|  |
| --- |
| 密级：机密 |
| 总页数：91页 |



**IWT前端框架概要设计**

**概要设计**

**(V 0.1)**

**英泰伟业信息技术股份有限公司**

**应用业务开发部**

**2016年11月**

**版本信息**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **日期** | **版本** | **描述** | **作者/修改人** | **备注** |
| 2016-11-7 | 0.1 | 初稿 | 苏晓燕 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**目 录**

[**(V 0.1)** 1](#_Toc459793659)

[1. 前端框架开发的目标 1](#_Toc459793660)

[1.1. 功能目标 1](#_Toc459793661)

[1.2. 安全、可靠性目标 1](#_Toc459793662)

[1.3. 性能目标 2](#_Toc459793663)

[2. 总体设计 3](#_Toc459793664)

[2.1. 技术体系 3](#_Toc459793665)

[2.2. 基础平台架构 3](#_Toc459793666)

[2.3. 系统总体结构 5](#_Toc459793667)

[2.4. 约定 6](#_Toc459793668)

[2.4.1. 接口设计约定 6](#_Toc459793669)

[2.4.2. Spring使用约定 7](#_Toc459793670)

[2.4.3. Redis应用 11](#_Toc459793671)

[2.4.4. 日志记录约定 29](#_Toc459793672)

[2.4.5. 数据防篡改验证 29](#_Toc459793673)

[2.4.6. 异常处理（修改） 30](#_Toc459793674)

[2.4.7. 分区表约定 31](#_Toc459793675)

[2.5. 浏览器约定 31](#_Toc459793676)

[3. 模型设计 32](#_Toc459793677)

[3.1. 日志类型字典数据初始化 32](#_Toc459793678)

[3.2. 期事件字典 33](#_Toc459793679)

[4. 中彩开奖管理系统接口实现概述 34](#_Toc459793680)

[4.1. 游戏期销售流程图 34](#_Toc459793681)

[4.2. 期状态变化图 35](#_Toc459793682)

[4.3. 期事件许可验证 35](#_Toc459793683)

[4.4. 政策文件下载 36](#_Toc459793684)

[4.4.1. 时序图 36](#_Toc459793685)

[4.4.2. 实现概述 36](#_Toc459793686)

[4.5. 销售数据上报 37](#_Toc459793687)

[4.5.1. 时序图 37](#_Toc459793688)

[4.5.2. 实现概述 37](#_Toc459793689)

[4.6. 新期导入 38](#_Toc459793690)

[4.6.1. 时序图 38](#_Toc459793691)

[4.6.2. 实现概述 38](#_Toc459793692)

[4.7. 开期 39](#_Toc459793693)

[4.7.1. 时序图 39](#_Toc459793694)

[4.7.2. 实现概述 39](#_Toc459793695)

[4.8. 期结、销售数据上报 40](#_Toc459793696)

[4.8.1. 时序图 40](#_Toc459793697)

[4.8.2. 实现概述 40](#_Toc459793698)

[4.9. 下载开奖号码、中奖计算 42](#_Toc459793699)

[4.9.1. 时序图 42](#_Toc459793700)

[4.9.2. 实现概述 42](#_Toc459793701)

[4.10. 下载开奖公告、导入中奖数据 44](#_Toc459793702)

[4.10.1. 时序图 44](#_Toc459793703)

[4.10.2. 实现概述 44](#_Toc459793704)

[4.11. 加载禁销信息 46](#_Toc459793705)

[4.11.1. 时序图 46](#_Toc459793706)

[4.11.2. 实现概述 46](#_Toc459793707)

[4.12. 删除无效禁销信息 47](#_Toc459793708)

[4.12.1. 时序图 47](#_Toc459793709)

[4.12.2. 实现概述 47](#_Toc459793710)

[5. 交易系统/CRM和管理系统接口实现概述 48](#_Toc459793711)

[5.1. 数据上报 48](#_Toc459793712)

[5.1.1. 时序图 49](#_Toc459793713)

[5.1.2. 实现概述 49](#_Toc459793714)

[5.2. 投注 50](#_Toc459793715)

[5.2.1. 时序图 50](#_Toc459793716)

[5.2.2. 实现概述 50](#_Toc459793717)

[5.3. 投注结果查询 52](#_Toc459793718)

[5.3.1. 时序图 52](#_Toc459793719)

[5.3.2. 实现概述 52](#_Toc459793720)

[5.4. 兑奖 53](#_Toc459793721)

[5.4.1. 时序图 53](#_Toc459793722)

[5.4.2. 实现概述 53](#_Toc459793723)

[5.5. 期信息查询 54](#_Toc459793724)

[5.5.1. 时序图 54](#_Toc459793725)

[5.5.2. 实现概述 54](#_Toc459793726)

[5.6. 休市信息查询 55](#_Toc459793727)

[5.6.1. 时序图 55](#_Toc459793728)

[5.6.2. 实现概述 55](#_Toc459793729)

[5.7. 节假日信息查询 56](#_Toc459793730)

[5.7.1. 时序图 56](#_Toc459793731)

[5.7.2. 实现概述 57](#_Toc459793732)

[5.8. 游戏基本参数查询 57](#_Toc459793733)

[5.8.1. 时序图 57](#_Toc459793734)

[5.8.2. 实现概述 57](#_Toc459793735)

[5.9. 投注核对文件 58](#_Toc459793736)

[5.9.1. 时序图 58](#_Toc459793737)

[5.9.2. 实现概述 59](#_Toc459793738)

[5.10. 弃奖文件 60](#_Toc459793739)

[5.10.1. 时序图 60](#_Toc459793740)

[5.10.2. 实现概述 60](#_Toc459793741)

[5.11. 未兑奖文件 62](#_Toc459793742)

[5.11.1. 时序图 62](#_Toc459793743)

[5.11.2. 实现概述 62](#_Toc459793744)

[5.12. 兑奖文件 63](#_Toc459793745)

[5.12.1. 时序图 63](#_Toc459793746)

[5.12.2. 实现概述 63](#_Toc459793747)

[6. 管理系统核心功能实现概述 64](#_Toc459793748)

[6.1. 通用后台管理设计 64](#_Toc459793749)

[6.1.1. 实现目标 64](#_Toc459793750)

[6.1.2. 权限管理设计 65](#_Toc459793751)

[6.2. 设计约定 66](#_Toc459793752)

[6.2.1. 创建类型约定 66](#_Toc459793753)

[6.2.2. 数据查询约定 66](#_Toc459793754)

[6.3. 日志管理 66](#_Toc459793755)

[6.3.1. 系统用户操作日志 66](#_Toc459793756)

[6.3.2. 系统处理日志 67](#_Toc459793757)

[6.4. 销售系统 67](#_Toc459793758)

[6.5. 游戏参数 68](#_Toc459793759)

[6.6. 游戏政策参数 68](#_Toc459793760)

[6.7. 休市查询 68](#_Toc459793761)

[6.8. 节假日查询 68](#_Toc459793762)

[6.9. 业务管理-彩票销售 68](#_Toc459793763)

[6.9.1. 期数据处理模式管理 68](#_Toc459793764)

[6.9.2. 新期参数管理 69](#_Toc459793765)

[6.9.3. 期结处理 70](#_Toc459793766)

[6.9.4. 上报期结数据 70](#_Toc459793767)

[6.9.5. 开奖号码管理 70](#_Toc459793768)

[6.9.6. 中奖计算处理 70](#_Toc459793769)

[6.9.7. 上报中奖数据 70](#_Toc459793770)

[6.9.8. 开奖公告管理 71](#_Toc459793771)

[6.9.9. 开奖公告通知处理 71](#_Toc459793772)

[6.9.10. 数据文件处理 71](#_Toc459793773)

[6.9.11. 游戏期信息查询 71](#_Toc459793774)

[6.9.12. 游戏开奖公告查询 71](#_Toc459793775)

[6.10. 业务监控-彩票销售 72](#_Toc459793776)

[6.10.1. 彩票销售期状态 72](#_Toc459793777)

[6.10.2. 当前销售票 72](#_Toc459793778)

[6.10.3. 历史销售票 73](#_Toc459793779)

[6.10.4. 中奖彩票 73](#_Toc459793780)

[6.10.5. 兑奖彩票 74](#_Toc459793781)

[6.10.6. 弃奖彩票 74](#_Toc459793782)

[6.10.7. 彩票销售期汇总数据 74](#_Toc459793783)

[6.11. 投注账户 74](#_Toc459793784)

[6.11.1. 投注账户信息 74](#_Toc459793785)

[6.12. 资金统计 75](#_Toc459793786)

[6.12.1. 销售资金统计 75](#_Toc459793787)

[6.12.2. 兑奖资金统计 75](#_Toc459793788)

[6.12.3. 弃奖资金统计 75](#_Toc459793789)

[6.12.4. 中奖情况统计 75](#_Toc459793790)

[6.13. 打印制式报表 75](#_Toc459793791)

[6.13.1. 当期中奖结果报告单 75](#_Toc459793792)

[6.13.2. 当前弃奖/销售报告单 76](#_Toc459793793)

[7. 报表计算 76](#_Toc459793794)

[7.1. 报表计算通用核查功能 77](#_Toc459793795)

[7.1.1. 通用核查原则 77](#_Toc459793796)

[7.1.2. 通用核查流程图 78](#_Toc459793797)

[7.1.3. 实现概述 78](#_Toc459793798)

[7.2. 报表中间表统计 79](#_Toc459793799)

[7.2.1. 销售数据中间表统计 79](#_Toc459793800)

[7.2.2. 中奖数据中间表统计 79](#_Toc459793801)

[7.2.3. 兑奖数据中间表统计 80](#_Toc459793802)

[7.2.4. 弃奖数据中间表统计 80](#_Toc459793803)

[7.3. 资金统计 80](#_Toc459793804)

[7.3.1. 销售资金统计 80](#_Toc459793805)

[7.3.2. 中奖情况统计 81](#_Toc459793806)

[7.3.3. 兑奖资金统计 82](#_Toc459793807)

[7.3.4. 弃奖资金统计 82](#_Toc459793808)

# 前端框架开发目标

前端发展迅速的今天，应运而生了许多的前端框架，开发人员在编写前端代码时，从网上找些可以用的代码，或者别的框架的代码粘贴到自己的项目中，这些代码都出自不同的库或框架，造成文件大，版本可能造成冲突，样式不统一，加载慢，渲染慢，执行慢等，前端人员写原始界面，静态页面和特效，开发人员接着改，加逻辑，发现问题要返工，在谁的基础上改和返工，还是合并，造成和很多问题。

IWT编写前端框架的本质是一种管理，对代码的管理。此框架的目的是可以快速的配合开发人员完成开发。达到：加载快，渲染快，执行快，开发快的目的。

结合我们的经验与实际需求，前端框架应实现的功能、达到的性能以及安全、可靠性目标如下：

## 功能目标

前端框架主要功能：前端模板框架、桌面UI框架、移动端UI框架

前端框架可划分为：

1. 后台管理UI框架

基于前端模板开发的javascript函数库，为了能建立一个高互动的网页，采用了Ajax,DHTML和DOM等程式码技术。

1. 移动端UI框架

构建数据驱动的web界面和框架、目标是实现响应式数据绑定和组合的视图插件，其核心是一个响应的数据绑定系统。

## 前端开发安全、可靠性目标

* **Web基本攻击手段**

*Web基本攻击大致可以分为：资源枚举、参数操纵、和其他攻击*

* **安全解决方案**

1. *对用户输入的信息进行编码处理和过滤*
2. *谨慎返回用户输入的信息*
3. *使用黑名单和白名单处理*
4. *检查、验证请求来源，对没一个重要的操作都进行验证*
5. *使用ssl防止第三方通信*
6. *不要将重要文件、备份文件存放在公众可访问到的地方；*
7. *会话ID无序化；*
8. *对用户上传的文件进行验证（不单单是格式验证，比方一张gif图片还应将其转为二进制并验证其每帧颜色值<无符号8位>和宽高值<无符号16位>）；*
9. *WSDL文档应当要求用户注册后才能获取；*

* **安全性：**

*Web的安全性问题大部分依赖于后端的过滤和拦截措施。*

## 前端开发性能目标

* **优化方向**

1. *请求数量：合并脚本和样式表，css Sprites拆分初始化负载，划分主域。*
2. *请求带宽：开启GZip，精简JavaScript，移除重复脚本，图像优化*
3. *缓存利用：使用CDN，使用外部JavaScript和CSS，添加Expires头，减少DNS查找，配置ETag，使AjaX可缓存*
4. *页面结构：将样式表放在顶部，将脚本放在底部，尽早刷新文档的输出*
5. *验证码：避免CSS表达式，避免重定向*
6. *全端工程师可利用yuicompressor 或者 goog closure compiler等压缩工具进行优化。*

# 总体设计

## 技术体系

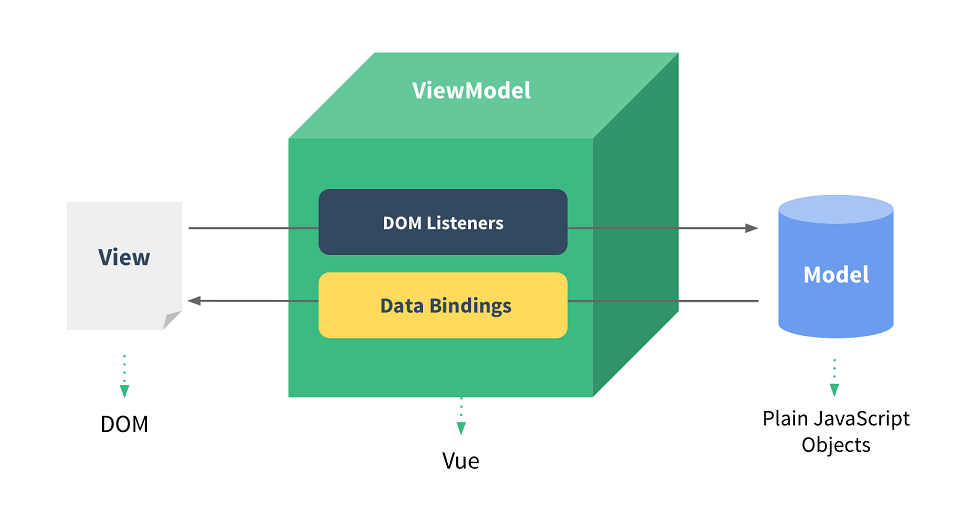
前端框架开发所使用的技术有：

1. Node.js：一个基于 Chrome V8 引擎的 JavaScript 运行环境。
2. Npm：Node.js 的包管理器 npm，是全球最大的开源库生态系统。
3. webpack：自动化。对于需要反复重复的任务，例如压缩（minification）、编译、单元测试、linting等，自动化工具可以减轻你的劳动，简化你的工作。
4. browserify：是一个编译工具,通过它可以在浏览器环境下像nodejs一样使用遵循commonjs规范的模块化编程.
5. Vue.js是一个构建数据驱动的web界面的库。
6. Bootstrap页面重构框架。

## 基础平台架构

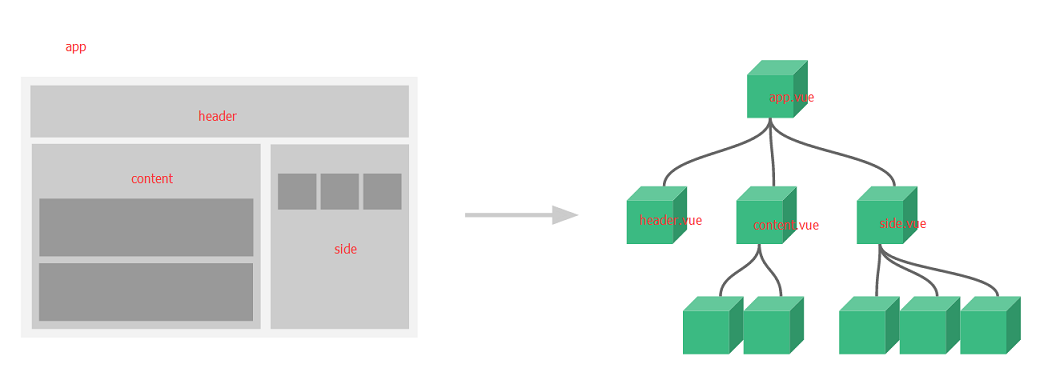
IWT前端框架是英泰伟业开发的性能良好前端框架平台。该框架具有良好的分层、模块化设计的特点。适用于移动端和pc桌面端。具有良好的兼容性和扩展性。

* Vue.js 的核心是一个响应的数据绑定系统，它让数据与 DOM 保持同步非常简单。在使用 jQuery 手工操作 DOM 时，我们的代码常常是命令式的、重复的与易错的。Vue.js 拥抱数据驱动的视图概念。通俗地讲，它意味着我们在普通 HTML 模板中使用特殊的语法将 DOM “绑定”到底层数据。一旦创建了绑定，DOM 将与数据保持同步。每当修改了数据，DOM 便相应地更新。这样我们应用中的逻辑就几乎都是直接修改数据了，不必与 DOM 更新搅在一起。这让我们的代码更容易撰写、理解与维护。

 图2‑1Vue.js结构图

* **组件系统**

组件系统是 Vue.js 另一个重要概念，因为它提供了一种抽象，让我们可以用独立可复用的小组件来构建大型应用。如果我们考虑到这点，几乎任意类型的应用的界面都可以抽象为一个组件树：

****

Web 组件规范仍然远未完成，并且没有浏览器实现。相比之下，Vue.js 组件不需要任何补丁，并且在所有支持的浏览器（IE9 及更高版本）之下表现一致。必要时，Vue.js 组件也可以放在原生自定义元素之内。

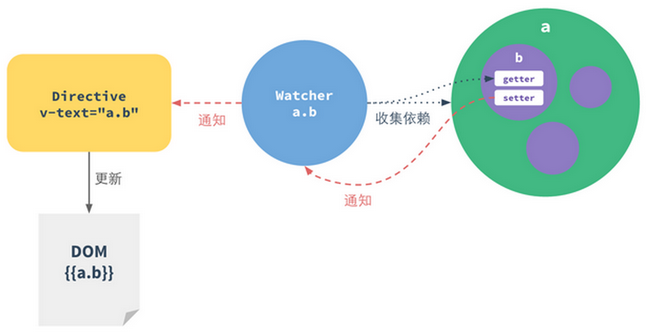
Vue.js 组件提供了原生自定义元素所不具备的一些重要功能，比如组件间的数据流，自定义事件系统，以及动态的、带特效的组件替换。

组件系统是用 Vue.js 构建大型应用的基础。另外，Vue.js 生态系统也提供了高级工具与多种支持库，它们和 Vue.js 一起构成了一个更加“框架”性的系统。

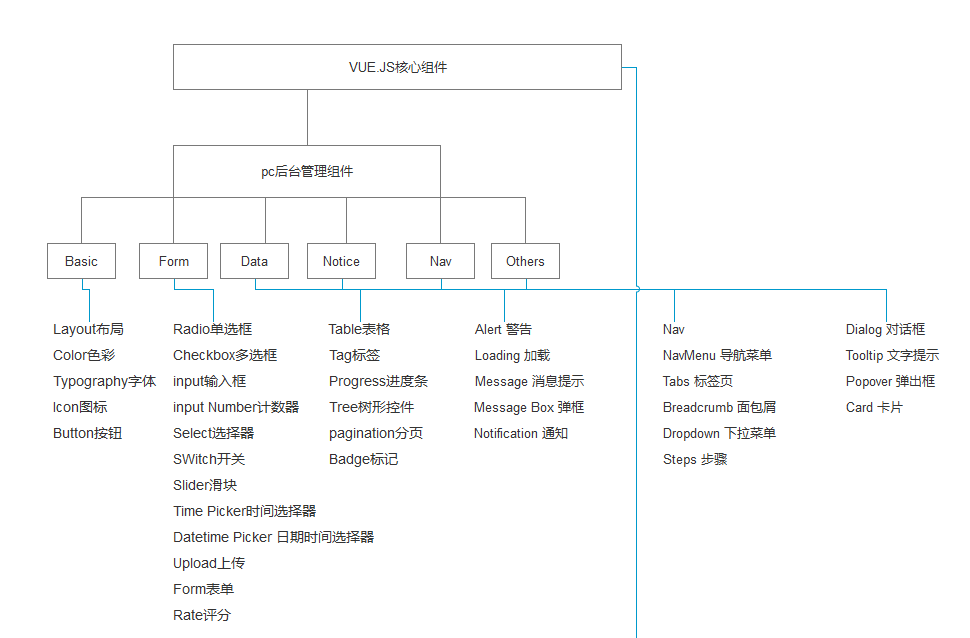
* **响应式原理**

Vue.js采用的则是基于依赖收集的观测机制。从原理上来说，和老牌MVVM框架Knockout是一样的。依赖收集的基本原理是：

1.将原生的数据改造成 “可观察对象”。一个可观察对象可以被取值，也可以被赋值。  
2.在watcher的求值过程中，每一个被取值的可观察对象都会将当前的watcher注册为自己的一个订阅者，并成为当前watcher的一个依赖。  
3.当一个被依赖的可观察对象被赋值时，它会通知所有订阅自己的watcher重新求值，并触发相应的更新。  
4.依赖收集的优点在于可以精确、主动地追踪数据的变化，不存在上述提到的脏检查的两个问题。但传统的依赖收集实现，比如Knockout，通常需要包裹原生数据来制造可观察对象，在取值和赋值时需要采用函数调用的形式，在进行数据操作时写法繁琐，不够直观；同时，对复杂嵌套结构的对象支持也不理想。  
Vue.js利用了ES5的Object.defineProperty方法，直接将原生数据对象的属性改造为getter和setter(这是ES5的特性，需要js解释引擎的支持，无法通过各种打shim补丁来实现。这也是为什么Vue不支持IE8及以下版本的原因)，在这两个函数内部实现依赖的收集和触发，而且完美支持嵌套的对象结构。对于数组，则通过包裹数组的可变方法（比如push）来监听数组的变化。这使得操作Vue.js的数据和操作原生对象几乎没有差别[注:在添加/删除属性，或是修改数组特定位置元素时，需要调用特定的函数，如obj.$add(key, value)才能触发更新。这是受ES5的语言特性所限。在操作对象类型数据的时候一定要注意这点，否则无法实现响应。



