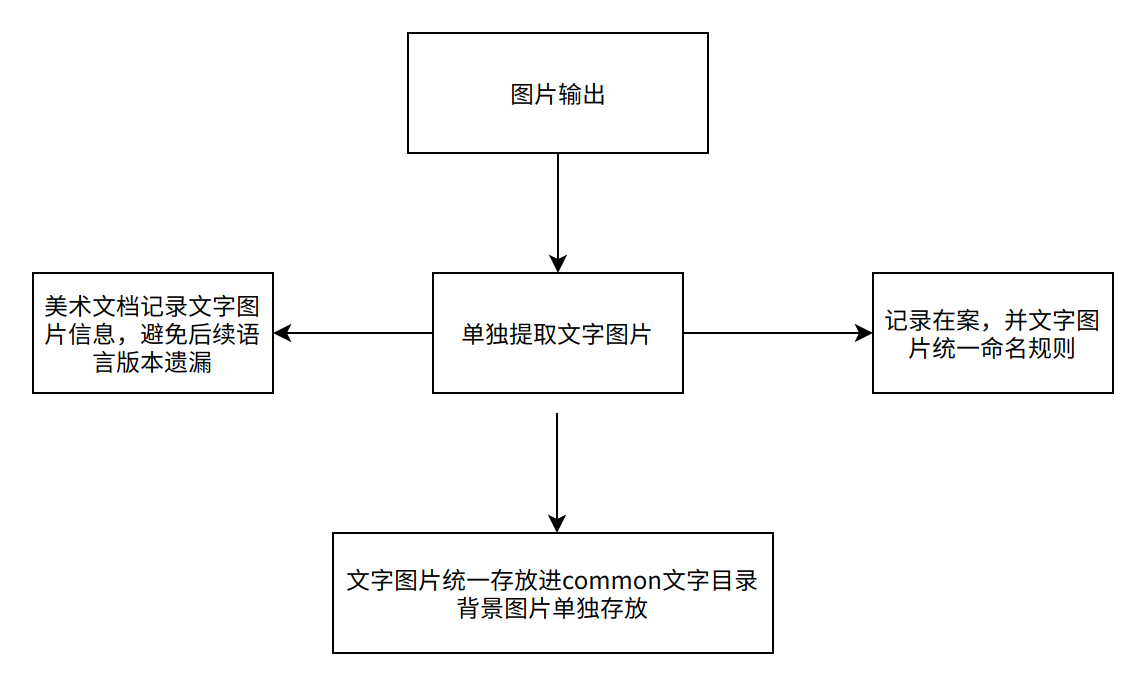
* 尽量减少策划工作量
* 可游戏运行时切换语言

**方案设计**

* 程序流程图

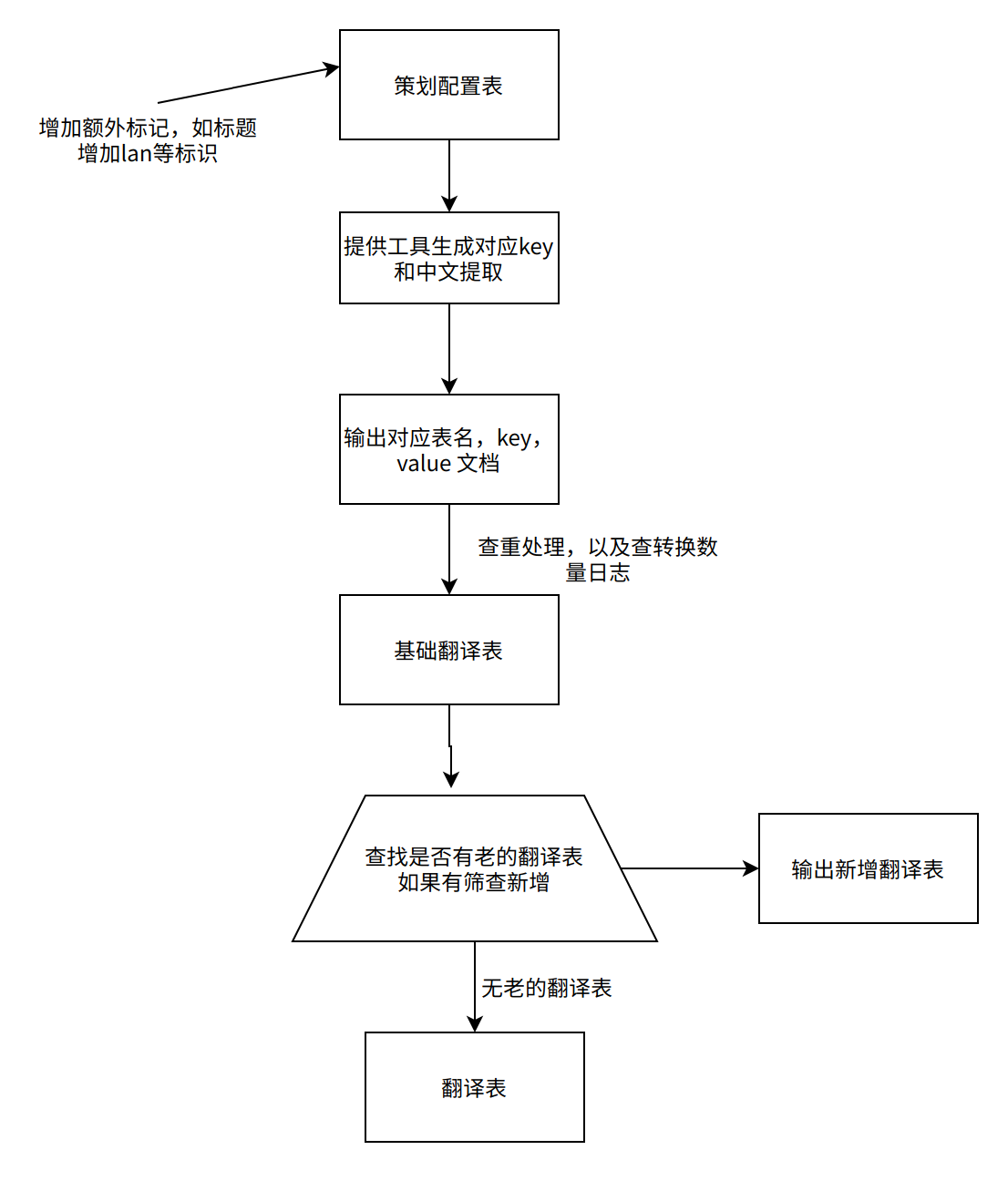


* 美术资源流程图



美术资源图集是公共图集，除非是大的背景图片单独存放，根据不同系统背景进行存放，客户端制作UI会根据特定情况进行路径记录管理加载，建议不要大的背景图片带文字，可以分割为文字和背景图片来处理，避免输出内容过多导致目录过多

* 策划配置流程图



**设计思路**

1.减少耦合产生的误差和逻辑不统一性，让多语言系统在统一管理相关多语言资源的功能，实现单一的输入输出口，减少代码的复杂度和可维护性

2.策划原有配置配置不影响逻辑情况下，进行二次工具转译绑定翻译表Key值进行匹配

3.资源独立分开处理，根据不同环境加载不同的资源，避免资源冗余现象

**需要处理的资源**

* **策划表格**
* 如何处理策划中的配置中文，流程图，生成

基于目前策划配置表的相关工作，需要策划进行表格标记处理，然后采用对应的工具进行key值生成进行匹配，生成规则根据游戏内置属性为定义：如道具，英雄等，

* **程序代码**

1.逻辑日志，错误日志不需要处理

* 需要统一所有日志类接口，打上特定的逻辑段标识：如GamePlay等，不可产生自主接口打印日志
* 方便筛查日志，进而可以减少文字类的信息遗漏

2.特定的Errorcode弹窗和tips

* 1.服务器和客户端通信的的发送tips，如：加点成功的属性增加
* 2.服务器和客户端通信发送操作类的弹窗 如：操作不成功的提示弹窗
* 3.客户端根据服务器提供的ErrorCode显示信息提取