## IWT前端框架总体结构



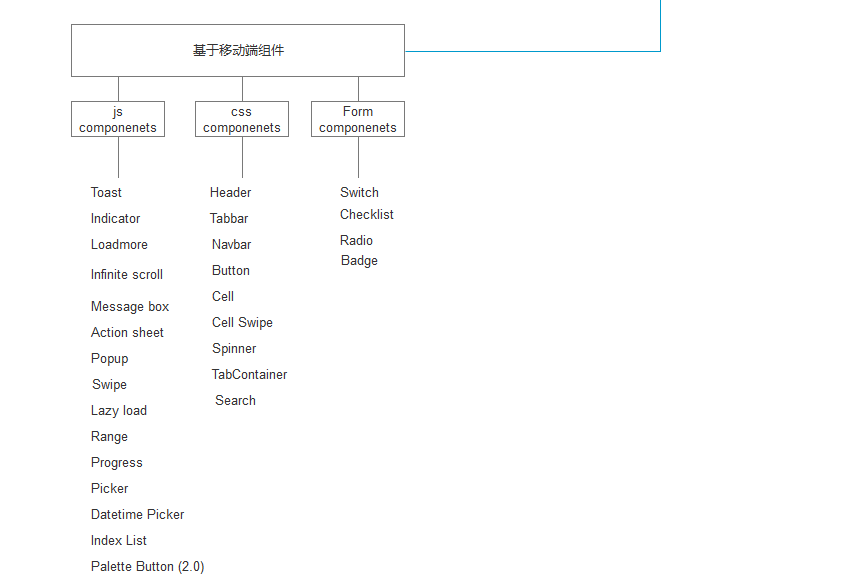


图2‑2游戏引擎系统结构

1. **Pc后台管理组件开发**

**配置文件：**新建项目，项目结构为

|- src/ --------------------- 项目源代码

|- App.vue --------------

|- main.js -------------- 入口文件

|- .babelrc ----------------- babel 配置文件

|- index.html --------------- HTML 模板

|- package.json ------------- npm 配置文件

|- README.md ---------------- 项目帮助文档

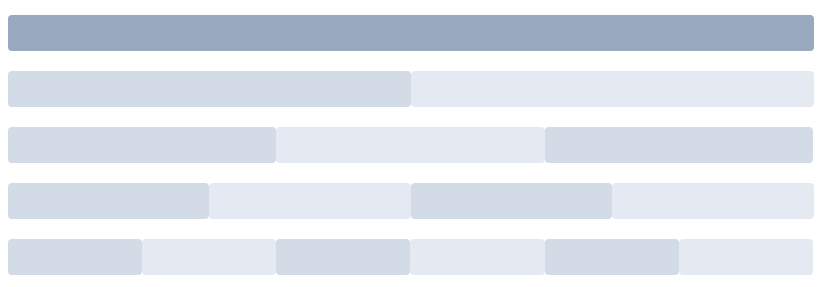
|- webpack.config.json ------ webpack 配置文件

**一、Basic基础**

1. Layout布局

栅格布局：通过基础的24分栏，迅速简便的创建布局。

1.1基础布局



1.2分栏间隔



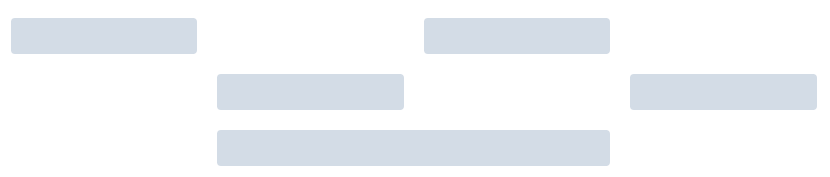
1.3混合布局

通过基础的 1/24 分栏任意扩展组合形成较为复杂的混合布局。



1.4芬兰偏移

支持偏移指定的栏数。



1.5对其方式

对分栏进行灵活的对齐。



1.6行的说明

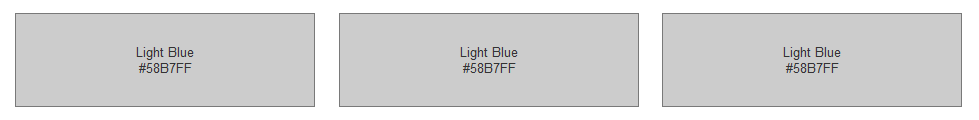
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数 | 说明 | 类型 | 可选值 | 默认值 |
| Gutter | 栅格间隔 | number |  | 0 |
| type | 布局模式，可选 flex，现代浏览器下有效 | string |  |  |
| justify | flex 布局下的水平排列方式 | string | start/  end/  center/  space-around/  space-between | start |
| align | flex 布局下的垂直排列方式 | string | top/middle/bottom | top |

1.7列的说明

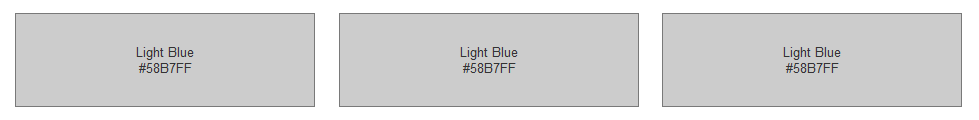
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数 | 说明 | 类型 | 可选值 | 默认值 |
| Span | 栅格占据的列数，**必选参数** | number |  | - |
| offset | 栅格左侧的间隔格数 | number |  | - |
| Push | 栅格向右移动格数 | number | start/  end/  center/  space-around/  space-between | - |
| Pull | 栅格向左移动格数 | number | top/middle/bottom | - |

1. Color色彩的定义：根据公司UI设计进行一下仅供参考

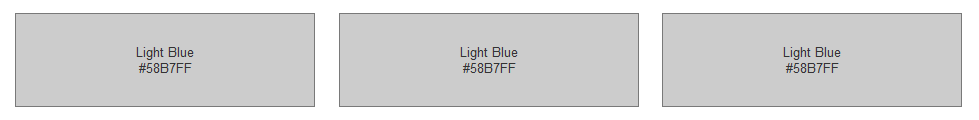
2.1主颜色：



* 1. 辅助颜色



* 1. 中性颜色



1. Typography 字体：

我们对字体进行统一规范，力求在各个操作系统下都有最佳展示效果。

3.1中文字体：

PingFang SC\ Microsoft YaHei\ Hiragino Sans GB

3.2英文字体：

Helvetica Neue\Helvetica\Arial

3.3 Font-family代码

font-family: "Helvetica Neue",Helvetica,"PingFang SC","Hiragino Sans GB","Microsoft YaHei","微软雅黑",Arial,sans-serif;

3.4字体使用规范：

|  |  |  |
| --- | --- | --- |
| 主标题 | 快速搭建页面 | 20px Extra large |
| 标题 | 快速搭建页面 | 18px Extra large |
| 小标题 | 快速搭建页面 | 16px Extra large |
| 正文 | 快速搭建页面 | 14px Extra large |
| 正文（小） | 快速搭建页面 | 13px Extra large |
| 辅助文字 | 快速搭建页面 | 12px Extra large |

1. Icon图标

使用公司自己开发的图标（暂定）

1. 按钮

5.1 基础用法

5.2禁用状态

5.3 有颜色倾向

5.4图标按钮

5.5 按钮组

5.6 记载中

5.7不同尺寸的按钮

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数 | 说明 | 类型 | 可选值 |  |
| Size | 尺寸 | String | large,small,mini |  |
| type | 类型 | String | primary,success,warning,danger,info,text |  |
| plain | 是否朴素按钮 | Boolean | true,false |  |
| disabled | 禁用 | Boolean | true,false |  |
| icon | 图标，已有的图标库中的图标名 | String | button,submit,reset |  |
| Native-type | 原生type属性 | String |  |  |

1. Redis：Redis Server，提供数据缓存服务。
2. 系统DB：Oracle DB，提供数据库服务。

**二、Form 表单**

1、Radio 单选表单

2、Checkbox 多选框

3、Input输入框

4、Input Number 计数器

5、Select 选择器

6、Switch 开关

7、Slider 滑块

8、Time Picker 时间选择器

9、Date Picker 日期选择器

10、Datetime Picker 日期时间选择器

11、Upload 上传

12、Form 表单

13、Rate 评分

**三、Data数据**

1. Table 表格
2. Tag 标签
3. Progress 进度条
4. Tree 树形控件
5. Pagination 分页
6. Badge 标记
7. **Notice 消息框**
8. Alert 警告
9. Loading 加载
10. Message 消息提示
11. Message Box 弹框
12. Notification 通知
13. **NAV导航**
    * + 1. NavMenu 导航菜单
        2. Tabs 标签页
        3. Breadcrumb 面包屑
        4. Dropdown 下拉菜单
        5. Steps 步骤
14. **其他**
    * + 1. Dialog 对话框
        2. Tooltip 文字提示
        3. Popover 弹出框
        4. Card 卡片
15. **移动端UI框架**

**配置文件：**新建项目，项目结构为

|- src/ --------------------- 项目源代码

|- App.vue

|- main.js -------------- 入口文件

|- .babelrc ----------------- babel 配置文件

|- index.html --------------- HTML 模板

|- package.json ------------- npm 配置文件

|- README.md ---------------- 项目帮助文档

|- webpack.config.json ------ webpack 配置文件

1. Js 组件
2. **Toast**简短的消息提示框，支持自定义位置、持续时间和样式。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **参数** | **说明** | **类型** | **可选值** | **默认值** |
| **message** | **文本内容** | String |  |  |
| **position** | **消息框位置** | String | **Top,bottom,middle** | **‘middle’** |
| **duration** | **持续时间（ms）** | Number |  | **3000** |
| **className** | **消息框的类名。可为其添加样式** | String |  |  |
| **iconClass** | **Icon图标的类名** | String |  |  |

1. **Indicator:**加载提示框，支持自定义文本和加载图标。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **参数** | **说明** | **类型** | **可选值** | **默认值** |
| **Text** | **文本内容** | String |  |  |
| **spinnerType** | **加载图标的类型** | String | **Top,bottom,middle** | **‘middle’** |

1. Loadmore
2. Infinite scroll
3. Message box
4. Action sheet
5. Popup
6. Swipe
7. Lazy load
8. Range
9. Progress
10. Picker
11. Datetime Picker
12. Index List
13. Palette Button (2.0)
14. css组件
15. Header
16. Tabbar
17. Navbar
18. Button
19. Cell
20. Cell Swipe
21. Spinner
22. TabContainer
23. Search
24. Form组件
25. Switch
26. Checklist
27. Radio
28. Field
29. Badge
30. **工程划分**
31. 基础工程。
    1. gameEngine1PO：PO对象。
    2. gameEngine1API：Service接口。
    3. gameEngine1RedisAPI：RedisService接口。
    4. gameEngine1SecurityAPI：权限接口。
32. 管理网站工程。
    1. gameEngine1ManageAgent：管理网站Agent。
    2. gameEngine1ManageProcessor：管理网站Processor。
33. 中彩开奖管理系统工程。
    1. gameEngine1DataExchangeAPI：中彩开奖管理系统API。
    2. gameEngine1DataExchangeAgent：中彩开奖管理系统Agent。
    3. gameEngine1DataExchangeProcessor：中彩开奖管理系统Processor。
34. 交易系统&Crm和管理系统工程。
    1. gameEngine1TradingAPI：交易系统API
    2. gameEngine1TradingAgent：交易系统Agent
    3. gameEngine1CrmManageAPI：Crm和管理系统API
    4. gameEngine1CrmManageAgent：Crm和管理系统Agent
    5. gameEngine1CoreProcessor：核心业务Processor，处理交易系统、Crm和管理系统相关业务。
35. 定时任务工程。
    1. gameEngine1ReportProcessor：报表计算Processor
    2. gameEngine1ScheduleProcessor：定时任务Processor。

## 约定

### 接口设计约定

目前游戏引擎与交易系统、CRM和管理系统采用同步接口、文件接口，与中彩开奖管理系统采用文件接口。将来如再增加新的外部系统或内部系统原则上使用异步接口，并遵从如下约定。

#### 异步接口

系统间接口原则上采用异步方式实现，异步方式共分为两个阶段，四个步骤：

第一阶段：

1. 接口调用方按照接口定义要求发送请求数据包。
2. 接口提供方接收请求数据后，进行快捷回复，仅回复是否正确接收数据包。

第二阶段：

1. 接口提供方完成业务处理后，主动发起接口调用，通知接口调用方业务处理结果。
2. 接口调用方接受结果通知数据后，进行快捷回复，回复是否正确接收结果通知数据包。

#### 接口部署访问

所有提供接口服务的部署地址必须位于系统的接入层；

所有对接口的访问发起必须位于接入层；

#### 接口响应码规划

0：成功

1～999是系统代码，由通讯接入Agent统一定义。

999以上由具体业务定义。

### Spring使用约定

#### Spring容器使用约定

Server端使用Spring容器管理Java对象，Processor使用原型、Service配置为单例。Service类必须设计并实现为线程安全对象。

#### Spring声明式事务使用约定

系统采用 Spring声明式事务来管理数据库事务，正确的使用Spring AOP管理数据库事务，有一些基本的概念需要统一思想和认识，使用方法、习惯需要统一约定。

##### Spring AOP管理事务原理

Spring 的声明式事务管理在底层是建立在 AOP 的基础之上的，其本质是对方法前后进行拦截，然后在目标方法开始之前创建或者加入一个事务，在执行完目标方法之后根据执行情况提交或者回滚事务。

##### 事务接口

在 Spring 中，事务是通过 TransactionDefinition 接口来定义的。该接口包含与事务属性有关的方法。具体下所示：

public interface TransactionDefinition{

int getIsolationLevel();

int getPropagationBehavior();

int getTimeout();

boolean isReadOnly();

}

接口只提供了获取属性的方法，而没有提供相关设置属性的方法。Spring 进行事务操作的时候，通过调用以上接口提供的方法必须能够返回事务相关的属性取值。事务属性的设置完全是程序员控制的，可以使用代码设置，也可以通过Spring容器进行配置。

##### Spring数据库事务属性

1. **事务隔离级别**

隔离级别是指若干个并发的事务之间的隔离程度。TransactionDefinition 接口中定义了五个表示隔离级别的常量：

TransactionDefinition.ISOLATION\_DEFAULT：这是默认值，表示使用底层数据库的默认隔离级别。对大部分数据库而言，通常这值就是TransactionDefinition.ISOLATION\_READ\_COMMITTED。

TransactionDefinition.ISOLATION\_READ\_UNCOMMITTED：该隔离级别表示一个事务可以读取另一个事务修改但还没有提交的数据。该级别不能防止脏读和不可重复读，因此很少使用该隔离级别。

TransactionDefinition.ISOLATION\_READ\_COMMITTED：该隔离级别表示一个事务只能读取另一个事务已经提交的数据。该级别可以防止脏读，这也是大多数情况下的推荐值。

TransactionDefinition.ISOLATION\_REPEATABLE\_READ：该隔离级别表示一个事务在整个过程中可以多次重复执行某个查询，并且每次返回的记录都相同。即使在多次查询之间有新增的数据满足该查询，这些新增的记录也会被忽略。该级别可以防止脏读和不可重复读。

TransactionDefinition.ISOLATION\_SERIALIZABLE：所有的事务依次逐个执行，这样事务之间就完全不可能产生干扰，也就是说，该级别可以防止脏读、不可重复读以及幻读。但是这将严重影响程序的性能。通常情况下也不会用到该级别。

1. **事务传播行为**

所谓事务的传播行为是指，如果在开始当前事务之前，一个事务上下文已经存在，此时有若干选项可以指定一个事务性方法的执行行为。在TransactionDefinition定义中包括了如下几个表示传播行为的常量：

TransactionDefinition.PROPAGATION\_REQUIRED：如果当前存在事务，则加入该事务；如果当前没有事务，则创建一个新的事务。

TransactionDefinition.PROPAGATION\_REQUIRES\_NEW：创建一个新的事务，如果当前存在事务，则把当前事务挂起。

TransactionDefinition.PROPAGATION\_SUPPORTS：如果当前存在事务，则加入该事务；如果当前没有事务，则以非事务的方式继续运行。

TransactionDefinition.PROPAGATION\_NOT\_SUPPORTED：以非事务方式运行，如果当前存在事务，则把当前事务挂起。

TransactionDefinition.PROPAGATION\_NEVER：以非事务方式运行，如果当前存在事务，则抛出异常。

TransactionDefinition.PROPAGATION\_MANDATORY：如果当前存在事务，则加入该事务；如果当前没有事务，则抛出异常。

TransactionDefinition.PROPAGATION\_NESTED：如果当前存在事务，则创建一个事务作为当前事务的嵌套事务来运行；如果当前没有事务，则该取值等价于TransactionDefinition.PROPAGATION\_REQUIRED。

这里需要指出的是，前面的六种事务传播行为是 Spring 从 EJB 中引入的，他们共享相同的概念。而 PROPAGATION\_NESTED是 Spring 所特有的。以 PROPAGATION\_NESTED 启动的事务内嵌于外部事务中（如果存在外部事务的话），此时，内嵌事务并不是一个独立的事务，它依赖于外部事务的存在，只有通过外部的事务提交，才能引起内部事务的提交，嵌套的子事务不能单独提交。如果熟悉 JDBC 中的保存点（SavePoint）的概念，那嵌套事务就很容易理解了，其实嵌套的子事务就是保存点的一个应用，一个事务中可以包括多个保存点，每一个嵌套子事务。另外，外部事务的回滚也会导致嵌套子事务的回滚。

1. **事务超时**

所谓事务超时，就是指一个事务所允许执行的最长时间，如果超过该时间限制但事务还没有完成，则自动回滚事务。在 TransactionDefinition 中以 int 的值来表示超时时间，其单位是秒。

1. **事务的只读属性**

事务的只读属性是指对事务性资源进行只读操作或者是读写操作。所谓事务性资源就是指那些被事务管理的资源，比如数据源、 JMS 资源，以及自定义的事务性资源等等。如果确定只对事务性资源进行只读操作，那么我们可以将事务标志为只读的，以提高事务处理的性能。在 TransactionDefinition 中以 boolean 类型来表示该事务是否只读。

1. **事务的回滚规则**

通常情况下，如果在事务中抛出了未检查异常（继承自 RuntimeException 的异常），则默认将回滚事务。如果没有抛出任何异常，或者抛出了已检查异常，则仍然提交事务。这通常也是大多数开发者希望的处理方式，也是 EJB 中的默认处理方式。但是，我们可以根据需要人为控制事务在抛出某些未检查异常时任然提交事务，或者在抛出某些已检查异常时回滚事务。

##### 使用约定

按照Spring AOP事务管理原理和事务属性，对与使用AOP管理事务约定如下：

###### 统一配置数据源、连接池、事务管理器

通过在Spring 配置文件中统一配置连接池和事务管理器，统一控制事务属性。

统一连接池：C3P0，单例。

统一实体工厂：LocalContainerEntityManagerFactoryBean，单例。

统一事务管理器：JpaTransactionManager，单例。

###### AOP配置

通过Spring配置文件统一切面配置，到具体包括以下所有方法。

通过约定方法名命名规则，来约定事务传播行为的配置，并通过配置文件统一配置：

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **传播行为** | **说明** | **方法名命名规则** |
| 1 | REQUIRED | 如果当前存在事务，则加入该事务；如果当前没有事务，则创建一个新的事务 | 方法名加上前缀doTrans |
| 2 | REQUIRES\_NEW | 创建一个新的事务，如果当前存在事务，则把当前事务挂起 | 方法名加上前缀doNewTrans |
| 3 | SUPPORTS | 如果当前存在事务，则加入该事务；如果当前没有事务，则以非事务的方式继续运行 | 方法名加上前缀doJoinTrans |
| 4 | MANDATORY | 如果当前存在事务，则加入该事务；如果当前没有事务，则抛出异常 | 方法名加上前缀doMandTrans |

游戏引擎项目中使用按业务实现需要使用以上事务传播行为，在方法上通过注解声明；其他事物传播行为谨慎使用。

###### Processor与Service实现约定

1. 所有事务在Service方法中开启、提交或回滚，processor不带事务。
2. Service中的所有包含事务行为的方法，需要通过接口的方式进行声明。

###### 使用事务回滚

在Service包含事务传播行为的方法中，如果执行过程发生异常，需要回滚事务，则应抛出指定异常，用来通知Spring AOP事务管理机制回滚事务。

所有开发人员应理解：事务回滚发生在事务开启的方法中，只有内层方法执行出现异常，并向外抛出指定异常，直到调用事务开启的方法中，才能回滚事务；未抛出异常（或捕获异常而不处理，也不抛出），事务不会回滚。

### Redis应用

#### Redis特性简介

Redis是一个开源的key-value存储系统，它通常被称为一个数据结构服务器，keys可以包含strings，hashes，lists，sets和sorted sets。

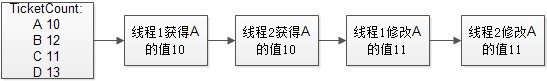
Redis优点是高速。

##### Redis原子性与并发控制

Redis不能达到数据库级别的一致性和可靠性，因此不能取代数据库作为关键应用的数据存储工具。

要保证数据操作的原子性，在传统的RDBMS应用中，通过事物和锁来保证数据的一致性和可靠性。

Redis对单个Key的单步操作是原子的，操作的并发控制依赖与客户端实现，未作并发访问控制会导致更新丢失。



Redis事务不能保证一致性，不会做回滚处理。

#### 游戏引擎的redis使用

作为首次应用Redis的项目，项目团队对Redis的特性与可靠性缺乏实践经验，因此使用原则为仅作为高速辅助缓存，加速业务处理过程；但业务处理过程中不能完全依赖Redis提供的缓存服务，当Redis缓存失效时，应有其他方法来获取正确的数据，并重建缓存，继续业务处理，以保证业务处理的可靠性。

Redis的maxmemory-policy设置为volatile-lru模式。

按照如下步骤添加/修改缓存：

1. 若缓存中有数据，则先删除缓存。如果删除缓存失败，则操作视为失败。
2. 添加/修改数据库数据。如果修改失败，则操作视为失败。
3. 添加/修改缓存。如果添加/修改失败则操作整体视为成功，但系统应立即报警。

按照如下步骤读取缓存：

1. 若缓存中有数据，则直接读取缓存数据。
2. 若缓存没有数据，则从数据库中读取，读取完成后再将数据添加至缓存中。如果添加缓存失败系统应立即报警。

##### Jedis封装

Jedis是Redis的Java版客户端，可通过jedis使用redis提供的服务。

对Jedis的使用，封装到JedisService中，JedisService提供的方法如下（后期可按需要增加新的功能封装接口）：

| **序号** | **方法** | **方法说明** |
| --- | --- | --- |
| 1 | public long expire(String key, int seconds) | /\*\*  \* 设置一个key的过期时间（单位：秒）  \* @param key key值  \* @param seconds 多少秒后过期  \* @return 1：设置了过期时间 0：没有设置过期时间/不能设置过期时间  \*/ |
| 2 | public long expireAt(String key, int unixTimestamp) | /\*\*  \* 设置一个key在某个时间点过期  \* @param key key值  \* @param unixTimestamp unix时间戳，从1970-01-01 00:00:00开始到现在的秒数  \* @return 1：设置了过期时间 0：没有设置过期时间/不能设置过期时间  \*/ |
| 3 | public String trimList(String key, long start, long end) | /\*\*  \* 截断一个List  \* @param key 列表key  \* @param start 开始位置 从0开始  \* @param end 结束位置  \* @return 状态码  \*/ |
| 4 | public long countSet(String key) | /\*\*  \* 检查Set长度  \* @param key  \* @return  \*/ |
| 5 | public boolean addSet(String key, int seconds, String... value) | /\*\*  \* 添加到Set中（同时设置过期时间）  \* @param key key值  \* @param seconds 过期时间 单位s  \* @param value  \* @return  \*/ |
| 6 | public boolean addSet(String key, String... value) | /\*\*  \* 添加到Set中  \*   \* @param key  \* @param value  \* @return  \*/ |
| 7 | public boolean containsInSet(String key, String value) | /\*\*  \* @param key  \* @param value  \* @return 判断值是否包含在set中  \*/ |
| 8 | public Set<String> getSet(String key) | /\*\*  \* 获取Set  \*   \* @param key  \* @return  \*/ |
| 9 | public boolean removeSetValue(String key, String... value) | /\*\*  \* 从set中删除value  \*   \* @param key  \* @return  \*/ |
| 10 | public int removeListValue(String key, List<String> values) | /\*\*  \* 从list中删除value 默认count 1  \*   \* @param key  \* @param values 值list  \* @return  \*/ |
| 11 | public int removeListValue(String key, long count, List<String> values) | /\*\*  \* 从list中删除value \* @param key  \* @param count  \* @param values 值list  \* @return  \*/ |
| 12 | public boolean removeListValue(String key, long count, String value) | /\*\*  \* 从list中删除value  \*   \* @param key  \* @param count 要删除个数  \* @param value  \* @return  \*/ |
| 13 | public List<String> rangeList(String key, long start, long end) | /\*\*  \* 截取List  \* @param key  \* @param start 起始位置  \* @param end 结束位置  \* @return  \*/ |
| 14 | public long countList(String key) | /\*\*  \* 检查List长度  \*   \* @param key  \* @return  \*/ |
| 15 | public boolean addList(String key, int seconds, String... value) | /\*\*  \* 添加到List中（同时设置过期时间）  \*   \* @param key key值  \* @param seconds 过期时间 单位s  \* @param value  \* @return  \*/ |
| 16 | public boolean addList(String key, String... value) | /\*\*  \* 添加到List  \*   \* @param key  \* @param value  \* @return  \*/ |
| 17 | public boolean addList(String key, List<String> list) | /\*\*  \* 添加到List(只新增)  \*   \* @param key  \* @param value  \* @return  \*/ |
| 18 | public List<String> getList(String key) | /\*\*  \* 获取List  \*   \* @param key  \* @return  \*/ |
| 19 | public boolean setHSet(String domain, String key, String value) | /\*\*  \* 设置HashSet对象  \*   \* @param domain  \* 域名  \* @param key  \* 键值  \* @param value  \* Json String or String value  \* @return  \*/ |
| 20 | public String getHSet(String domain, String key) | /\*\*  \* 获得HashSet对象  \*   \* @param domain  \* 域名  \* @param key  \* 键值  \* @return Json String or String value  \*/ |
| 21 | public long delHSet(String domain, String key) | /\*\*  \* 删除HashSet对象  \*   \* @param domain  \* 域名  \* @param key  \* 键值  \* @return 删除的记录数  \*/ |
| 22 | public long delHSet(String domain, String... key) | /\*\*  \* 删除HashSet对象  \*   \* @param domain  \* 域名  \* @param key  \* 键值  \* @return 删除的记录数  \*/ |
| 23 | public boolean existsHSet(String domain, String key) | /\*\*  \* 判断key是否存在  \*   \* @param domain  \* 域名  \* @param key  \* 键值  \* @return  \*/ |
| 24 | public List<Map.Entry<String, String>> scanHSet(String domain, String match); | /\*\*  \* 全局扫描hset  \*   \* @param match  \* field匹配模式  \* @return  \*/ |
| 25 | public List<String> hvals(String domain) | /\*\*  \* 返回 domain 指定的哈希集中所有字段的value值  \*   \* @param domain  \* @return  \*/ |
| 26 | public String getHSet(String domain, String key) | /\*\*  \* 返回 domain 指定的哈希集中所有字段的key值  \*   \* @param domain  \* @return  \*/ |
| 27 | public long lenHset(String domain) | /\*\*  \* 返回 domain 指定的哈希key值总数  \* @param domain  \* @return  \*/ |
| 28 | public boolean setSortedSet(String key, long score, String value) | /\*\*  \* 设置排序集合  \* @param key  \* @param score  \* @param value  \* @return  \*/ |
| 29 | public Set<String> getSoredSet(String key, long startScore, long endScore, boolean orderByDesc) | /\*\*  \* 获得排序集合  \*   \* @param key  \* @param startScore  \* @param endScore  \* @param orderByDesc  \* @return  \*/ |
| 30 | public long countSoredSet(String key, long startScore, long endScore) | /\*\*  \* 计算排序长度  \*   \* @param key  \* @param startScore  \* @param endScore  \* @return  \*/ |
| 31 | public boolean delSortedSet(String key, String value) | /\*\*  \* 删除排序集合  \*   \* @param key  \* @param value  \* @return  \*/ |
| 32 | public Set<String> getSoredSetByRange(String key, int startRange, int endRange, boolean orderByDesc) | /\*\*  \* 获得排序集合  \* @param key  \* @param startRange  \* @param endRange  \* @param orderByDesc  \* @return  \*/ |
| 33 | public Double getScore(String key, String member); | /\*\*  \* 获得排序打分  \*   \* @param key  \* @return  \*/ |
| 34 | public boolean set(String key, String value, int second) ; | 设置key值为value，过期时间second |
| 35 | public boolean set(String key, String value) ; | 设置key值为value |
| 36 | public String get(String key, String defaultValue); | 获取key值 |
| 37 | public boolean del(String key) ; | 删除key |
| 38 | public long incr(String key); | Key值增加1 |
| 39 | public long decr(String key) | Key值减少1 |

##### 配置使用示例（Server端）

* Spring配置

<!—jedisSerivce服务对象-->

<bean id=*"jedisService"* class=*"jedis.test.JedisService"*>

<property name=*"shardedJedisPool"* ref=*"shardedJedisPool"* />

</bean>

<!--连接池-->

<bean id=*"shardedJedisPool"* class=*"redis.clients.jedis.ShardedJedisPool"*>

<constructor-arg index=*"0"* ref=*"jedisPoolConfig"* />

<constructor-arg index=*"1"*>

<list>

<bean name=*"master"* class=*"redis.clients.jedis.JedisShardInfo"*>

<constructor-arg index=*"0"* value=*"192.168.80.53"* />

<constructor-arg index=*"1"* value=*"6379"* type=*"int"* />

</bean>

</list>

</constructor-arg>

</bean>

<!--连接池配置-->

<bean id=*"jedisPoolConfig"* class=*"redis.clients.jedis.JedisPoolConfig"*>

<property name=*"maxTotal"* value=*"2048"* />

<property name=*"maxIdle"* value=*"200"* />

<property name=*"numTestsPerEvictionRun"* value=*"1024"* />

<property name=*"timeBetweenEvictionRunsMillis"* value=*"30000"* />

<property name=*"minEvictableIdleTimeMillis"* value=*"-1"* />

<property name=*"softMinEvictableIdleTimeMillis"* value=*"10000"* />

<property name=*"maxWaitMillis"* value=*"1500"* />

<property name=*"testOnBorrow"* value=*"true"* />

<property name=*"testWhileIdle"* value=*"true"* />

<property name=*"testOnReturn"* value=*"false"* />

<property name=*"jmxEnabled"* value=*"true"* />

<property name=*"jmxNamePrefix"* value=*"youyuan"* />

<property name=*"blockWhenExhausted"* value=*"false"* />

</bean>

* Jedis使用

需要使用*jedisService*的Service、Processor，通过Spring属性注入即可。

##### 部署方式

部署版本：3.0

集群方式：推荐主备部署方式

持久化支持：否

##### 应用场景

缓存的key均以game\_engine1为前缀。Redis缓存每增加一个值均需要讨论，同时需要修改所有与缓存值有关的功能，当值修改时需要同步修改缓存的值。

###### 平台

每个平台必须有授权游戏信息，如果集合数量为0，视为缓存无数据

|  |  |  |
| --- | --- | --- |
| 对应数据库表名 | PLATFORM\_T | |
| Redis数据类型 | 散列型 | |
| 过期时间（分） | 30 | |
| key | game\_engine1\_platform\_【平台ID】 | |
| value | key | value |
| iwoid | IWOID |
| platformId | PLATFORM\_ID |
| platformName | PLATFORM\_NAME |
| virtualTerminalId | VIRTUAL\_TERMINAL\_ID |
| contractBegin | CONTRACT\_BEGIN |
| contractEnd | CONTRACT\_END |
| state | STATE |

|  |  |  |
| --- | --- | --- |
| 对应数据库表名 | PLATFORM\_GAME\_T | |
| Redis数据类型 | 列表型 | |
| 过期时间（分） | 30 | |
| key | game\_engine1\_platform\_game\_【平台ID】 | |
| value | key | value |
| gameOid | GAME\_OID |

###### 地域

|  |  |  |
| --- | --- | --- |
| 对应数据库表名 | SYS\_REGION\_T | |
| Redis数据类型 | 散列型 | |
| 过期时间（分） | 30 | |
| key | game\_engine1\_region\_【地域ID】 | |
| value | key | value |
| iwoid | IWOID |
| regionId | REGION\_ID |
| regionName | REGION\_NAME |
| displayOrder | DISPLAY\_ORDER |
| aliasName | ALIAS\_NAME |
| state | STATE |

###### 游戏信息

|  |  |  |
| --- | --- | --- |
| 对应数据库表名 | GAME\_T | |
| Redis数据类型 | 散列型 | |
| 过期时间（分） | 30 | |
| key | game\_engine1\_game\_【游戏编码】 | |
| value | key | value |
| iwoid | IWOID |
| gameId | GAME\_ID |
| gameName | GAME\_NAME |
| state | STATE |

###### 游戏销售规则

|  |  |  |
| --- | --- | --- |
| 对应数据库表名 | GAME\_SALES\_T | |
| Redis数据类型 | 散列型 | |
| 过期时间（分） | 30 | |
| key | game\_engine1\_game\_sales\_【游戏编码】 | |
| value | key | value |
| iwoid | IWOID |
| gameId | GAME\_ID |
| salesState | SALES\_STATE |
| encashState | ENCASH\_STATE |
| lastSaleDate | LAST\_SALE\_DATE |
| lastSaleIssue | LAST\_SALE\_ISSUE |
| lastEncashDate | LAST\_ENCASH\_DATE |
| lastEncashIssue | LAST\_ENCASH\_ISSUE |
| limitAmount | LIMIT\_AMOUNT |
| limitCount | LIMIT\_COUNT |
| limitMuliple | LIMIT\_MULTIPLE |
| price | PRICE |

###### 游戏当前销售期

|  |  |  |
| --- | --- | --- |
| 对应数据库表名 | ISSUE\_T | |
| Redis数据类型 | 散列型 | |
| 过期时间（分） | 30 | |
| key | game\_engine1\_cur\_issue\_【游戏编码】 | |
| value | key | value |
| iwoid | IWOID |
| issue | ISSUE |
| issueBeginTime | ISSUE\_BEGIN\_TIME |
| issueEndTime | ISSUE\_END\_TIME |
| gameRate | GAME\_RATE |
| state | ISSUE\_STATE |
| encashEndTime | ENCASH\_END\_TIME |
| taxRate | TAX\_RATE |
| taxBegin | TAX\_BEGIN |
| taxEnd | TAX\_END |

###### 游戏期

|  |  |  |
| --- | --- | --- |
| 对应数据库表名 | ISSUE\_T | |
| Redis数据类型 | 散列型 | |
| 过期时间（分） | 30 | |
| key | game\_engine1\_issue\_【游戏编码】\_【期号】 | |
| value | key | value |
| iwoid | IWOID |
| issue | ISSUE |
| issueBeginTime | ISSUE\_BEGIN\_TIME |
| issueEndTime | ISSUE\_END\_TIME |
| gameRate | GAME\_RATE |
| state | ISSUE\_STATE |
| encashEndTime | ENCASH\_END\_TIME |
| taxRate | TAX\_RATE |
| taxBegin | TAX\_BEGIN |
| taxEnd | TAX\_END |

###### 游戏上一期

|  |  |  |
| --- | --- | --- |
| 对应数据库表名 | ISSUE\_T | |
| Redis数据类型 | 散列型 | |
| 过期时间（分） | 30 | |
| key | game\_engine1\_issue\_pre\_【游戏编码】\_【期号】 | |
| value | key | value |
| iwoid | IWOID |
| issue | ISSUE |
| issueBeginTime | ISSUE\_BEGIN\_TIME |
| issueEndTime | ISSUE\_END\_TIME |
| gameRate | GAME\_RATE |
| state | ISSUE\_STATE |
| encashEndTime | ENCASH\_END\_TIME |
| taxRate | TAX\_RATE |
| taxBegin | TAX\_BEGIN |
| taxEnd | TAX\_END |
| preIssueFlag | 0-无上期数据，以上字段均为空字符串。1-有上期数据 |
| exportFlag | EXPORT\_FLAG |

###### 投注方式

|  |  |  |
| --- | --- | --- |
| 对应数据库表名 | SYS\_BET\_TYPE\_T | |
| Redis数据类型 | 散列型 | |
| 过期时间（分） | 30分钟 | |
| key | game\_engine1\_bet\_type\_【投注方式ID】 | |
| value | key | value |
| iwoid | IWOID |
| commonBetType | COMMON\_BET\_TYPE |
| state | STATE |

###### 游戏投注方式

|  |  |  |
| --- | --- | --- |
| 对应数据库表名 | GAME\_BET\_TYPE\_T | |
| Redis数据类型 | 散列型 | |
| 过期时间（分） | 30分钟 | |
| key | game\_engine1\_game\_bet\_type\_【游戏ID】\_【投注方式ID】 | |
| value | key | value |
| iwoid | IWOID |
| maxCount | MAX\_COUNT |
| maxMultiple | MAX\_MULTIPLE |
| minPoint | MIN\_POINT |
| maxPoint | MAX\_POINT |
| state | STATE |

###### 游戏号码投注规则

|  |  |  |
| --- | --- | --- |
| 对应数据库表名 | GAME\_NUMBER\_BET\_T、GAME\_NUMBER\_T | |
| Redis数据类型 | 散列型 | |
| 过期时间（分） | 30分钟 | |
| key | game\_engine1\_game\_number\_bet\_【游戏投注方式OID】【号码分区】 | |
| value | key | value |
| iwoid | IWOID |
| minCount | GAME\_NUMBER\_BET\_T.MIN\_COUNT |
| maxCount | GAME\_NUMBER\_BET\_T.MAX\_COUNT |
| maxBetNumber | GAME\_NUMBER\_T.MAX\_BET\_NUMBER |
| minBetNumber | GAME\_NUMBER\_T.MIN\_BET\_NUMBER |
| minCountTm | MIN\_COUNT\_TM |
| maxCountTm | MAX\_COUNT\_TM |
| tmRuleFlag | TM\_RULE\_FLAG |

###### 游戏号码胆拖投注规则

|  |  |  |
| --- | --- | --- |
| 对应数据库表名 | GAME\_NUMBER\_BET\_TM\_RULE\_T | |
| Redis数据类型 | 散列型 | |
| 过期时间（分） | 30分钟 | |
| key | game\_engine1\_game\_number\_bet\_tm\_rule【游戏投注方式OID】【胆码个数】 | |
| value | key | value |
| iwoid | IWOID |
| number1Count | NUMBER1\_COUNT |
| number2Count | NUMBER2\_MAX\_COUNT |

###### 禁销

定时任务Server启动时自动将有效禁销数据加载至Redis缓存中，如果加载失败则应用启动失败。

定时任务Server增加删除无效禁销缓存任务，每天执行一次。

禁销规则个数与禁销规则OID集合个数不符，则视为缓存无数据。

禁销规则OID集合个数与禁销规则个数不符，则视为缓存无数据。

地域禁销规则数量、游戏禁销规则数量、平台禁销规则数量与缓存数据数量不符，则视为缓存无数据。

|  |  |  |
| --- | --- | --- |
| 描述 | 禁销缓存标识 | |
| Redis数据类型 | 散列型 | |
| 过期时间（分） | 永久 | |
| key | game\_engine1\_ban\_sell\_flag | |
| value | key | value |
| flag | 1-标示缓存数据有效。2-标示缓存数据无效，需要从数据库读取 |
| count | 禁销规则个数 |

|  |  |
| --- | --- |
| 描述 | 存储禁销规则OID列表 |
| Redis数据类型 | 列表型 |
| 过期时间（分） | 永久 |
| key | game\_engine1\_ban\_sell\_oid |
| value | SYS\_BAN\_SELL\_T.IWOID |

|  |  |  |
| --- | --- | --- |
| 对应数据库表名 | SYS\_BAN\_SELL\_T | |
| Redis数据类型 | 散列型 | |
| 过期时间（分） | 永久 | |
| key | game\_engine1\_ban\_sell\_【禁销规则OID】 | |
| value | key | value |
| iwoid | IWOID |
| beginTime | BEGIN\_TIME |
| endTime | END\_TIME |
| zoneType | ZONE\_TYPE |
| gameType | GAME\_TYPE |
| platformType | PLATFORM\_TYPE |
| zoneCount | 地域禁销规则数量 |
| gameCount | 游戏禁销规则数量 |
| platFormCount | 平台禁销规则数量 |
| state | STATE |

|  |  |  |
| --- | --- | --- |
| 对应数据库表名 | SYS\_BAN\_SELL\_GAME\_T | |
| Redis数据类型 | 列表 | |
| 过期时间（分） | 永久 | |
| key | game\_engine1\_ban\_sell\_game\_【禁销规则OID】 | |
| value | key | value |
| gameOId | GAME\_OID |

|  |  |  |
| --- | --- | --- |
| 对应数据库表名 | SYS\_BAN\_SELL\_PLATFORM\_T | |
| Redis数据类型 | 列表 | |
| 过期时间（分） | 永久 | |
| key | game\_engine1\_ban\_sell\_platform\_【禁销规则OID】 | |
| value | key | value |
| platformOid | PLATFORM\_OID |

|  |  |  |
| --- | --- | --- |
| 对应数据库表名 | SYS\_BAN\_SELL\_ZONE\_T | |
| Redis数据类型 | 列表 | |
| 过期时间（分） | 永久 | |
| key | game\_engine1\_ban\_sell\_zone\_【禁销规则OID】 | |
| value | key | value |
| sysRegionOid | SYS\_REGION\_OID |