* **带中文字体的美术图片**
* UI贴图

1.图集的图片和背景图片

带有特定字体的图片，命名用于标识，如\_font\_zh\_cn等，这样用于后续工具查漏比较方便

* 特效贴图

和UI贴图一样，区分语言种类，带上特定标识，用于后期工具查漏对比

* **带中文音效文件**
* 音效制作要区分好语言标识，不同语音要放入不同的语言目录下
* **带中文视频文件**
* 视频制作要区分好语言标识，不同语音要放入不同的语言目录下
* **服务器代码中写死的发给客户端显示的内容**

和服务器用同一套的转译规则key，避免产生不必要的key不同的问题

* **技能编辑器产生的资源中包含的中文**

需要提供对应的key封装，不再单独配置文字信息，具体可以建立一套key的配置表

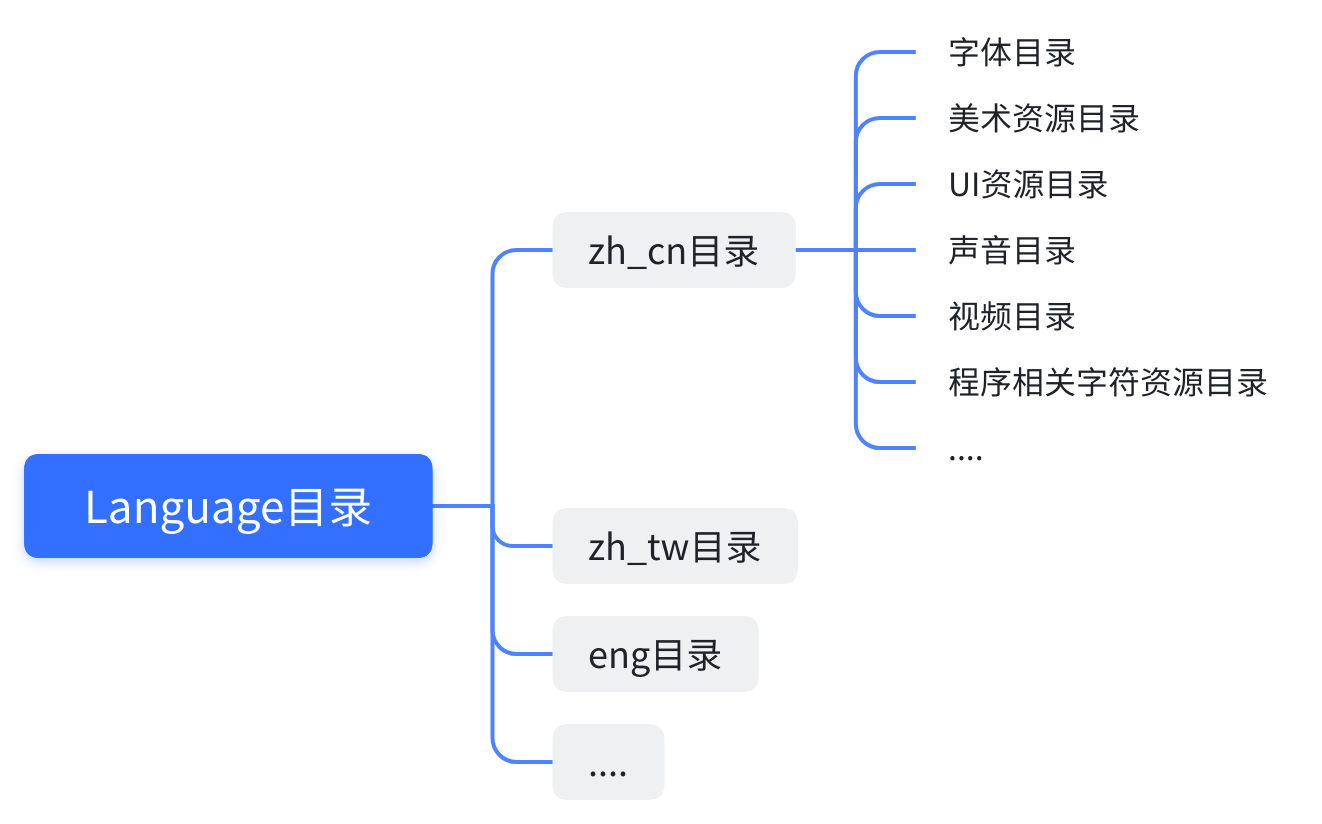
* **剧情编辑器产生的资源中包含的中文**

需要提供对应的key封装，不再单独配置文字信息，具体可以建立一套key的配置表

* **翻译表新旧合并**

新的翻译表产生时候需要和老的转移表进行工具匹配，然后输出新增的翻译表输出，等新的翻译表翻译完成后，用工具合并产生新的翻译表

**目录设计**



目录格式是区分语言种类，建立不同的语言种类去获取不同的文字图片，翻译表，字体库等

**命名设计**

采用统一的内置属性命名规则：

如图片：图片名\_font\_zh\_cn代表的是国服的某个字体图片

如策划字段：角色ID\_HeroName，代表是国服的某个字体图片

程序内置命名规则：(特定的逻辑)\_xxxxx 代表是国服的输出信息提示

如果是错误信息：GameLogic\_ErrorCode001\_zh\_cn,代表是国服游戏逻辑错误提示信息001的

说明：xxxx是客户端自定义会根据语言标识进行组装，因为是有语言区分，会根据不同语言进行切割不同的资源和翻译表，客户端会自行根据语言进行区分的。翻译表只需要翻译之后根据语言进行切割。

**策划表格**

策划表格由各个系统策划进行需要客户端显示的字段进行标记：如在字段头加入lan进行标记，然后工具组进行提取。

**需要关注的问题**

* 1.UI资源的分类处理
* 2.客户端资源分类如：图集，图片，音效，视频
* 3.音效组件和剧情编辑器和演绎编辑器等是否提供接口来支撑

**相关工具**

1.策划翻译标记的文本提取工具

1.能够写入key的前缀，方便输出组合key和文本内容 如角色名字，描述等组合产生翻译唯一key

2.能够输出日记记录，查询提取的配置表的表名，数量，以及输出字段是什么

3.能够去对比老的翻译表，看是否产生新增，新增表输出

2.翻译好的表的新旧区分语言版本工具

1.根据目前版本的语言，和老表进行凭借整合，产生新的表

2.做好筛查重合度，如80%的value重合输出记录能够反馈给策划组进行筛查定稿

3. UI图集的资源各个版本的查漏工具能够根据规则能否及时查找图片是否具备对应的语言版本图片以及命名规则是否符合规范以及存入文件夹是否正确等

4.音效筛查工具类似于UI图集工具

5.视频筛查工具

6.现在版本的代码文字提取工具，输出日志进行代码逻辑接口

7.翻译表各个语言的比对工具，发现是否有缺失不足

**版本发布需求**

1.增加区分概念以及国家概念

a.增加区分概念默认语言模式

b.增加国家概念语言模式

c.每个国家之下可以配置多个语言配置