###### 按平台交易流水号查询当期销售记录

|  |  |  |
| --- | --- | --- |
| 对应数据库表名 | SALE\_DETAIL\_T | |
| Redis数据类型 | 散列型 | |
| 过期时间（分） | 游戏期结（期结时间+1小时） | |
| key | game\_engine1\_cur\_sell\_platform\_【平台ID】\_【交易流水号】 | |
| value | key | value |
| iwoid | IWOID |
| ticketId | TICKET\_ID |
| transNumber | TRANS\_NUMBER |
| channelTransNumber | CHANNEL\_TRANS\_NUMBER |
| beginTime | BEGIN\_TIME |
| state | TICKET\_STATE |
| resultCode | RESULT\_CODE |

###### 按彩票特征码查询当期销售记录

|  |  |  |
| --- | --- | --- |
| 对应数据库表名 | SALE\_DETAIL\_T | |
| Redis数据类型 | 散列型 | |
| 过期时间（分） | 游戏期结（期结时间+1小时） | |
| key | game\_engine1\_cur\_sell\_tickeid\_【游戏编码】\_【期号】\_【特征码】 | |
| value | key | value |
| iwoid | IWOID |
| gameId | GAME\_ID |
| gameName | 创建彩票记录时设置 |
| issueNumber | ISSUE\_NUMBER |
| ticketId | TICKET\_ID |
| regionName | 创建彩票记录时设置 |
| transNumber | TRANS\_NUMBER |
| userId | USER\_ID |
| channelTransNumber | CHANNEL\_TRANS\_NUMBER |
| betDetail | BET\_DETAIL |
| totalMoney | TOTAL\_MONEY |
| wagerType | WAGER\_TYPE |
| beginTime | BEGIN\_TIME |
| ticketState | TICKET\_STATE |
| resultCode | RESULT\_CODE |
| platformName | 创建彩票记录时设置 |

###### 按投资账户号码查询当前销售记录

|  |  |  |
| --- | --- | --- |
| 对应数据库表名 | SALE\_DETAIL\_T | |
| Redis数据类型 | 有序集合 | |
| 过期时间（分） | 游戏期结（期结时间+1小时） | |
| key | game\_engine1\_cur\_sell\_user\_【游戏编码】\_【期号】\_【投注账户】 | |
| score | BEGIN\_TIME（时间戳，数据类型为LONG） | |
| value | key | value |
| iwoid | IWOID |

|  |  |  |
| --- | --- | --- |
| 对应数据库表名 | SALE\_DETAIL\_T | |
| Redis数据类型 | 散列型 | |
| key | game\_engine1 \_sell\_【销售记录OID】 | |
| value | key | value |
| iwoid | IWOID |
| gameId | GAME\_ID |
| gameName | 创建彩票记录时设置 |
| issueNumber | ISSUE\_NUMBER |
| ticketId | TICKET\_ID |
| regionName | 创建彩票记录时设置 |
| transNumber | TRANS\_NUMBER |
| userId | USER\_ID |
| channelTransNumber | CHANNEL\_TRANS\_NUMBER |
| betDetail | BET\_DETAIL |
| totalMoney | TOTAL\_MONEY |
| wagerType | WAGER\_TYPE |
| beginTime | BEGIN\_TIME |
| ticketState | TICKET\_STATE |
| resultCode | RESULT\_CODE |
| platformName | 创建彩票记录时设置 |

### 日志记录约定

#### 日志分类与记录要求

日志分为用户后台操作日志和系统处理日志，用户操作日志通过权限验证时记录日志，需要对所有涉及数据变更操作和数据导出的地方进行记录（即除查询数据外），另外还需要对用户的登录记录日志。

#### 日志数据记录规则

用户后台操作日志操作成功后记录一条日志记录。

系统处理日志处理逻辑如下：

1. 业务处理，记录日志，状态为成功。
2. 业务处理出现异常，数据库事务回滚，开启一个新事务，记录日志，状态为失败

### 数据防篡改验证

系统对

SALE\_DETAIL\_T（投注）

SALE\_DETAIL\_WIN\_T（中奖记录）

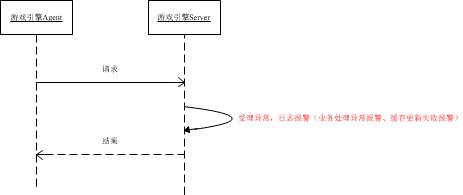
与钱有关的数据表进行了防篡改设计，在生成此类表的数据时，按照算法生成摘要；修改时先验证摘要再修改数据，再按照算法生成新的摘要。

上报至中彩开奖管理系统的数据均需进行防篡改验证。

### 异常处理（修改）

异常处理分为业务处理异常、通讯异常。

#### 业务受理请求异常



受理请求时如果是业务处理文本日志输出【受理请求处理异常，系统报警】

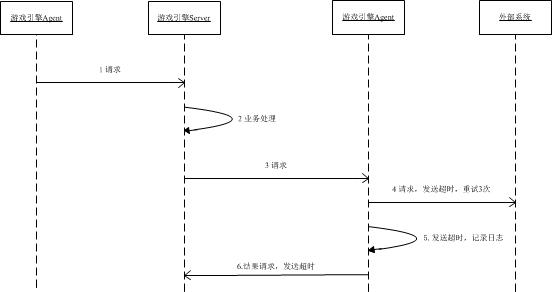
缓存更新文本日志输出【更新缓存失败，系统报警】报警系统监控到此日志内容时需要报警。

期结销售临时表未导出至正式表，报警日志内容为：【销售临时表未导出至正式表】。

日志输出级别WARN。

#### 通讯异常

通讯异常分为发送失败、发送超时、接收超时，采用统一处理策略。日志输出级别WARN。



典型操作步骤：

|  |
| --- |
| **事务1（DB）（processor1）** |
| 1. 创建受理记录 |
| **事务2（DB）（processor1）** |
| 1.业务处理  2.发送请求至外部系统 |
| **事务3（DB）（processor2）** |
| 1.发送失败或超时，Aagent将数据包抛弃，并在文本日志输出【发送处理外部系统请求失败，系统报警】，报警系统监控到此日志内容时需要报警。此时交易记录状态为处理中 |

### 分区表约定

为了提高系统处理，将彩票销售记录（SALE\_DETAIL\_T）、彩票只奖记录（SALE\_DETAIL\_WIN\_T），彩票兑奖（SALE\_DETAIL\_ENCASH\_DETAIL\_T）记录按月进行了分区，查询这三张表必须的where条件必须加上分区列。

SALE\_DETAIL\_T表的分区列：BEGIN\_TIME。

SALE\_DETAIL\_WIN\_T表的分区列：BEGIN\_TIME。

SALE\_DETAIL\_ENCASH\_DETAIL\_T表的分区列：ENCASH\_TIME。

如果安装按照主键、唯一索引查询不受此限制。

## 浏览器约定

开发人员在开发过程中必须在以下浏览器中进行调试。浏览器类别如下：

IE8-IE11，兼容模式。

360安全模式。

火狐浏览器。

谷歌浏览器。

# 模型设计

数据库模型详细信息参考《GAME\_ENGINE数据库设计.PDM》。

数据库中所有金额字段在没有特殊说明的情况下单位均为分。

游戏政策参数比例均按照乘以10000存储，例如20%，在数据库里的值为200。

## 日志类型字典数据初始化

|  |  |
| --- | --- |
| **ID** | **类型名称** |
| ISSUE\_CLOSE\_TO\_FILE | 期结数据汇总 |
| UPLOAD\_ISSUE\_DATA | 期结数据上报 |
| TEMP\_SELL\_IMPORT | 销售临时表导出 |
| SELL\_MERGE | 销售数据上报合并 |
| DOWNLOAD\_OPEN\_NUMBER | 下载开奖号码 |
| WIN\_COMPUTE | 中奖计算 |
| UPLOAD\_WIN\_DATA | 中奖统计上报 |
| UPLOAD\_WIN\_MEGRE | 中奖统计上报合并 |
| DOWNLOAD\_WIN\_INFO | 下载开奖公告 |
| UPDATE\_WIN\_DATA | 更新开奖信息 |
| UPLOAD\_BONUS | 中奖明细上报 |
| UPLOAD\_BONUS\_ MEGRE | 中奖明细上报合并 |
| PUBLISH\_WIN\_INFO | 发布开奖公告 |
| PUBLISH\_WIN\_DATA | 中奖数据发布 |
| USER\_ACTION | 用户操作 |
| USER\_LOGIN\_CODE | 用户登录 |
| USER\_LOGOUT\_CODE | 用户退出 |
| CRETE\_ISSUE | 新期参数导入 |
| OPEN\_ISSUE | 开期 |
| END\_ISSUE | 期结 |
| ABANDON\_COMPUTE | 弃奖计算 |
| BET\_IMPORT | 投注文件核对 |
| ABANDON\_IMPORT | 弃奖文件核对 |
| UN\_ENCASH\_IMPORT | 未兑奖文件核对 |
| ENCASH\_IMPORT | 兑奖文件核对 |
| SELL\_BAN\_REDIS\_FLUSH | 禁销缓存刷新 |
| SELL\_BAN\_QUERY | 休市信息查询 |
| HOLIDAYS\_QUERY | 节假日信息查询 |
| GAME\_POLICY\_QUERY | 游戏基本参数查询 |
| DATA\_IMPORT | 数据导出 |

## 期事件字典

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 当前状态 | 检查事件码 | 重复执行事件码 | 事件码 |
| 新期参数初始化 | 初始化 | 无 | 开期期处理 | 1 |
| 开期处理 | 1 | 期结处理 | 10 |
| 期结处理 | - | 10 | 下载开奖号码 | 20 |
| 弃奖计算 | 期结（中奖明细数据合并不用验证期状态） | 20 | 30 |
| 期汇总处理 | 30 | 40 |
| 期结数据上报 | 40 | 50 |
| ~~期结数据合并~~ | ~~50~~ | ~~60~~ |
| 下载开奖号码 | 50 | 可兑奖（中奖明细数据合并为兑奖结束） | 70 |
| 计算中奖 | 70 | 80 |
| 中奖数据上报 | 80 | 90 |
| ~~中奖数据合并~~ | ~~90~~ | ~~100~~ |
| 下载开奖公告 | 90 | 110 |
| 销售临时表导出 | 110 | 120 |
| 中奖处理 | 120 | 130 |
| 中奖明细数据上报 | 130 | 140 |
| ~~中奖明细数据合并~~ | ~~140~~ | ~~150~~ |
| 可兑奖 | 140 | 160 |
| 发布开奖公告 | 已开奖 | 160 | 兑奖结束 | 170 |
| 发布中奖数据 | 160 | 180 |
| 兑奖结束 | 兑奖结束 | 180 | 无 | 190 |
| 投注文件核对 | 无 | 无 | -1 |
| 弃奖文件核对 | -10 |
| 未兑奖文件核对 | -20 |
| 兑奖文件核对 | -30 |

# 中彩开奖管理系统接口实现概述

## 游戏期销售流程图

中国福利彩票游戏按期销售，游戏销售每期的流程如下：

图4-1游戏期销售流程图

上图是以期为核心讲述游戏期的整个流程，在实际运营中，不同期的期状态可以并行发生的。例如双色球2016001期正在进行中奖计算，2016002期已经处于销售中。

## 期状态变化图



图4-2期状态变化图

## 期事件许可验证

期事件的每一步操作（开期、期结、中奖计算、更新浮动奖等，新期导入除外）都需要验证期事件的操作许可，验证内容主要包含两方面：

1. 操作模式（自动/手动）验证。
2. 前置事件是否已完成。例如：若要执行期结操作，则开期操作必须已成功执行；若要执行开期操作，则新期导入操作已成功执行。以此类推。
3. 实现概述如下：
   1. 根据事件码、期查询期事件执行许可（SYS\_ISSUE\_EVENT\_EXE\_T），执行许可的操作模式必须与操作触发时的操作模式一致。例如系统执行自动期结操作，但执行许可的操作模式为手动，则系统不执行期结操作。
   2. 执行许可的期状态（SYS\_ISSUE\_EVENT\_EXE\_T.SYS\_ISSUE\_EVENT\_OID.ISSUE\_STATE）必须与当前期状态一致（手动模式的期结操作除外）。
   3. 根据事件码、期查询期事件处理记录（ISSUE\_EVENT\_T），处理状态查询条件为：自动模式处理状态为处理中、完成，手动模式处理状态为处理中。如果记录存在则不允许执行。
   4. 根据前置事件码（SYS\_ISSUE\_EVENT\_T.BEFOR\_EVENT\_CODE）、期信息查询期事件处理记录（ISSUE\_EVENT\_T），状态必须已完成，如果数据不存在则不允许执行。例如新期导入都没执行成功，期结操作则不允许执行。
   5. 如果重复事件码不为-1（SYS\_ISSUE\_EVENT\_T.FORBID\_REPEAT\_AFTER\_EVENT\_CODE），则根据重复事件码、期信息查询期事件处理记录（ISSUE\_EVENT\_T），状态为处理中、完成，如果数据存在则不允许执行。例如已完成中奖计算，则不允许再执行期结操作。
   6. 如果是开期操作，则当前时刻不能有2个同时销售的期信息。
   7. 如果当前事件码之后有期事件处理记录，则统一删除。

## 政策文件下载

新期参数文件、开奖号码、开奖公告下载时使用此时序完成文件下载。

### 时序图

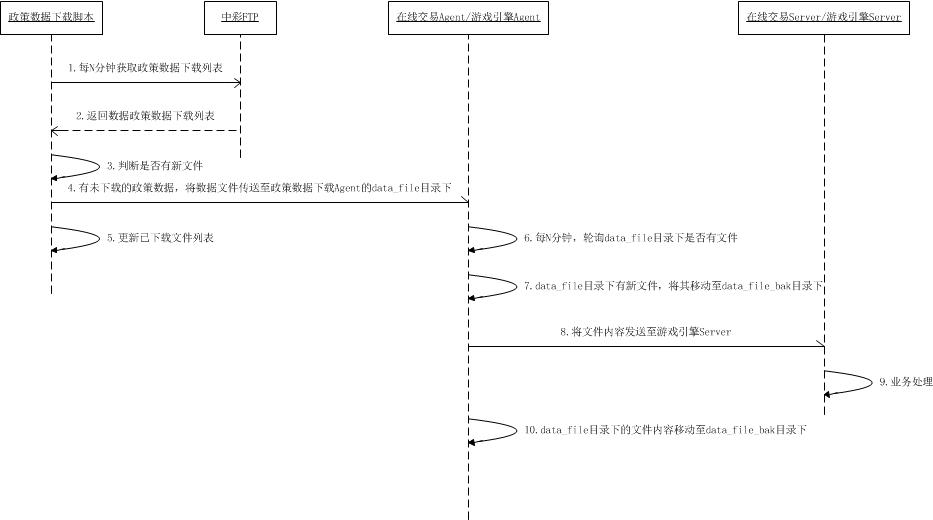


图4-3政策文件下载时序图

### 实现概述

1. 政策数据下载脚本每隔N分钟从中彩FTP获取文件列表，并与本地已下载文件列表进行比对，如果未下载，则从中彩FTP获取文件。
2. 政策数据下载脚本将文件分别存放至在线交易系统Agent、游戏引擎Agent的data\_file目录下。
3. 在线交易系统Agent、游戏引擎Agent每隔N分钟轮询data\_file目录下是否有文件，如果有则将文件内容发送至Server进行业务处理，并将data\_file目录下的文件移动至data\_file\_bak目录下。

## 销售数据上报

中彩开奖管理系统的期结销售/弃奖数据上报、中奖统计/兑奖明细上报、中奖明细上报使用此时序完成上报，交易系统/CRM和管理系统的开奖公告上报、中奖数据上报参考第五章节的【数据上报】时序的第三小节时序。

时序图中的标志文件名的定义与上报数据文件名一致，文件内容为空。

### 时序图

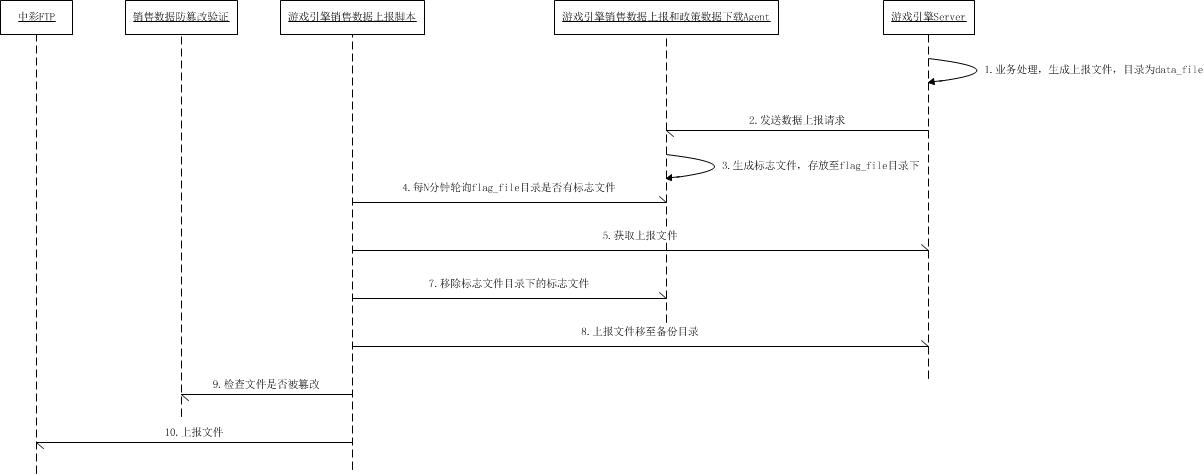


图4-4销售数据上报时序图

### 实现概述

1. 游戏引擎生成上报文件。
   1. Server生成上报原始文件，文件存储在cwl\_data\_file目录下。
   2. 发送上报请求至Agent，Agent在flag\_file目录下生成标志文件。
2. 上报文件。
   1. 数据上报脚本每隔N分钟轮询上报Agent的flag\_file目录下是否有标志文件，如果有则从Server的cwl\_data\_file目录获取文件，调用上报文件验证程序验证上报文件是否被篡改，如果未被篡改则将文件上报至中彩FTP。
   2. 删除标志文件，上报文件移至备份目录。
3. 特别约定。
   1. 期结上报完成后，游戏引擎中奖统计文件暂不删除，以便手动任务可以重复执行。

## 新期导入

### 时序图

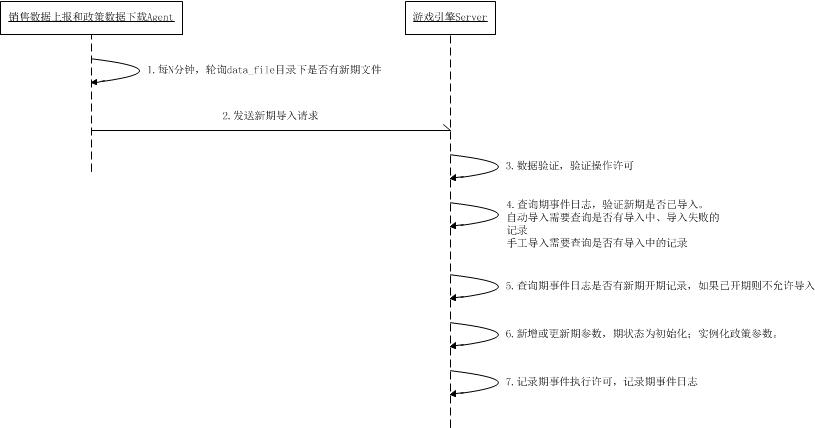


图4-5新期导入时序图

### 实现概述

1. 期事件执行许可验证。
   1. 查询期事件日志，验证新期是否已导入。自动导入需要查询是否有导入中、导入失败的记录。手工导入需要查询是否有导入中的记录。
   2. 查询期事件日志是否有新期开期记录，如果已开期则不允许导入。
2. 重置政策参数比例。
3. 检查数据文件中的促销码是否在系统中存在，即GAME\_PROMOTION\_T表里是否有促销码记录，PROMOTION\_ID为促销码。
4. 创建新期记录，期状态为初始化。
5. 创建期事件执行许可（SYS\_ISSUE\_EVENT\_EXE\_T）。
6. 创建期事件执行记录（ISSUE\_EVENT\_T）。
7. 创建期事件日志（LOG\_EVENT\_T）。

## 开期

### 时序图



图4-6开期时序图

### 实现概述

1. 期事件许可验证。
2. 创建期事件处理记录（ISSUE\_EVENT\_T、LOG\_EVENT\_T），状态为处理中。
3. 从缓存中删除当前销售期、游戏期信息。
4. 期状态设置为销售。
5. 创建销售临时表（SALE\_DETAIL\_GAME\_ISSUE\_T），例如双色球2016001期则销售临时表名为SALE\_DETAIL\_10001\_2016001。
6. 更新期事件处理记录状态。
7. 在Redis缓存中添加游戏当前销售期、游戏期缓存数据。

## 期结、销售数据上报

### 时序图

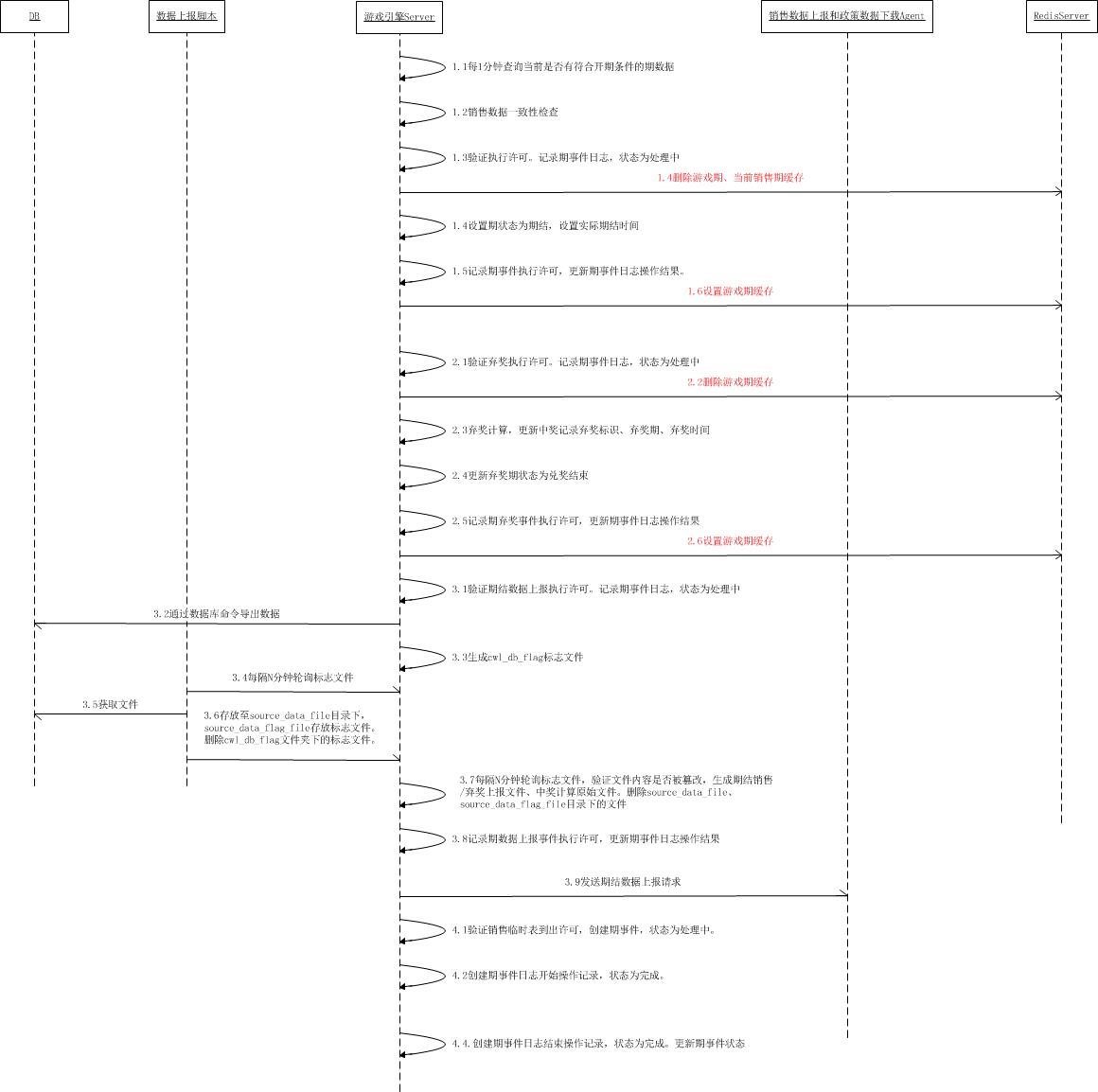


图4-7期结、销售数据上报时序图

### 实现概述

1. 销售数据一致性检查。
   1. 按BEGIN\_TIME倒叙从SALE\_DETAIL\_GAME\_ISSUE\_T获取IWOID。
   2. 间隔15秒中按照上一步逻辑再获取SALE\_DETAIL\_GAME\_ISSUE\_T.IWOID
   3. 如果上两步操作获取的数据一致则检查通过，否则继续从头执行，如果执行次数超过5此则终止期结操作。
2. 期结处理。
   1. 期事件许可验证。
   2. 创建期事件处理记录（ISSUE\_EVENT\_T），状态为处理中。
   3. 删除Redis游戏当前销售期、游戏期数据。
   4. 创建期时间日志开始操作记录（LOG\_EVENT\_T），状态为完成。
   5. 期状态设置为已期结。
   6. 更新期事件处理记录状态。
   7. 创建期事件日志结束操作记录（LOG\_EVENT\_T），状态为完成。
   8. 删除source\_data\_file、source\_data\_flag\_file目录下的文件。
   9. 设置Redis游戏期数据。
3. 弃奖处理。
   1. 期事件许可验证。
   2. 创建期事件处理记录（ISSUE\_EVENT\_T），状态为处理中。
   3. 删除Redis游戏游戏期数据。
   4. 创建期事件日志开始操作记录（LOG\_EVENT\_T），状态为完成。
   5. 获取期结期的弃奖起始期号，弃奖数量计算出最小开期时间，最大期结时间
   6. 从SALE\_DETAIL\_WIN\_T获取符合弃奖条件的记录，进行弃奖操作。
      1. 弃奖条件：
         1. BEGIN\_TIME区间范围为最小开期时间，最大期结时间（当日24点）。
         2. GAME\_ID：当前期结期的游戏ID。
         3. ABANDON\_TIME小于当前系统时间。
         4. ENCASH\_FLAG等于0。
         5. ABANDON\_FLAG等于0。
      2. 更新值：
         1. ABANDON\_FLAG设置为1。
         2. ABANDON\_ISSUE\_OID：当前期结期OID。
         3. ABANDON\_ISSUE：当前期结期期号。
         4. ABANDON\_AMOUNT：TOTAL\_WIN\_AMOUNT。
         5. SYSTEM\_ABANDON\_TIME：当前系统时间。
   7. 更新弃奖期的期状态为兑奖结束，更新期事件处理记录状态。
   8. 创建期时间日志结束操作记录（LOG\_EVENT\_T），状态为完成。
   9. 设置Redis游戏期数据。
4. 期结数据上报。
   1. 期事件许可验证。
   2. 创建期事件处理记录（ISSUE\_EVENT\_T），状态为处理中。
   3. 创建期事件日志开始操作记录（LOG\_EVENT\_T），状态为完成。
   4. 使用数据库命令从SALE\_DETAIL\_GAME\_ISSUE\_T导出数据，投注状态必须为成功，付费类型为付费。并在cwl\_db\_flag生成标志文件。
   5. 数据上报脚本每隔N分钟轮标志文件，从数据库服务器将文件下载至source\_data\_file文件夹下，并在source\_data\_flag\_file文件下生成标志文件。删除cwl\_db\_flag文件夹下的标志文件。
   6. 每隔N分钟轮询source\_data\_flag\_file文件夹下是否有标志文件，生成销售上报文件、弃奖上报文件、中奖计算文件，记录中奖文件记录（ISSUE\_COMPUTE\_FILE\_T）。
   7. 如果是自动任务则删除source\_data\_file、source\_data\_flag\_file目录下的文件。手动任务不删除source\_data\_file、source\_data\_flag\_file目录下的文件，以便手动任务可以重复执行。。
   8. 创建期事件日志结束操作记录（LOG\_EVENT\_T），状态为完成。更新期事件处理状态。

## 下载开奖号码、中奖计算

### 时序图

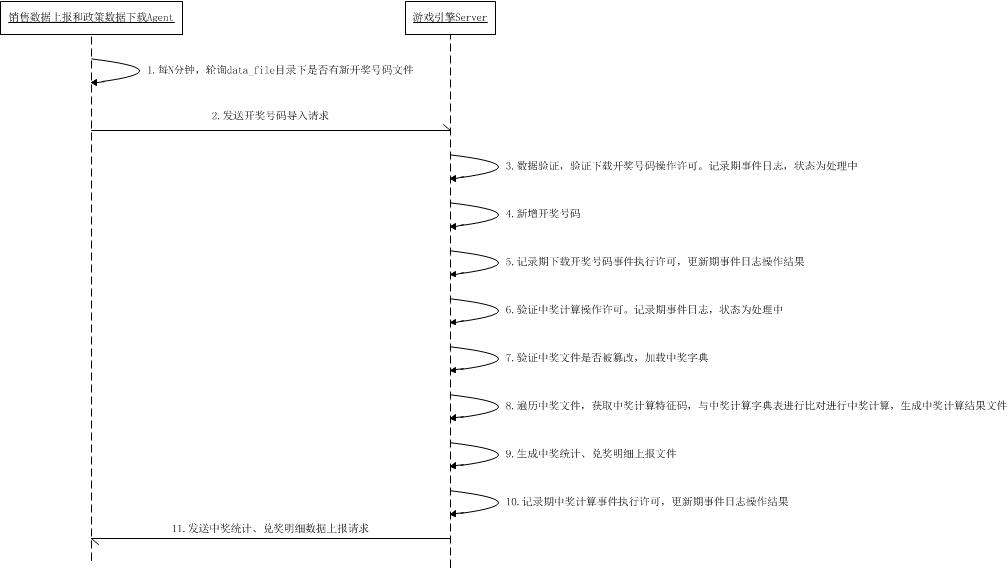


图4-8下载开奖号码、中奖计算时序图

### 实现概述

1. 开奖号码录入。
   1. 期事件许可验证。
   2. 创建期事件处理记录（ISSUE\_EVENT\_T），状态为处理中。
   3. 创建期事件日志操作记录（LOG\_EVENT\_T），状态为处理中。
   4. 新增开奖号码。（ISSUE\_OPEN\_T、ISSUE\_OPEN\_GROUP\_T）。
   5. 清空销售临时表。
   6. 更新期事件处理记录状态。
   7. 更新期事件日志状态。
   8. 删除兑奖明细原始文件。
   9. 删除商业积分换彩票期结上报文件。
   10. 删除期结原始文件。
2. 中奖计算。
   1. 期事件许可验证。
   2. 创建期事件处理记录（ISSUE\_EVENT\_T），状态为处理中。
   3. 创建期事件日志开始操作记录（LOG\_EVENT\_T），状态为完成。
   4. 获取中奖计算文件记录（ISSUE\_COMPUTE\_FILE\_T）。
   5. 验证中奖记录文件计算标志（COMPUTE\_FLAG）。自动模式状态应为未计算，手动模式为未计算、已计算。
   6. 验证中奖计算文件是否被篡改
   7. 获取开奖号码（ISSUE\_OPEN\_T）、游戏奖等（GAME\_GRADE\_T）
   8. 加载中奖字典（SYS\_GAME\_COMPUTE\_DICT\_T）。
   9. 遍历中奖计算文件，验证每条销售记录是否被篡改。
   10. 根据每条销售记录生成中奖索引，与中奖字典比对，如果存在则说明中奖，该彩票每个奖等的中奖注数为中奖字典里的每个奖等的中奖注数。
   11. 生成中奖结果文件、中奖统计上报文件
   12. 中奖计算文件记录计算标识设置为已计算。
   13. 更新期事件处理记录状态。
   14. 创建期事件日志结束操作记录（LOG\_EVENT\_T），状态为完成。
3. 兑奖明细上报文件。
   1. 期事件许可验证。
   2. 创建期事件处理记录（ISSUE\_EVENT\_T），状态为处理中。
   3. 创建期事件日志开始操作记录（LOG\_EVENT\_T），状态为完成。
   4. 使用数据库命令从SALE\_DETAIL\_WIN\_T导出兑奖数据，并在cwl\_db\_flag生成标志文件。
   5. 数据上报脚本每隔N分钟轮标志文件，从数据库服务器将文件下载至source\_data\_file文件夹下，并在source\_data\_flag\_file文件下生成标志文件。删除cwl\_db\_flag文件夹下的标志文件。
   6. 每隔N分钟轮询source\_data\_flag\_file文件夹下是否有标志文件，生成兑奖明细上报文件
   7. 如果是自动任务则删除source\_data\_file、source\_data\_flag\_file目录下的文件。手动任务不删除source\_data\_file、source\_data\_flag\_file目录下的文件，以便手动任务可以重复执行。
   8. 创建期事件日志结束操作记录（LOG\_EVENT\_T），状态为完成。更新期事件处理状态。

## 下载开奖公告、导入中奖数据

### 时序图

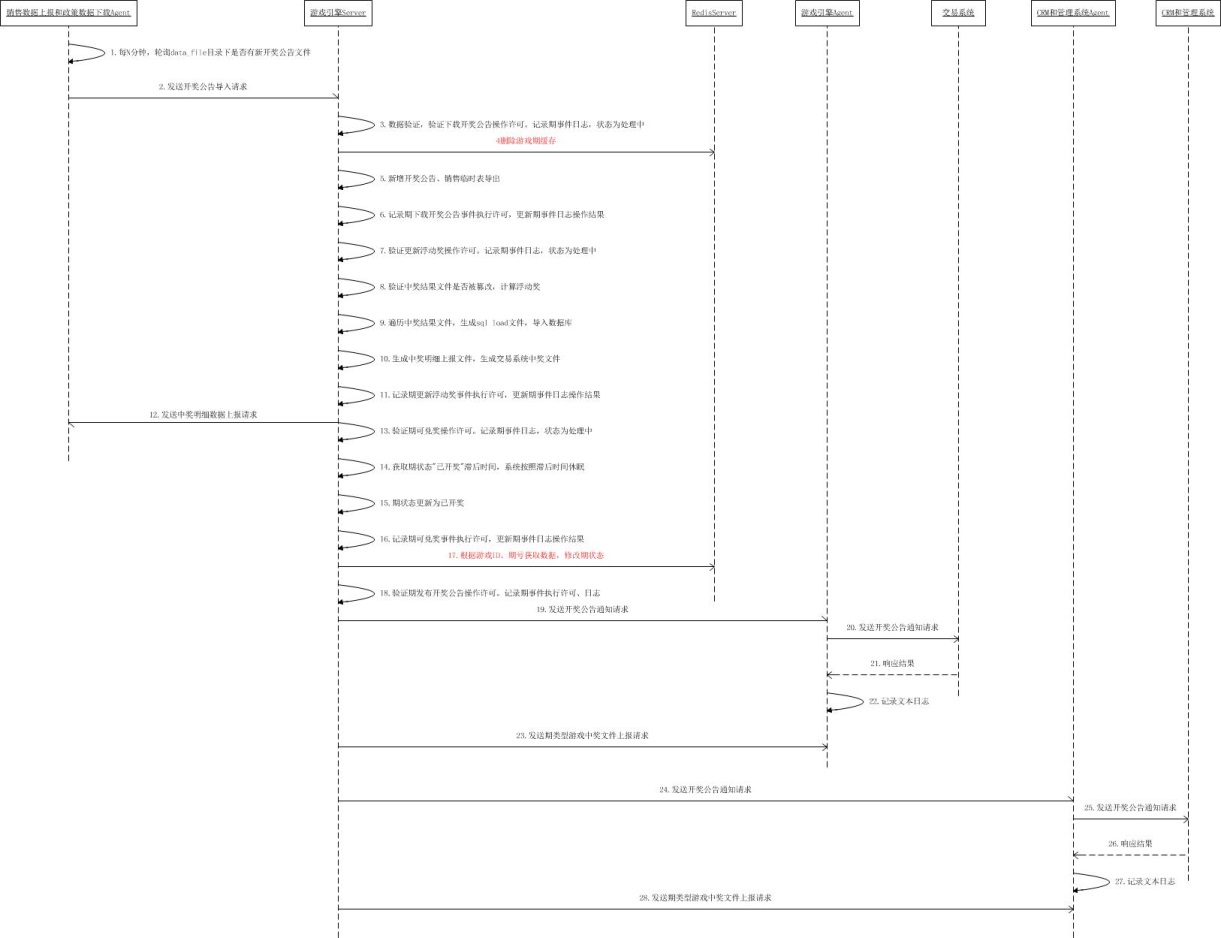


图4-9下载开奖公告、导入中奖数据时序图

### 实现概述

1. 更新开奖公告、中奖导入。
   1. 期事件许可验证。
   2. 创建更新开奖公告、中奖导入期事件处理记录（ISSUE\_EVENT\_T），状态为处理中。
   3. 删除缓存游戏期数据。
   4. 创建更新开奖公告期事件日志开始操作记录（LOG\_EVENT\_T），状态为完成。
   5. 更新开奖号码的销售额、奖池数据（ISSUE\_OPEN\_T），新增开奖公告（ISSUE\_OPEN\_GRADE\_T）。
   6. 更新更新开奖公告期事件处理记录状态。
   7. 创建更新开奖公告期事件日志结束操作记录（LOG\_EVENT\_T），状态为完成。
   8. 创建中奖导入期事件日志开始操作记录（LOG\_EVENT\_T），状态为完成。
   9. 删除该期中奖记录（SALE\_DETAIL\_WIN\_T）BEGIN\_TIME为该期的开始时间、期结时间（当日的24点）。
   10. 验证中奖结果文件是否被篡改。
   11. 根据开奖公告设置浮动奖奖金，通过数据库命令导入至SALE\_DETAIL\_WIN\_T表中。
   12. 更新中奖导入期事件处理记录状态。
   13. 创建中奖导入期事件日志完成操作记录（LOG\_EVENT\_T），状态为完成。
2. 销售临时表导出
   1. 验证销售临时表到出许可，创建期事件（ISSUE\_EVENT\_T），状态为处理中。
   2. 创建期事件日志开始操作记录（LOG\_EVENT\_T），状态为完成。
   3. 销售临时表导出。
   4. 创建期事件日志结束操作记录（LOG\_EVENT\_T），状态为完成。更新期事件处理状态。
3. 中奖明细数据上报
   1. 生成中奖明细上报文件，生成交易系统中奖文件
   2. 创建中奖明细数据上报期事件日志操作记录（LOG\_EVENT\_T），状态为完成。
4. 期状态变更至可兑奖。
   1. 期事件许可验证。
   2. 获取开奖滞后时间（ISSUE\_T.ISSUE\_OPEN\_DELAY\_TIME），系统根据开奖滞后时间进行休眠。
   3. 期事件许可验证。
   4. 期状态设置为已开奖，更新Redis游戏期缓存数据。
   5. 创建期事件处理记录（ISSUE\_EVENT\_T），状态为完成。
   6. 设置游戏期缓存数据。
   7. 删除游戏引擎中奖统计文件
5. 根据《HQXC-TWLS-SD-07-电话销售福利彩票系统与游戏引擎通讯接口协议》发布开奖公告、中奖数据。

## 加载禁销信息

### 时序图

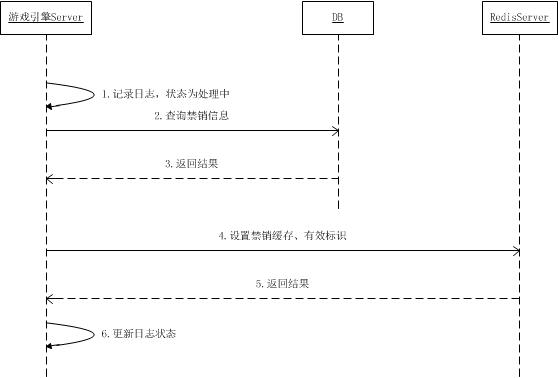


图4-10加载禁销信息时序图

### 实现概述

1. 记录日志（LOG\_T、LOG\_REDIS\_T），状态为处理中。
2. 设置Redis禁销标识缓存为无效。删除禁销缓存。
3. 从DB读取缓存。
4. 设置Redis禁销缓存。
5. 设置Redis禁销标识缓存为有效。
6. 更新日志记录状态。

## 删除无效禁销信息

### 时序图

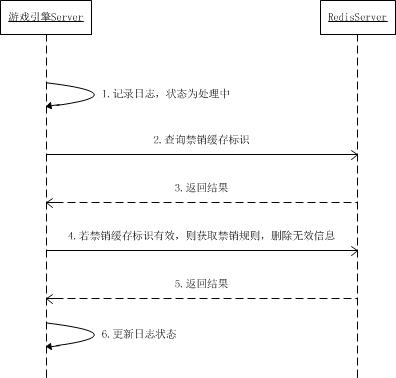


图4-11删除无效禁销信息时序图

### 实现概述

1. 记录日志（LOG\_T、LOG\_REDIS\_T），状态为处理中。
2. 查询禁销标识是否为有效。
3. 从Redis缓存中获取禁销规则OID列表。
4. 根据OID获取SYS\_BAN\_SELL\_T缓存。
5. 如果状态为失效或者结束时间小于当前时间则从缓存中删除，并删除SYS\_BAN\_SELL\_GAME\_T、SYS\_BAN\_SELL\_PLATFORM\_T、SYS\_BAN\_SELL\_ZONE\_T对应的缓存。
6. 更新日志记录状态。

# 交易系统/CRM和管理系统接口实现概述

所有接口受理操作都需要先验证平台的合法性，平台的数据先从Redis读取，如果未读取到则从数据库读取。交易系统的透传、游戏派奖信息查询、即开型游戏刮开通知、即开票刮开通知文件接口暂不实现。CRM和管理系统的即开型游戏刮开通知、即开票刮开通知文件接口暂不实现。

## 数据上报

投注核对文件、弃奖文件、未兑奖文件、兑奖文件上报时使用此时序完成上报。

时序图中的标志文件名的定义如下：

1. 文件名分为四段：平台ID\_\_游戏ID\_\_日期（yyyyMMdd）上报文件名。文件内容为空。
2. 交易系统上报脚本解析标志文件名，获取平台ID、游戏ID、日期、上报文件名，到$上报目录$\$平台ID$\$游戏ID$\日期（yyyyMMdd）\trading文件夹下按照上报文件名找到上报明细文件进行上报。
3. CRM系统上报脚本解析标志文件名，获取平台ID、游戏ID、日期、上报文件名，到$上报目录$\$平台ID$\$游戏ID$\日期（yyyyMMdd）\crm文件夹下按照上报文件名找到上报明细文件进行上报。