逻辑层：当玩家所在国家进行语言识别根据国家语言模式进行提取，如果没有配置找区域配置进行语言配置，都没有走统一的默认方案

所有逻辑玩法和配置相关以区服为概念，区分下可以任意组织国家，1对N的关系.在区服下可以任意切换所在国家语言。但是不会切换玩法以及配置资源和数据。

2.资源和配置表的相关

针对于不同国家的美术以及特效、音频，视频等资源如果存在差异性，那么就需要全部分开加上命名标记

配置表走根据区服功能进行配置

3.玩法划分：

玩法和活动根据区服来设定。

4.资源加载流程：

1.打包工具必须根据区服配置表ZoneConfig来配置区，国家，语言字段，然后进行资源重组组合打包成APK

策划配置表以及逻辑层都必须服从区服概念来设计

2.每个区服持有一套默认语言类型的default资源，然后如果有国家层面不同需要配置对应的资源带上国家资源的后缀，不然统一使用default资源为准。

5.多语言区服版本：

每个区服概念都必须有一个分支版本，支持差异化以及本地化的功能开发。然后每个区服下可以支撑多个国家的语言的资源和翻译表等。

主线分支更多支撑的是主线玩法的更新和区服版本进行同步。但是不会影响每个区服差异化的功能。

**区服目录设计**

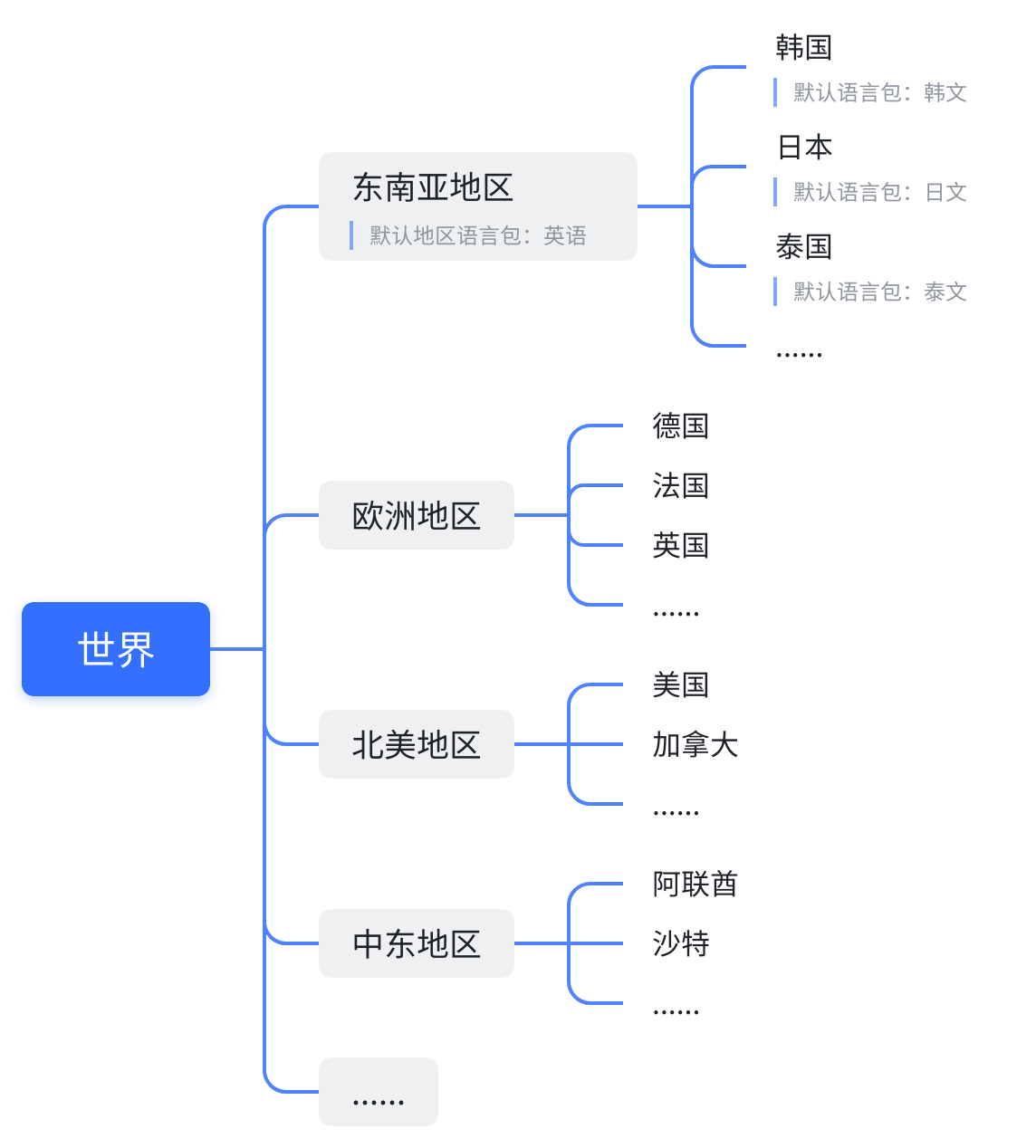
相关目录和多语言目录设计一样，这里只是划分了区的概念，增加了版本服对于逻辑玩法层面的差异化，需要区的玩法逻辑会有一些不同的配置以及显示。