### 时序图

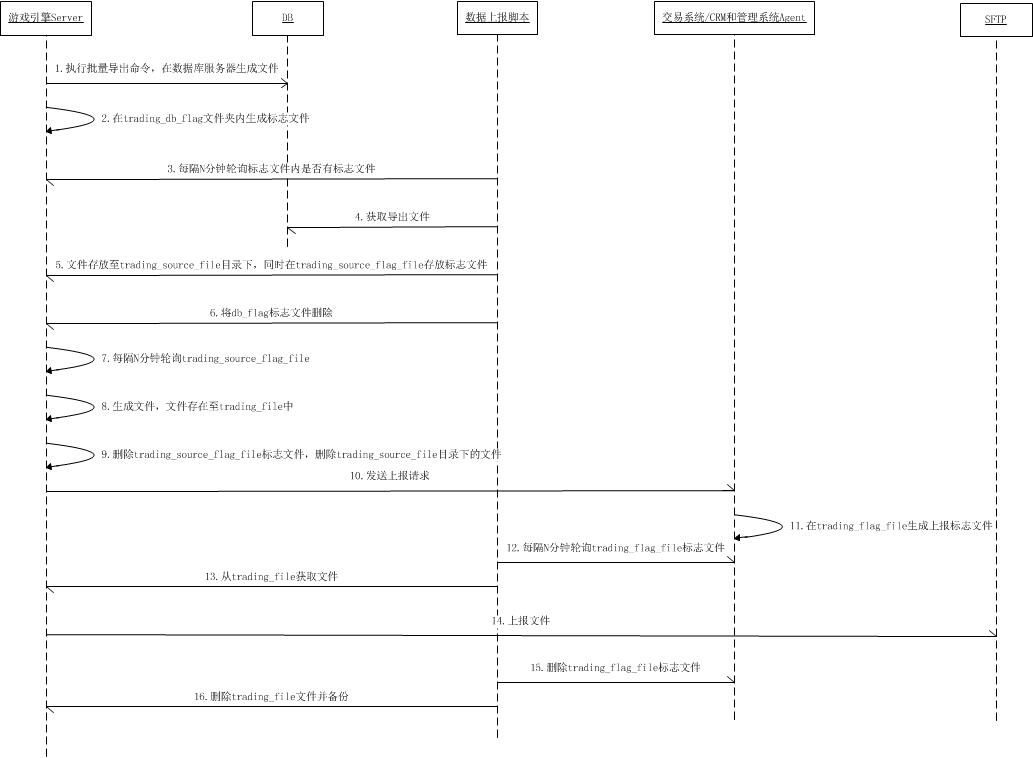


图5-1数据上报时序图

### 实现概述

1. 数据库导出原始文件。
   1. 使用数据库命令从数据库中导出原始文件，并在Server本地的trading\_db\_flag文件夹内生成标志文件，文件名与导出原始文件名一致。
   2. 数据上报脚本每隔N分钟轮询Server本地trading\_db\_flag文件夹内是否有标志文件，如果有则从数据库服务下载文件至Server本地的trading\_source\_file文件夹内，并在trading\_source\_flag\_file文件夹内生成标志文件，文件名与trading\_source\_file文件夹内的文件名一致。
   3. 删除trading\_db\_flag文件夹内的标志文件。
2. 生成上报文件。
   1. Server轮询本地trading\_source\_flag\_file是否有标志文件，如果有则从trading\_source\_file获取导出原始文件，对其进行加工生成上报文件，文件存储在trading\_file文件夹内，删除trading\_source\_flag\_file标志文件，删除trading\_source\_file目录下的文件
   2. 发送文件上报请求至交易系统或CRM和管理系统Agent，Agent收到请求后在trading\_flag\_file文件夹内生成标志文件，文件名与Server的trading\_file文件夹内的文件名一致。
3. 文件上报。
   1. 数据上报脚本每N分钟轮询交易系统或CRM和管理系统Agent的trading\_flag\_file文件夹内是否有标志文件，如果有则从Server的trading\_file文件夹内获取上报文件，上报至指定的SFTP中。
   2. 删除交易系统或CRM和管理系统Agent的trading\_flag\_file文件夹内的标志文件。
   3. 删除Server的trading\_file文件夹内的文件并备份。

## 投注

投注注码采用AES进行加密、解密，每个平台一个密钥。

### 时序图

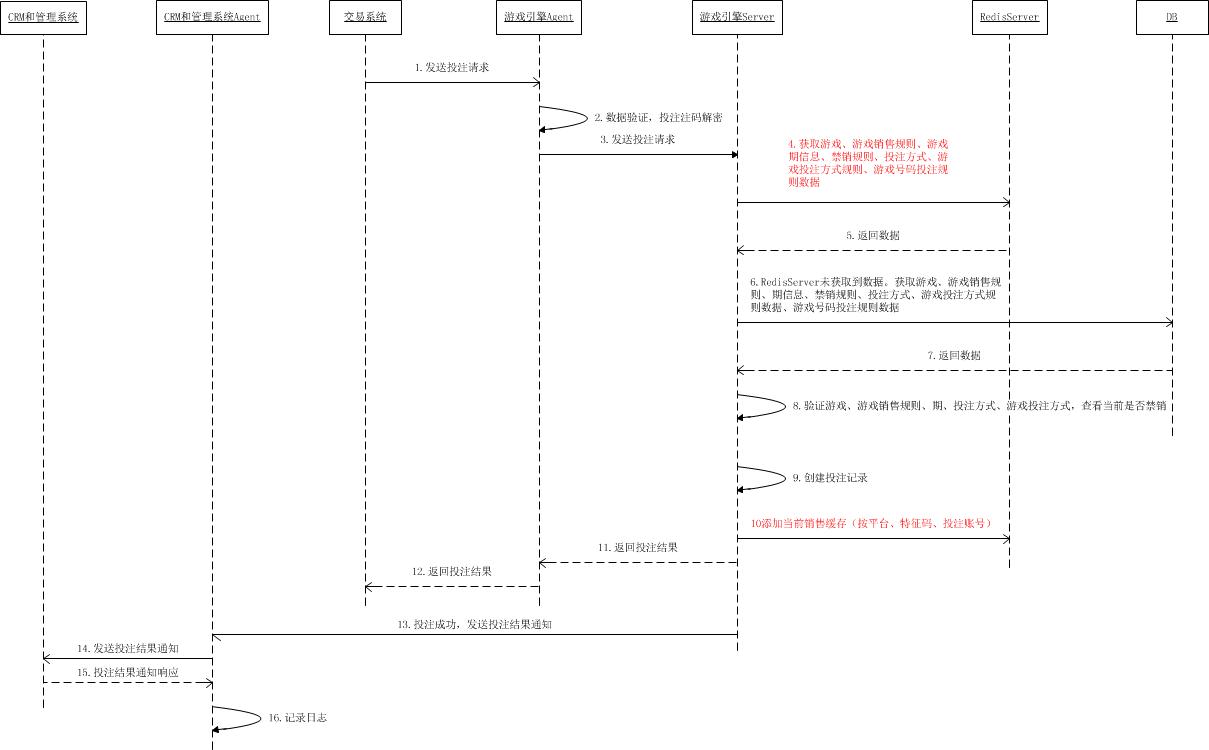


图5-2投注时序图

### 实现概述

1. 验证交易流水号是否在系统中存在（SALE\_DETAIL\_GAME\_ISSUE\_T、SALE\_DETAIL\_T）。
   1. 从当前销售临时表验证流水号是否重复。
   2. 从往期销售表验证流水号是否重复。
   3. 如果上期销售数据未导入至分区表中(ISSUE\_T.EXPORT\_FLAG等于0)，则从上期销售临时表验证流水号是否重复。
2. 从Redis或DB获取平台（PLATFORM\_T）数据，验证平台状态。
3. 从Redis或DB获取游戏、游戏销售规则、游戏期信息、投注方式、游戏投注方式规则数据、游戏号码投注规则。
4. 从Redis获取禁销缓存标识，如果标识为有效则从获取禁销规则，否则从数据库读取禁销缓存信息。
5. 验证游戏状态，只有游戏状态为正常。
6. 验证游戏最后销售时间、最后销售期号。
7. 验证游戏销售规则，销售状态必须为正常，且投注金额未超过单票最大金额。
8. 验证期状态，必须为正常。
9. 验证当前是否禁销。
10. 验证平台合同是否到期。
11. 验证平台是否有权销售游戏
12. 投注方式、游戏投注方式状态必须为正常。
13. 根据投注方式、游戏投注方式规则数据、游戏号码投注规则验证注码、投注金额是否正确。
14. 创建投注受理记录（SALE\_DETAIL\_GAME\_ISSUE\_T），状态为成功或失败。
15. 增加按平台交易流水号、按彩票特征码、按投注账号当前销售信息Redis缓存。

异常处理：

1. 平台状态异常、平台合同到期、平台游戏无权限：错误码1。
2. 游戏未找到：2015。
3. 游戏状态异常：2001。
4. 游戏最后销售时间、最后销售期号验证失败：2001。
5. 游戏销售状态异常：2001。
6. 投注金额超过限制：2002
7. 投注金额错误：2004
8. 投注注码解密失败：2006
9. 期状态异常：2003。
10. 禁销（包含地域验证失败）：2007
11. 投注方式、游戏投注方式状态异常：2005
12. 注码错误：2005
13. 订单号已存在：2016

## 投注结果查询

### 时序图

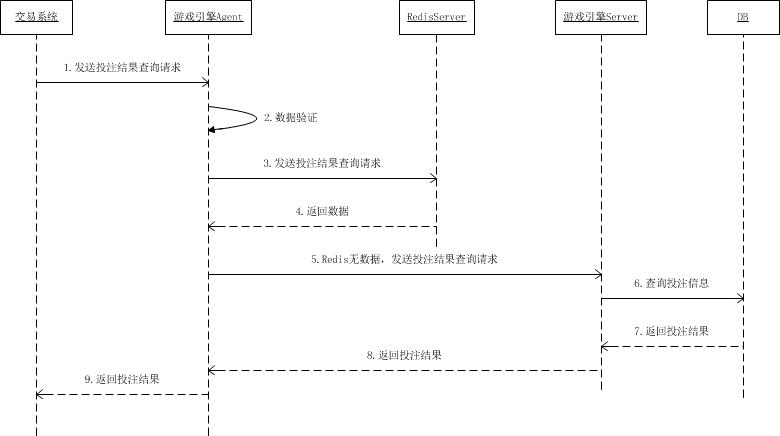


图5-3投注结果查询时序图

### 实现概述

1. Agent从Redis获取投注记录。
2. Redis未获取到投注记录，从DB获取投注记录（SALE\_DETAIL\_GAME\_ISSUE\_T），返回投注结果。如果缓存无数据则查询完毕后将期放入缓存。

异常处理：

1. 未查找到投注记录：返回错误码为2007。

## 兑奖

### 时序图

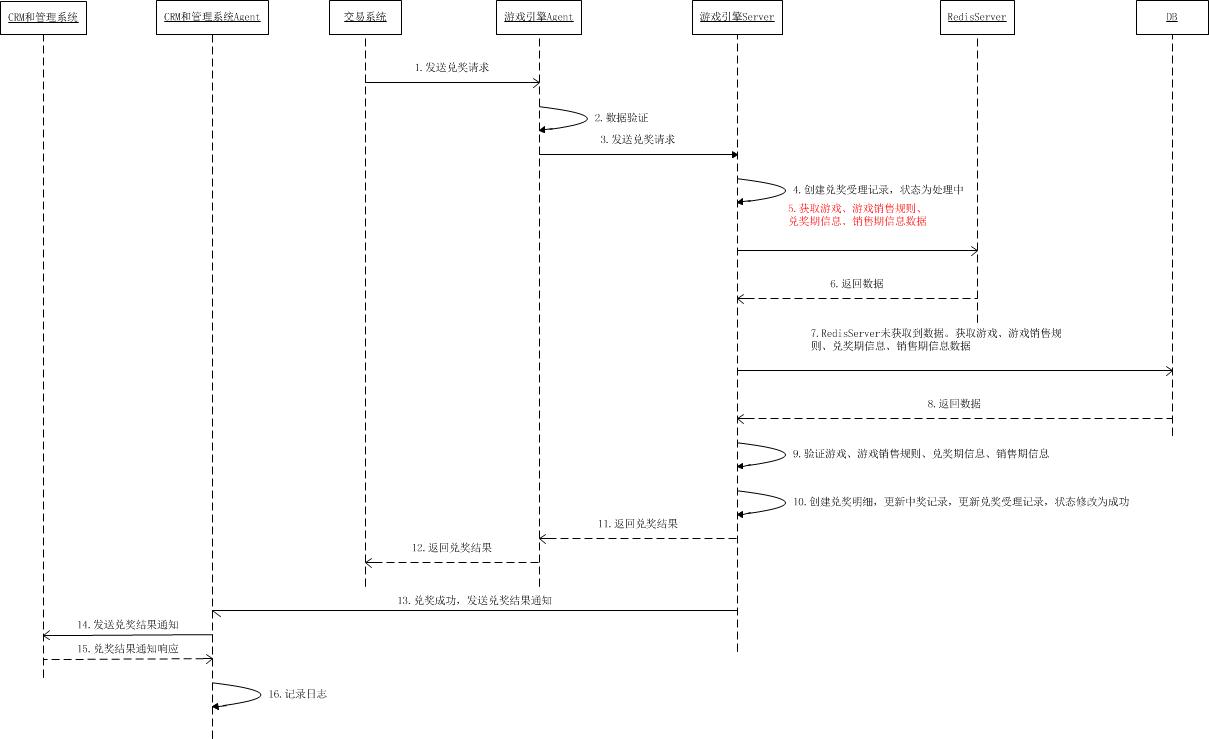


图5-4兑奖时序图

### 实现概述

1. 创建受理记录（SALE\_DETAIL\_ENCASH\_T），状态为处理中。
2. 遍历请求信息中的兑奖请求列表。
3. 从Redis或DB获取游戏、游戏销售规则、兑奖期信息、当前销售期信息数据。
4. 从SALE\_DETAIL\_WIN\_T获取中奖明细。
5. 中奖记录的游戏ID必须与请求信息中的游戏ID一致。
6. 当前必须有销售期数据。
7. SALE\_DETAIL\_WIN\_T.ECNASH\_FLAG必须为0。
8. SALE\_DETAIL\_WIN\_T.ABANDON\_FLAG必须为0，ABANDON\_TIME必须大于当前系统时间。
9. 兑奖期状态必须为已开奖。
10. 验证游戏状态，只有游戏状态为正常。
11. 验证游戏最后兑奖时间、最后兑奖期号。
12. 验证游戏销售规则，兑奖状态必须为正常。
13. 验证中奖彩票是否被篡改。
14. 兑奖操作。SALE\_DETAIL\_WIN\_T.ECNASH\_FLAG设置为1，SALE\_DETAIL\_WIN\_T.ECNASH\_TIME设置为当前系统时间，SALE\_DETAIL\_WIN\_T.ENCASH\_AMOUNT为税前奖金。
15. 重新设置中奖彩票信息的摘要。
16. 创建兑奖明细（SALE\_DETAIL\_ENCASH\_DETAIL\_T），更新兑奖受理记录状态、兑奖记录OID（SALE\_DETAIL\_ENCASH\_T.IWOID）。

## 期信息查询

### 时序图

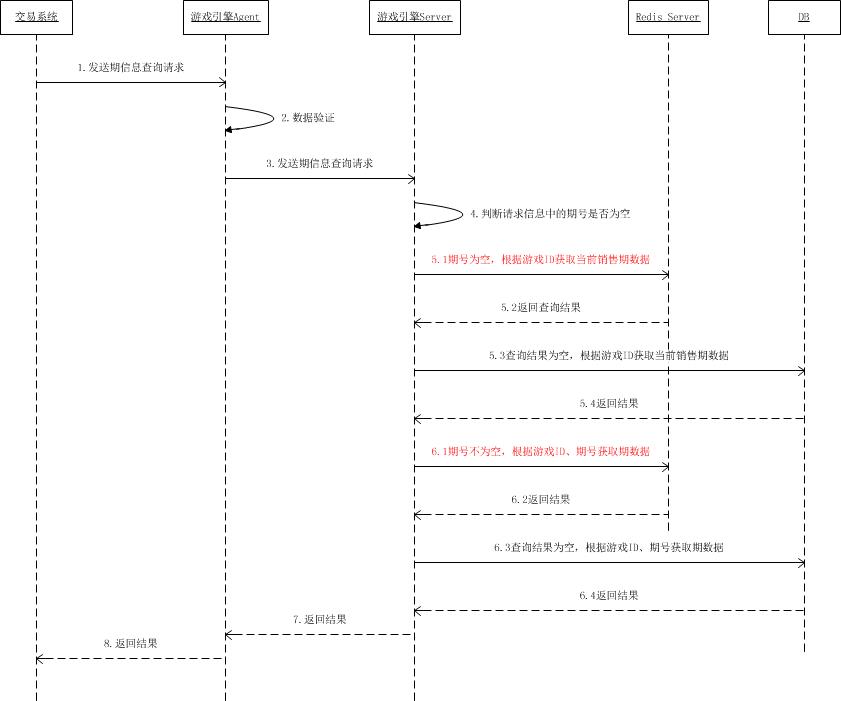


图5-5期信息查询时序图

### 实现概述

1. 从Redis或DB获取平台（PLATFORM\_T）数据，验证平台状态。
2. 根据游戏ID、期号从Redis或DB获取期信息(ISSUE\_T)、游戏销售规则属性（GAME\_SALES\_T）。如果缓存无数据则查询完毕后将期放入缓存。

异常处理：

1. 平台状态异常：返回错误码为1。
2. 未查找到期信息：返回错误码为2012。

## 休市信息查询

### 时序图

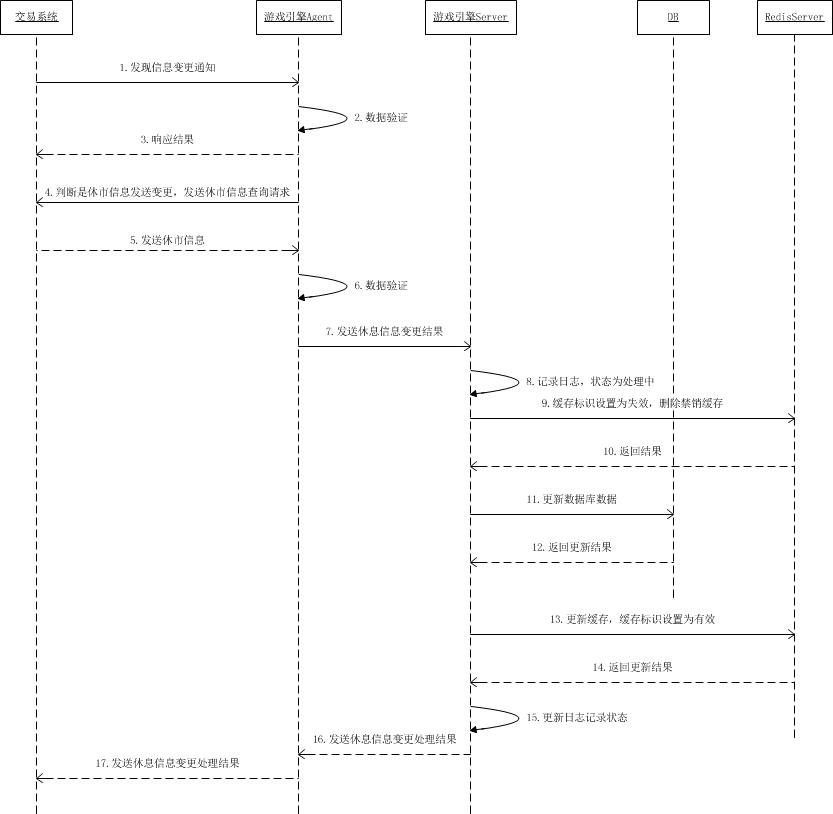


图5-6休市信息查询时序图

### 实现概述

1. 记录日志状态（LOG\_T、LOG\_TRADING\_T）为处理中。
2. 从Redis或DB获取平台（PLATFORM\_T）数据，验证平台状态。
3. 禁销有效标识设置为失效，删除缓存数据。
4. 根据请求信息的年份、名称从SYS\_BAN\_SELL\_T查找BAN\_SELL\_TYPE等于1的数据，如果有重复数据则更新，否则新增。
5. 在数据库中存储休市信息，重新将休市信息加入缓存，禁销缓存标识设置有效。数据库存储逻辑如下：
   1. BEGIN\_TIME：请求信息的开始时间。
   2. END\_TIME：请求信息的结束时间。
   3. BAN\_SELL\_DESC：请求信息的休市名称。
   4. ZONE\_TYPE、GAME\_TYPE、PLATFORM\_TYPE均为0。
   5. STATE：0。
6. 更新日志状态。

异常处理：

1. 平台状态异常：返回错误码为1。

## 节假日信息查询

### 时序图

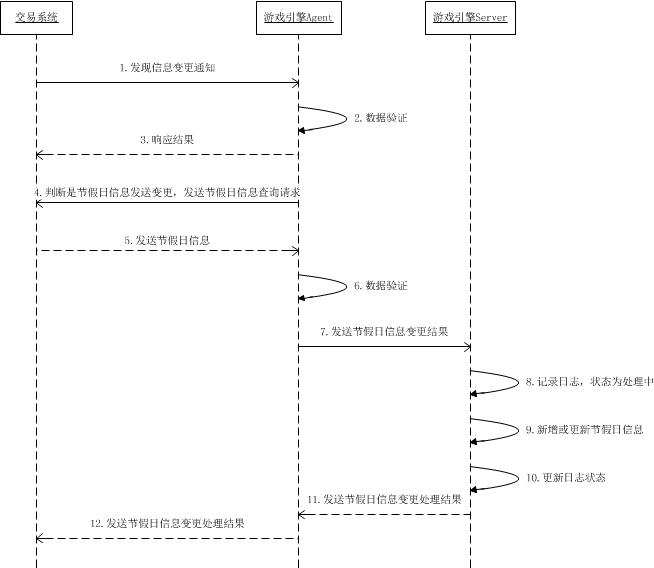


图5-7节假日信息查询时序图

### 实现概述

1. 记录日志状态（LOG\_T、LOG\_TRADING\_T）为处理中。
2. 从Redis或DB获取平台（PLATFORM\_T）数据，验证平台状态。
3. 根据请求信息的年份从HOLIDAYS\_T查找数据，如果有重复数据则更新，否则新增。
4. 新增或更新节假日信息。
5. 更新日志状态。

异常处理：

1. 平台状态异常：返回错误码为1。

## 游戏基本参数查询

### 时序图

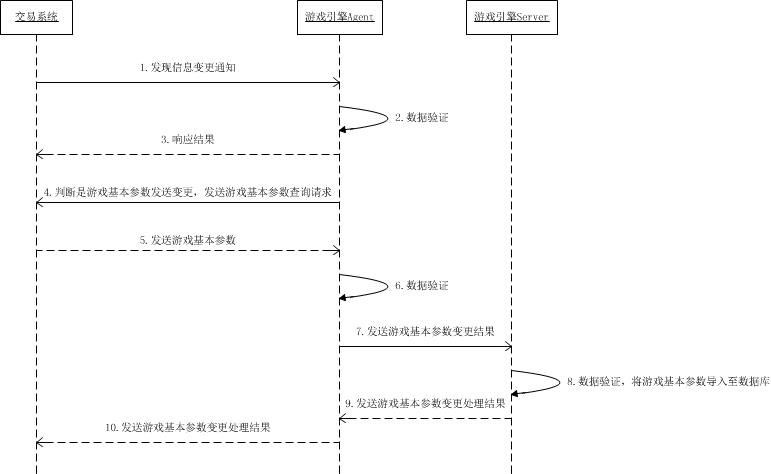


图5-8游戏基本参数查询时序图

### 实现概述

1. 记录日志状态（LOG\_T、LOG\_TRADING\_T）为处理中。
2. 从Redis或DB获取平台（PLATFORM\_T）数据，验证平台状态。
3. 根据游戏ID查询游戏数据（GAME\_T），如果数据存在则返回。
4. 根据游戏ID查询游戏政策参数数据（GAME\_POLICY\_T）
   1. 数据不存在，新增数据，将政策参数添加至RETURN\_RATE、CUR\_RETURN\_RATE、ADJUST\_RATE、SELL\_RATE\_SUM、AUTHORITY\_RATE、SELL\_RATE\_POINT、DEALER\_RATE、GAME\_RATE、WELFARE\_RATE、SELL\_RATE\_CENTER、SELL\_RATE\_PROVINCE、TAX\_BEGIN、TAX\_RATE。
   2. 数据已存在，重置政策参数比例，更新数据，更新数据表的CHANGE\*字段，EXECUTE\_BEGIN\_TIME为请求信息中的生效时间。
5. 更新日志状态。

异常处理：

1. 平台状态异常：返回错误码为1。
2. 游戏编码不存在：返回错误码为2001。

## 投注核对文件

### 时序图

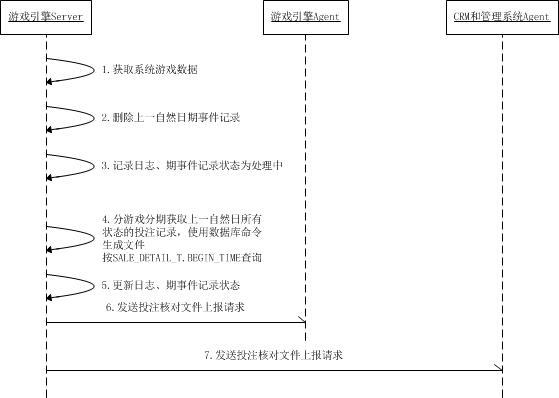


图5-9投注核对文件时序图

### 实现概述

1. 查询当前系统所有游戏数据。
2. 查询当前系统所有平台数据。
3. 删除上一自然日期事件记录（每游戏每期每平台一条日志记录）
4. 记录日志（LOG\_T、LOG\_EVENT\_T）、期事件（ISSUE\_EVENT\_T）状态为处理中
5. 确定导出数据表。
   1. 期结日大于上一自然日且期状态不等于0，从销售临时表导出数据。
   2. 期结日为上一自然日且期状态不等于0，且查询期事件执行日志，已完成销售临时表导出至正式表，从SALE\_DETAIL\_T导出数据，否则终止操作，系统报警，报警日志内容为：【销售临时表未导出至正式表】。
6. 使用数据库导出命令分游戏分期分平台导出数据，付费类型为付费。BEGIN\_TIME为上一自然日，导出数据如下：
   1. GAME\_ID
   2. PLATFORM\_ID
   3. ISSUE\_NUMBER
   4. TOTAL\_MONEY
   5. TRANS\_NUMBER
   6. CHANNEL\_TRANS\_NUMBER
   7. USER\_ID
   8. BEGIN\_TIME
   9. TICKET\_STATE
7. 更新日志、期事件记录状态
8. 按照《HQXC-TWLS-SD-07-电话销售福利彩票系统与游戏引擎通讯接口协议》生成文件，并上传至交易系统、CRM和管理系统指定目录下。
9. 根据投注文件汇总生成投注账号信息（BET\_USER\_T、BET\_USER\_GAME\_T数据）（暂不实现）。

## 弃奖文件

### 时序图

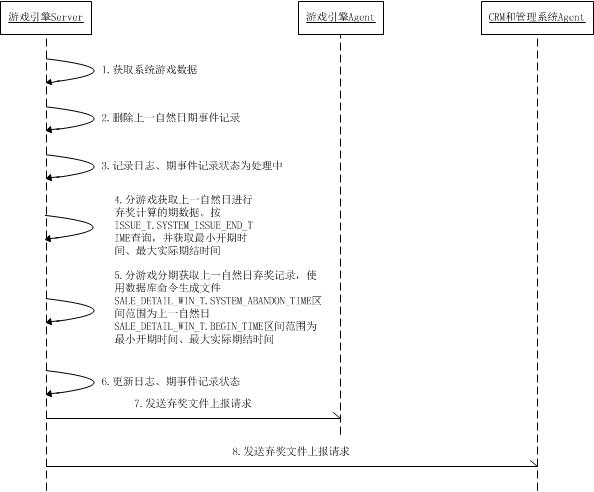


图5-10弃奖文件时序图

### 实现概述

1. 查询当前系统所有游戏数据。
2. 查询当前系统所有平台数据。
3. 删除上一自然日期事件记录（每游戏每天每平台一条日志记录）
4. 记录日志（LOG\_T、LOG\_EVENT\_T）、期事件（ISSUE\_EVENT\_T）状态为处理中
5. 查询上一自然日弃奖的弃奖期，条件为ISSUE\_T.SYSTEM\_ISSUE\_END\_TIME为上一自然日。
6. 使用数据库导出命令从SALE\_DETAIL\_WIN\_T分游戏分平台导出数据，导出条件如下：
   1. SALE\_DETAIL\_WIN\_T.SYSTEM\_ABANDON\_TIME区间范围为上一自然日。
   2. SALE\_DETAIL\_WIN\_T.BEGIN\_TIME区间范围为最小开期时间（当日0点）、最大实际期间时间（当日24点）。
   3. SALE\_DETAIL\_WIN\_T. ABANDON\_FLAG等于1。
7. 导出数据如下：
   1. GAME\_ID
   2. PLATFORM\_ID
   3. ABANDON\_AMOUNT
   4. TRANS\_NUMBER
   5. CHANNEL\_TRANS\_NUMBER
   6. USER\_ID
   7. ISSUE\_NUMBER
   8. TOTAL\_WIN\_AMOUNT
   9. WIN\_COUNT\*
   10. WIN\_AMOUNT\*
8. 更新日志、期事件记录状态
9. 按照《HQXC-TWLS-SD-07-电话销售福利彩票系统与游戏引擎通讯接口协议》生成文件，并上传至交易系统、CRM和管理系统指定目录下。

## 未兑奖文件

### 时序图

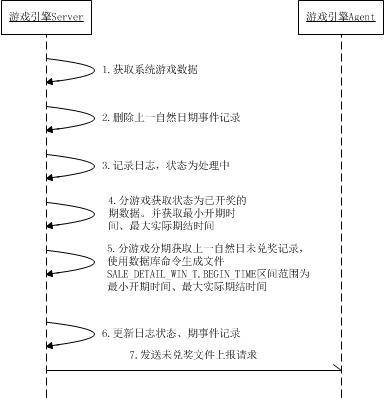


图5-11未兑奖文件时序图

### 实现概述

1. 查询当前系统所有游戏数据。
2. 查询当前系统所有平台数据。
3. 删除上一自然日期事件记录（每游戏每天每平台一条日志记录）
4. 记录日志（LOG\_T、LOG\_EVENT\_T）、期事件（ISSUE\_EVENT\_T）状态为处理中
5. 分游戏获取状态为已开奖的期数据。并获取最小开期时间、最大期结时间。
6. 使用数据库导出命令从SALE\_DETAIL\_WIN\_T分游戏分平台导出数据，导出条件如下：
   1. SALE\_DETAIL\_WIN\_T.BEGIN\_TIME区间范围为最小开期时间（当日0点）、最大实际期间时间（当日24点）
7. 导出数据如下：
   1. GAME\_ID
   2. PLATFORM\_ID
   3. TRANS\_NUMBER
   4. CHANNEL\_TRANS\_NUMBER
   5. USER\_ID
   6. ISSUE\_NUMBER
   7. TOTAL\_WIN\_AMOUNT
   8. AFTER\_TAX\_WIN\_BONUS
   9. WIN\_COUNT\*
   10. WIN\_AMOUNT\*
8. 更新日志、期事件记录状态
9. 按照《HQXC-TWLS-SD-07-电话销售福利彩票系统与游戏引擎通讯接口协议》生成文件，并上传至交易系统指定目录下。

## 兑奖文件

### 时序图

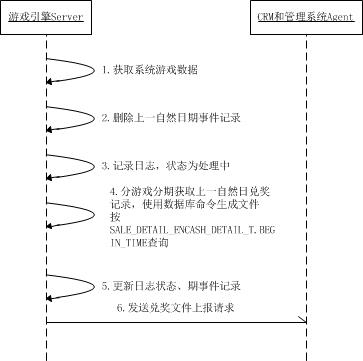


图5-12兑奖文件时序图

### 实现概述

1. 查询当前系统所有游戏数据。
2. 查询当前系统所有平台数据。
3. 删除上一自然日期事件记录（每游戏每天每平台一条日志记录）
4. 记录日志（LOG\_T、LOG\_EVENT\_T）、期事件（ISSUE\_EVENT\_T）状态为处理中
5. 使用数据库导出命令从SALE\_DETAIL\_ENCASH\_DETAIL\_T分游戏分平台导出数据，导出条件如下：
   1. ENCASH\_TIME区间范围为上一自然日
   2. SALE\_DETAIL\_ENCASH\_DETAIL\_T.ECNASH\_STATE等于1。
6. 导出数据如下：
   1. GAME\_ID
   2. PLATFORM\_ID
   3. BET\_TRANS\_NUMBER
   4. BET\_CHANNEL\_TRANS\_NUMBER
   5. USER\_ID
   6. TOTAL\_WIN\_AMOUNT
   7. AFTER\_TAX\_WIN\_BONUS
   8. WIN\_COUNT\*
   9. WIN\_AMOUNT\*
7. 更新日志、期事件记录状态
8. 按照《HQXC-TWLS-SD-07-电话销售福利彩票系统与游戏引擎通讯接口协议》生成文件，并上传至CRM和管理系统指定目录下。

# 管理系统核心功能实现概述

## 通用后台管理设计

### 实现目标

通用后台管理系统的计划实现以下目标：

1. 提供统一的权限管理框架，将权限验证与业务操作分离，做到对功能级权限粗粒度或细粒度的管理。
2. 提供独立的后台管理功能，如用户管理、角色管理、功能项管理、权限管理等；提供统一的界面风格，提供通用的功能组件，如分页组件、菜单树等。

### 权限管理设计

后台管理系统中的权限分为两种：功能级权限和数据级权限。功能级权限包括用户可以使用的功能菜单（粗粒度），以及在菜单中可以操作的按钮（细粒度）；数据级权限包括用户可以操作哪些数据。在本次的通用后台管理系统中只包含对功能级权限的管理，不包含数据级权限的部分。对于数据级权限可以在操作时根据业务需求增加限定条件来完成。

后台管理系统中的权限管理参考了RBAC（Role-Based Access Control，基于角色的访问控制）中的RBAC0模型，同时结合开源的安全框架（如Spring Security3）的特性，将权限验证的重心转移到对功能资源地址（即功能URL）的访问控制上来。整体流程如下图：



1. 用户发送请求访问后台管理系统中某个具体的功能项（URL）
2. 安全框架会拦截这个请求，并根据请求的URL查询出能够访问这个URL的角色信息列表。
3. 框架将当前登录用户的角色列表与查询出的角色列表进行对比。
4. 如果当前用户拥有可以访问此URL的角色，则请求放行，继续调用具体的功能项，反之则拒绝请求。

## 设计约定

### 创建类型约定

在后台管理系统的实体模型（用户、角色、功能项）中均定义了字段build\_type（创建类型），该字段取值为0时，表示实体信息为初始化时生成，属于后台管理系统提供的基础数据；该字段取值为1时，表示实体信息为基于后台管理系统提供的功能新增数据。当实体的创建类型为内置时，实体的信息不可通过功能修改（用户修改登录密码除外）、删除，用户角色与角色权限也不可变更。

### 数据查询约定

查询数据时如果涉及到游戏，则需要通过SYS\_USER\_GAME\_T表获取当前管理员已授权的游戏，查询已授权的游戏。如果查询条件有游戏列表，从通过GAME\_T、SYS\_USER\_GAME\_T获取游戏列表。

## 日志管理

### 系统用户操作日志

根据查询条件从LOG\_T、LOG\_USER\_T表获取数据。

查询条件设置规则：

1. 登录名：SYS\_USER\_T.USER\_ID
2. 真实姓名：SYS\_USER\_T.USER\_NAME
3. 角色：SYS\_AUTHORITY\_T.SYS\_ROLE\_OID
4. 操作开始时间：LOG\_T.PROCESS\_BEGIN
5. 操作结束时间：LOG\_T.PROCESS\_END
6. 日志类型：LOG\_TYPE\_T.TYPE\_LEVEL
7. 功能名称：LOG\_USER\_T.FUNCTION\_OID
8. 操作名称：LOG\_USER\_T.ACTION\_TYPE

查询结果展现规则：

1. 登录名：SYS\_USER\_T.USER\_ID
2. 真实姓名：SYS\_USER\_T.USER\_NAME
3. 角色：SYS\_AUTHORITY\_T.SYS\_ROLE\_OID
4. 日志类型：LOG\_TYPE\_T.TYPE\_LEVEL
5. 功能名称：LOG\_USER\_T.FUNCTION\_OID
6. 操作名称：LOG\_USER\_T.ACTION\_TYPE
7. 操作前数据：LOG\_USER\_T.DATA\_BEFORE
8. 操作后数据：LOG\_USER\_T.DATA\_AFTER
9. 日志描述：LOG\_T.ABSTRACT
10. 操作开始时间：LOG\_T.PROCESS\_BEGIN
11. 操作结束时间：LOG\_T.PROCESS\_END
12. 执行状态：LOG\_T.STATE

### 系统处理日志

从ISSUE\_EVENT\_T表获取数据。

查询条件设置规则如下：

1. 日志类型：ISSUE\_EVENT\_T.SYS\_ISSUE\_EVENT\_T
2. 游戏名称：ISSUE\_EVENT\_T.GAME\_OID
3. 开始时间：ISSUE\_EVENT\_T.START\_DATE
4. 结束时间：ISSUE\_EVENT\_T.END\_DATE

查询结果展现规则如下：

1. 日志类型：ISSUE\_EVENT\_T.SYS\_ISSUE\_EVENT\_T
2. 日志描述：
   1. 日志为按期生成：$期事件名称$,gameId=$游戏编号$,issue=$期号$
   2. 日志为按日：$期事件名称$,$操作数据归属日期（yyyy-MM-dd）$,,gameId=$游戏编号$,issue=$期号$,$平台名称$。操作数据归属日期为ISSUE\_EVENT\_T. PROCESS\_DATE。如果期号为空则不显示。
3. 开始时间：ISSUE\_EVENT\_T.START\_DATE
4. 结束时间：ISSUE\_EVENT\_T.END\_DATE
5. 执行状态：ISSUE\_EVENT\_T.PROCESS\_STATE
6. 操作人：ISSUE\_EVENT\_T.CREATOR

## 销售系统

基于PLATFORM\_T、PLATFORM\_GAME\_T按照需求进行查询、新增、修改。新增和修改需要安装Redis使用约定更新缓存。

新增、修改需要根据日志记录约定记录日志（LOG\_T、LOG\_USER\_T）。

## 游戏参数

基于GAME\_T、GAME\_SALES\_T按照需求进行查询。

## 游戏政策参数

基于GAME\_POLICY\_T按照需求进行查询。

如果EXECUTE\_BEGIN\_TIME不为NULL且小于当前系统时间，则展现数据从CHANGE\*获取，否则从非CHANGE\*字段获取。

## 休市查询

基于SYS\_BAN\_SELL\_T按照需求获取BAN\_SELL\_TYPE等于1的数据。

## 节假日查询

基于HOLIDAYS\_T按照需求进行查询。

## 业务管理-彩票销售

### 期数据处理模式管理

查询条件设置规则：

1. 游戏名称：按照数据查询约定获取游戏列表。
2. 根据期从SYS\_ISSUE\_EVENT\_EXE\_T获取期事件列表，按事件编码生序排序SYS\_ISSUE\_EVENT\_T.SHOW\_FLAG1等于1，按GROUP\_NAME展现名字。
3. 根据获取期事件执行记录。
   1. 如果SYS\_ISSUE\_EVENT\_ EXE\_T. EXE\_PERMIT等于0则不可设置。同时检查EXE\_PERMIT\_GROUP\_ID字段值相同的其他事件。
   2. 如果某事件正在处理中，则不允许设置。
   3. 如果某事件已手动执行成功，则不允许设置为自动模式。同时检查EXE\_PERMIT\_GROUP\_ID字段值相同的其他事件。
   4. 如果某事件执行成功，则需要进行重复执行事件码的检查，如果检查不通过则不允许设置。同时检查EXE\_PERMIT\_GROUP\_ID字段值相同的其他事件。
4. 界面选择为许可模式，获取该许可分组ID，并查询EXE\_PERMIT\_GROUP\_ID等于该分组ID的许可记录，一并修改为许可模式。

### 新期参数管理

查询条件设置规则：游戏名称：按照数据查询约定获取游戏列表。

操作逻辑如下：

1. 根据游戏、期号查询，如果期状态不为0则不允许修改，否则允许修改。数据不存在允许新增。
2. 逻辑如下：
   1. 验证期号是否连续、本期开始销售时间是否在上期期结时间之后。
      1. 期号后三位为001，根据游戏ID查询当前系统是否有期数据，如果无期数据则验证通过。否则获取期号前4位为年，查询去年最后一期期数据，如果数据不存在则表示期号不连续，同时验证本期开始销售时间是否在上期期结时间之后。
      2. 期号后三位不为001，则按期号-1查询期数据。如果期数据不存在，则表示期号不连续，同时验证本期开始销售时间是否在上期期结时间之后。
   2. 验证弃奖期号。
      1. 弃奖数量为0，则弃奖起始期号必须为空。
      2. 弃奖数量为1。
         1. 根据弃奖起始期号获取期数据（简称弃奖期），如果数据不存在则验证失败。
         2. 如果弃奖起的兑奖截止时间在本期开期时间之前或本期开期时间之后，则验证失败。
      3. 弃奖数量大于1。
         1. 弃奖起始期年份大于本期年份，则验证失败。
         2. 弃奖起始期年份等于本期年份。
            1. 弃奖起始期号+弃奖期数量大于等于本期期号，则验证失败。
            2. 获取弃奖截止期号期信息，期号=弃奖起始期号+弃奖期数量-1，如果期信息在本期开期时间之前或本期开期时间之后，则验证失败。
         3. 弃奖起始期年份小于本期年份。查询期信息，期号大于等于弃奖起始期，小于本期，按期号升序，简称期信息集合。
            1. 如果期信息集合数量小于弃奖数量，则验证失败。
            2. 如果期信息集合第一条记录的期号不等于弃奖起始期号，则验证失败。
            3. 如果期信息集合第一条期信息的兑奖时间在本期开期时间之前或本期开期时间之后，则验证失败。
            4. 如果期信息集合第N条期信息的兑奖时间在本期开期时间之前或本期开期时间之后，则验证失败。N等于弃奖数量。
            5. 参照【中彩开奖管理系统接口实现概述】【新期导入逻辑】。