1.特定策划调用资源类：唯一不同的是增加了一个default的文件夹，然后区分了语言种类的文件夹，当所在国家的语言没有配置相关的图片和资源，统一走default的文件夹，获取对应的defualt。其他的UI系统特效资源走多语言系统资源

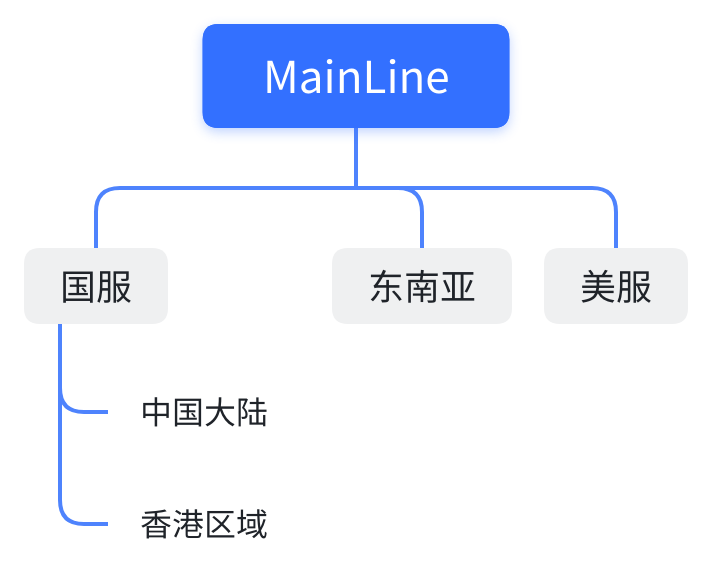
2.多语言的文字Id:

采用没有封装的ID为默认ID，然后去检索是否存在特定语言的ID翻译，如果没有采用defualt的字段为输出。

**世界区域分布设计图**



**区服分支以及打包流程**



* **1.静态配置当前区服分支的Zone以及国家地区概念 然后区服本身有自己的默认语言类型，国家地域也有自己的语言类型，客户端采取从下到上的规则进行筛选，有默认读取没有默认找最上层的读取，这个针对的是一些地区差异化的道具相关的图片资源**
* **2.其他的美术系统和特效资源还是走多语言规则。**
* **3.配置表跟玩法层面是一样的，一套区服共用一套玩法和配置表，不同的区服需要不同的语言表进行区分。**

**客户端资源分类：**

1. **策划表采用多套根据不同区服进行存放**
2. **UI和视频、音频根据对应下的语言类型进行切割**

**特殊说明：可能不同区服下存在相同语言类型，这个资源必须进行合理化划分，不然找不到多语言的资源对应关系，就统一采用default的资源**

**相关工具**

1.策划打表工作扩充：需要提供区服概念的打标

2.差异化增量配置表导出：需要提供策划全量表和差异表的之间做对比实现增删改然后根据新生成的配置表走打标工具生成新的客户端使用匹配表

3.需要提供策划配置表的中文提取（建议是对于做完增删改之后的策划配置表进行），根据对应规则生成翻译表以及相关的key值表,具体key规则到时候是按照中文key还是重新组合，进行商定

4.翻译表查重率工具，对于部分的符号忽略进行文字查重，避免因为标点符号不一样导致key不一样，提供策划进行文字整合

5.打包的工具进行扩展：实现根据配置区服和多语言查证对比进行相关美术资源查证是否漏，上报提醒日志输出和可以实现区服和语言的区分来提取资源进行打包，并能重写客户端标识。也可以做全量打包，不区分区服和语言实现游戏内部逻辑切换。

6.客户端内部的多语言资产查证：音频，特效，UI，视频，每次打包前进行资源筛查，避免因为相关资产不一致导致的错误显示

7.剧情相关的配置Asset能够提取里面的中文字，以及能够转换为翻译表并记录，等翻译表来了之后进行反向替换操作进行key的反向赋值，还有音频等文件的多语言资产替换。

8.多语言的翻译表进行对比，进行多语言翻译是否存在老旧不匹配

9.美术资产对于UI图集的所在目录筛查，避免出现不一致的情况。

10.简单的逻辑端文字提取工具，尽量检查是否存在中文逻辑并打印日志进行相关修改

11.其他待定