### 期结处理

查询ISSUE\_EVENT\_T获取期执行事件明细，根据需求完成验证。

参考【中彩开奖管理系统接口实现概述】完成操作。

参考日志约定记录管理员操作日志（LOG\_T、LOG\_USER\_T）。

监控日志从LOG\_EVENT\_T获取。

### 上报期结数据

查询ISSUE\_EVENT\_T获取期执行事件明细，根据需求完成验证。

操作前需检查期结原始文件是否已下载。

参考【中彩开奖管理系统接口实现概述】完成操作。

参考日志约定记录管理员操作日志（LOG\_T、LOG\_USER\_T）。

### 开奖号码管理

查询ISSUE\_EVENT\_T获取期执行事件明细，根据需求完成验证。

参考【中彩开奖管理系统接口实现概述】完成操作。

参考日志约定记录管理员操作日志（LOG\_T、LOG\_USER\_T）。

### 中奖计算处理

查询ISSUE\_EVENT\_T获取期执行事件明细，根据需求完成验证。

参考【中彩开奖管理系统接口实现概述】完成操作。

参考日志约定记录管理员操作日志（LOG\_T、LOG\_USER\_T）。

监控日志从LOG\_EVENT\_T获取。

### 上报中奖数据

查询ISSUE\_EVENT\_T获取期执行事件明细，根据需求完成验证。

操作前需检查兑奖明细原始文件是否已下载。

参考【中彩开奖管理系统接口实现概述】完成操作。

参考日志约定记录管理员操作日志（LOG\_T、LOG\_USER\_T）。

### 开奖公告管理

查询ISSUE\_EVENT\_T获取期执行事件明细，根据需求完成验证。

参考【中彩开奖管理系统接口实现概述】完成操作。

参考日志约定记录管理员操作日志（LOG\_T、LOG\_USER\_T）。

监控日志从LOG\_EVENT\_T获取。

### 开奖公告通知处理

查询ISSUE\_EVENT\_T获取期执行事件明细，根据需求完成验证。

参考【中彩开奖管理系统接口实现概述】完成操作。

参考日志约定记录管理员操作日志（LOG\_T、LOG\_USER\_T）。

### 数据文件处理

查询ISSUE\_EVENT\_T获取期执行事件明细，根据需求完成验证。

参考【中彩开奖管理系统接口实现概述】完成操作。

参考日志约定记录管理员操作日志（LOG\_T、LOG\_USER\_T）。

### 游戏期信息查询

基于ISSUE\_T、GAME\_T获取期数据。

查询条件设置规则如下：

1. 游戏名称：按照数据查询约定获取游戏列表。
2. 开始期号：从ISSUE\_T表按照期号倒序获取第三十条数据。如果不足三十条数据则取最后条数据。
3. 结束期号：从ISSUE\_T表按照期号倒序获取第一条数据。

查询结果展现规则如下：

1. 促销状态：ISSUE\_T. GAME\_PROMOTION\_OID. PROMOTION\_NAME
2. 其余字段从ISSUE\_T、GAME\_T表获取。

### 游戏开奖公告查询

基于ISSUE\_OPEN\_T、ISSUE\_OPEN\_GROUP\_T、ISSUE\_OPEN\_GRADE\_T获取数据。

查询条件设置规则如下：

1. 游戏名称：按照数据查询约定获取游戏列表。
2. 开始期号：从ISSUE\_EVENT\_T表按照期号倒序获取事件码等于130第三十条数据。如果不足三十条数据则取最后条数据。
3. 结束期号：从ISSUE\_EVENT\_T表按照期号倒序获取事件码等于130第一条数据。

列表查询结果展现规则如下：

1. 从ISSUE\_OPEN\_T获取

详情结果展现规则如下：

1. 根据ISSUE\_OPEN\_T.IWOID获取开奖号码组记录（ISSUE\_OPEN\_GROUP\_T）
2. 根据开奖号码组获取开奖公告记录（ISSUE\_OPEN\_GRADE\_T）
3. 开奖轮数：ISSUE\_OPEN\_GROUP\_T. NUMBER\_GROUP
4. 奖等编号：ISSUE\_OPEN\_GRADE\_T.GRADE
5. 中奖注数：ISSUE\_OPEN\_GRADE\_T.WIN\_COUNT
6. 奖等奖金：ISSUE\_OPEN\_GRADE\_T.WIN\_AMOUNT

## 业务监控-彩票销售

### 彩票销售期状态

基于ISSUE\_EVENT\_T按照需求查询数据。

查询条件设置规则：

1. 游戏名称：按照数据查询约定获取游戏列表。
2. 开始期号：从ISSUE\_T表按照期号倒序获取第三十条数据。如果不足三十条数据则取最后一条数据。
3. 结束期号：从ISSUE\_T表按照期号倒序获取第一条数据。

查询结果展现规则：

1. 按照游戏、期号从ISSUE\_EVENT\_T获取SYS\_ISSUE\_EVENT\_T.SHOW\_FLAG2等于1的数据。
2. 根据ISSUE\_EVENT\_T（PROCESS\_STATE等于1）数据设置高亮显示。
3. 选择某一期，按照期OID从ISSUE\_EVENT\_T获取数据，按照START\_DATE升序展现记录。

### 当前销售票

基于Redis、销售临时表（SALE\_DETAIL\_GAME\_ISSUE\_T）按照需求获取数据。

查询条件设置规则：

1. 游戏名称：按照数据查询约定获取游戏列表。
2. 期号：按游戏从ISSUE\_T获取ISSUE\_STATE等于1的数据。

查询结果规则：

1. 按特征码查询：
   1. 从Redis按特征码查询当前销售记录。
   2. 缓存无数据不存在从销售临时表（SALE\_DETAIL\_GAME\_ISSUE\_T）查询数据。
   3. 投注结果失败原因根据错误码展现。
2. 按投注账号查询
   1. 从Redis按投注账号查询当期销售记录OID有序集合，按照分页条件倒序获取销售记录OID列表。
   2. 根据销售列表从Redis获取按投注账号查询当期销售记录明细。
   3. 缓存无数据不存在从销售临时表（SALE\_DETAIL\_GAME\_ISSUE\_T）查询数据。
   4. 投注结果失败原因根据错误码展现。

### 历史销售票

基于SALE\_DETAIL\_T按照需求获取数据。

主要查询条件设置规则：

1. 游戏名称：按照数据查询约定获取游戏列表。

查询结果规则：

1. 根据游戏、期号获取期数据，从SALE\_DETAIL\_T获取数据，BEGIN\_TIME时间范围为，开期时间（当日0点）、期结时间或实际期结时间（当日24点）（如果实际期结时间不为空则按实际期结时间查询）。
2. 查询明细规则：
   1. 先从SALE\_DETAIL\_WIN\_T表获取数据，如果数据不存在则从SALE\_DETAIL\_T表获取数据（中奖信息均不填写）。
   2. 中奖标志：
      1. 从ISSUE\_EVENT\_T表获取事件码等于130的记录,如果没有该期的记录则为未开奖。
      2. 从ISSUE\_EVENT\_T表获取事件码等于130的记录,如果有该期的记录且中奖表无数据，则为未中奖，否则为已中奖

### 中奖彩票

基于SALE\_DETAIL\_WIN\_T按照需求获取数据。

主要查询条件设置规则：

1. 游戏名称：按照数据查询约定获取游戏列表。

查询结果规则：

1. 根据游戏、期号获取期数据，从SALE\_DETAIL\_WIN\_T获取数据，BEGIN\_TIME时间范围为，开期时间（当日0点）、期结时间或实际期结时间（当日24点）（如果实际期结时间不为空则按实际期结时间查询）。

### 兑奖彩票

基于SALE\_DETAIL\_WIN\_T按照需求获取数据。

主要查询条件设置规则：

1. 游戏名称：按照数据查询约定获取游戏列表。

查询结果规则：

1. 根据游戏、期号获取期数据，从SALE\_DETAIL\_WIN\_T获取ENCASH\_FLAG等于1的数据，BEGIN\_TIME时间范围为，开期时间（当日0点）、期结时间或实际期结时间（当日24点）（如果实际期结时间不为空则按实际期结时间查询）。

### 弃奖彩票

基于SALE\_DETAIL\_WIN\_T按照需求获取数据。

主要查询条件设置规则：

1. 游戏名称：按照数据查询约定获取游戏列表。

查询结果规则：

1. 根据游戏、期号获取期数据，从SALE\_DETAIL\_WIN\_T获取ABANDON\_FLAG等于1的数据，BEGIN\_TIME时间范围为，开期时间（当日0点）、期结时间或实际期结时间（当日24点）（如果实际期结时间不为空则按实际期结时间查询）。

### 彩票销售期汇总数据

基于RPT\_SELL\_STATISTICS\_T按照需求查询数据。

查询条件设置规则：

1. 游戏名称：按照数据查询约定获取游戏列表。
2. 开始期号：从ISSUE\_EVENT\_T表按照期号倒序获取事件码等于50第三十条数据。如果不足三十条数据则取最后条数据。
3. 结束期号：从ISSUE\_EVENT\_T表按照期号倒序获取事件码等于50第一条数据。

## 投注账户

### 投注账户信息

基于BET\_USER\_T、BET\_USER\_GAME\_T按照需求查询数据。

查询条件设置规则：

1. 游戏名称：按照数据查询约定获取游戏列表。

查询结果展现规则：

1. 根据投注账号从BET\_USER\_T获取用户数据。
2. 如果游戏名称选择全部则从BET\_USER\_T获取展现数据，否则从BET\_USER\_GAME\_T获取展现数据。

## 资金统计

### 销售资金统计

基于RPT\_SELL\_FUND\_DAY\_T、RPT\_SELL\_FUND\_MONTH\_T按照需求查询数据。

### 兑奖资金统计

基于RPT\_ENCASH\_FUND\_STAT\_T按照需求查询数据。

### 弃奖资金统计

基于RPT\_ABANDON\_FUND\_STAT\_T按照需求查询数据。

### 中奖情况统计

基于RPT\_WIN\_FUND\_STAT\_T按照需求查询数据。

## 打印制式报表

### 当期中奖结果报告单

基于RPT\_ISSUE\_PRIZE\_MERGE\_T按照需求数据。

查询条件设置规则：

1. 游戏名称：按照数据查询约定获取游戏列表。
2. 销售期号：从ISSUE\_EVENT\_T表按照期号倒序获取事件码等于90第一条数据。

查询结果展现条件设置规则：

1. 按游戏、按期号从RPT\_ISSUE\_PRIZE\_MERGE\_T查询MERGE\_TYPE等于3的数据。
2. 奖等：RPT\_ISSUE\_PRIZE\_MERGE\_T.GRADE
3. 中奖注数：RPT\_ISSUE\_PRIZE\_MERGE\_T.WIN\_AMOUNT
4. 幸运二等奖、特别奖为0。

### 当前弃奖/销售报告单

基于RPT\_ISSUE\_PRIZE\_MERGE\_T按照需求数据。

查询条件设置规则：

1. 游戏名称：按照数据查询约定获取游戏列表。
2. 销售期号：从ISSUE\_EVENT\_T表按照期号倒序获取事件码等于50第一条数据。

查询结果展现条件设置规则：

1. 按游戏、按期号从RPT\_ISSUE\_PRIZE\_MERGE\_T查询MERGE\_TYPE等于3的数据。
2. 销售额：RPT\_ISSUE\_PRIZE\_MERGE\_T.SELL\_MONEY
3. 弃奖额：SUM(RPT\_ISSUE\_PRIZE\_MERGE\_T.ABANDON\_MONEY)
4. 弃奖奖等：RPT\_ISSUE\_PRIZE\_MERGE\_T.GRADE
5. 弃奖奖等金额：RPT\_ISSUE\_PRIZE\_MERGE\_T.ABANDON\_MONEY
6. 幸运二等奖、特别奖为0。
7. 弃奖期区间：
   1. 按照游戏期号从ISSUE\_T获取数据。如果弃奖起始期为空则弃奖区间无。
   2. 弃奖起始期号为ISSUE\_T.ABANDON\_BEGIN\_ISSUE\_NUMBER
   3. 从ISSUE\_T按游戏查询期号大于弃奖期号的记录，按期号升序排序，记录条数为ISSUE\_T.ABANDON\_ISSUE\_AMOUNT，最后一条记录的期号为弃奖结束期号。

# 报表计算

报表统计的业务特点主要有如下方面：

1. 数据量巨大

各种销售、中奖等数据量巨大。

1. 逻辑较复杂

各报表数据统计口径都不相同，比如按照期结日、销售日、弃奖日等统计时间。

因此可以先将销售、中奖等数据按照各种统计口径每日生成一批报表中间表数据，然后再从此中间表中汇总统计出各个报表的展示数据。



## 报表计算通用核查功能

### 通用核查原则

每张报表计算时都应该先检查一下该报表所依赖的数据源表是否计算完成，核查通过则正常进行报表统计，并在统计结束后向“报表核查表”中插入该报表当前计算日期已计算完成的记录。

### 通用核查流程图



### 实现概述

1. Spring注入属性增加commonCheckFlag公共核查标识字段（com.iwt.vasoss.gameengine1.report.util.ParamBean）
2. clearFirst为真的时候，除了清除报表数据之外还应清除“报表核查表”相对应数据。
3. 根据commonCheckFlag判断是否执行公共核查。
4. 所有报表计算完毕后均向“报表核查表”中插入当前日期该报表计算结果。

## 报表中间表统计

### 销售数据中间表统计

* 检查当天期结的期是否在计算时已经期结：

根据期结时间ISSUE\_END\_TIME在当天00:00:00~23:59:59之间，查询ISSUE\_T，检查ISSUE\_T. ISSUE\_STATE状态是否<2（2：期结），是则打印警告日志结束报表计算，否则继续；

* 每天从SALE\_DETAIL\_T表获取销售数据，具体条件如下：

1. TICKET\_STATE等于1
2. WAGER\_TYPE等于1
3. BEGIN\_TIME（受理开始时间，如果投注成功该时间为出票时间）在当天00:00:00~23:59:59.999之间

* 按照游戏、期、省、销售系统分组将数据记录至RPT\_MIDDLE\_SELL\_T表中。
* 向“报表核查表”（RPT\_COMMON\_CHECK\_T）插入当前日期该报表计算结果。

### 中奖数据中间表统计

* 检查当天期结的期是否在计算时已经中奖计算

根据期结时间ISSUE\_END\_TIME在当天00:00:00~23:59:59之间，查询ISSUE\_T，检查ISSUE\_T. ISSUE\_STATE状态是否<3（3:已开奖），是则打印警告日志结束报表计算，否则继续；

* 每天从SALE\_DETAIL\_WIN\_T表获取中奖数据，具体条件如下：

1. TICKET\_STATE等于1
2. 实际期结时间ISSUE\_T. ACTUAL\_ISSUE\_END\_TIME在当天00:00:00~23:59:59之间
3. 分区条件：BEGIN\_TIME在ISSUE\_T. ISSUE\_BEGIN\_TIME的00:00:00~ ISSUE\_T. ACTUAL\_ISSUE\_END\_TIME的23:59:59之间

* 按照游戏、期、省、销售系统、大小奖标记分组将将数据记录至RPT\_MIDDLE\_WIN\_T表中；
* 向“报表核查表”（RPT\_COMMON\_CHECK\_T）插入当前日期该报表计算结果。

### 兑奖数据中间表统计

* 每天从SALE\_DETAIL\_ENCASH\_DETAIL\_T表获取兑奖数据，具体条件如下：

1. ENCASH\_TIME为时间条件，在当天00:00:00~23:59:59之间
2. ECNASH\_STATE等于0

* 按照游戏、期、省、销售系统、大小奖标记、兑奖当前期分组将将数据记录至RPT\_MIDDLE\_ENCASH\_T表中；
* 向“报表核查表”（RPT\_COMMON\_CHECK\_T）插入当前日期该报表计算结果。

### 弃奖数据中间表统计

* 检查当天期结的期是否在计算时已经期结：

根据期结时间ISSUE\_END\_TIME在当天00:00:00~23:59:59之间，查询ISSUE\_T，检查ISSUE\_T. ISSUE\_STATE状态是否<2（2：期结），是则打印警告日志结束报表计算，否则继续；

* 每天从SALE\_DETAIL\_WIN\_T表获取弃奖数据，具体条件如下：

1. TICKET\_STATE等于1
2. ABANDON\_FLAG等于1
3. SYSTEM\_ABANDON\_TIME在当天00:00:00~23:59:59之间
4. 分区条件：SALE\_DETAIL\_WIN\_T.ISSUE\_OID关联期表, SALE\_DETAIL\_WIN\_T.BEGIN\_TIME在ISSUE\_T. ISSUE\_BEGIN\_TIME的00:00:00~ ISSUE\_T. ACTUAL\_ISSUE\_END\_TIME的23:59:59之间

* 按照游戏、销售期、弃奖期、省、销售系统、大小奖标记、弃奖当前期分组将将数据记录至RPT\_MIDDLE\_ABANDON\_T表中
* 向“报表核查表”（RPT\_COMMON\_CHECK\_T）插入当前日期该报表计算结果。

## 资金统计

### 销售资金统计

日报：

* 每天从RPT\_MIDDLE\_SELL\_T表计算销售数据，具体条件如下：

1. START\_TIME为查询条件，日报为开始时间为当天的00:00:00，结束时间为当天的23:59:59；
2. 分省、销售系统、游戏、期汇总出销售总注数、总金额，数据记录至RPT\_SELL\_FUND\_DAY\_T；

* 无效数据统计：

GAME\_T、ISSUE\_T、SYS\_REGION\_T、PLATFORM\_T关联，根据ISSUE\_T. ISSUE\_END\_TIME在00:00:00~23:59:59范围，且省份、游戏、期、平台不存在RPT\_MIDDLE\_SELL\_T表记录为条件，统计为无效数据（DATA\_TYPE=0）并记录至RPT\_SELL\_FUND\_DAY\_T;

* 向“报表核查表”（RPT\_COMMON\_CHECK\_T）插入当前日期该报表计算结果。

月报：

* 每月从RPT\_SELL\_FUND\_DAY\_T表计算销售数据，具体条件如下：

1. 检查当月日报是否计算完毕，未计算完毕则打印警告日志结束报表计算，完毕则继续；
2. 月报为月初第一天00:00:00和月末最后一天的23:59:59；
3. 分省、销售系统、游戏、DATA\_TYPE汇总出销售总注数、总金额，数据记录至RPT\_SELL\_FUND\_MONTH\_T;

* 向“报表核查表”（RPT\_COMMON\_CHECK\_T）插入当前日期该报表计算结果。

### 中奖情况统计

* 每天从RPT\_MIDDLE\_WIN\_T表计算中奖数据，具体条件如下：

1. START\_TIME为查询条件，日报为开始时间为当天的00:00:00，结束时间为当天的23:59:59；
2. 分省、销售系统、游戏、期汇总出中奖注数、中奖金额和各奖等信息；
3. 单注奖金用销售期关联ISSUE\_OPEN\_T、ISSUE\_OPEN\_GROUP\_T和ISSUE\_OPEN\_GRADE\_T获取；

* 将数据记录至RPT\_WIN\_FUND\_STAT\_T;
* 无效数据统计：

GAME\_T、ISSUE\_T、SYS\_REGION\_T、PLATFORM\_T关联，根据ISSUE\_T. ISSUE\_END\_TIME在00:00:00~23:59:59范围，且省份、游戏、期、平台不存在RPT\_MIDDLE\_WIN\_T表记录为条件，统计为无效数据（DATA\_TYPE=0）并记录至RPT\_WIN\_FUND\_STAT\_T;

* 向“报表核查表”（RPT\_COMMON\_CHECK\_T）插入当前日期该报表计算结果。

### 兑奖资金统计

* 每天从RPT\_MIDDLE\_ENCASH\_T计算出当日的已兑奖数据，具体条件如下：

1. SELL\_ENCASH\_END\_TIME为查询条件，日报为开始时间为当天的00:00:00，结束时间为当天的23:59:59；
2. 分省、销售系统、游戏、期汇总出兑奖注数、兑奖金额；
3. 单注奖金用销售期关联ISSUE\_OPEN\_T、ISSUE\_OPEN\_GROUP\_T和ISSUE\_OPEN\_GRADE\_T获取；

* 将数据记录至RPT\_ENCASH\_FUND\_STAT\_T;
* 无效数据统计：

GAME\_T、ISSUE\_T、SYS\_REGION\_T、PLATFORM\_T关联，根据ISSUE\_T. ENCASH\_END\_TIME在00:00:00~23:59:59范围，且省份、游戏、期、平台不存在RPT\_MIDDLE\_ENCASH\_T表记录为条件，统计为无效数据（DATA\_TYPE=0）并记录至RPT\_ENCASH\_FUND\_STAT\_T;

* 向“报表核查表”（RPT\_COMMON\_CHECK\_T）插入当前日期该报表计算结果。

### 弃奖资金统计

* 每天从RPT\_MIDDLE\_ABANDON\_T计算出弃奖时间为当日的已弃奖数据，具体条件如下：

1. SELL\_ENCASH\_END\_TIME为查询条件，日报为开始时间为当天的00:00:00，结束时间为当天的23:59:59；
2. 分省、销售系统、游戏、期汇总出弃奖注数、弃奖金额；
3. 单注奖金用销售期关联ISSUE\_OPEN\_T、ISSUE\_OPEN\_GROUP\_T和ISSUE\_OPEN\_GRADE\_T获取；

* 将数据记录至RPT\_ABANDON\_FUND\_STAT\_T;
* 无效数据统计：

GAME\_T、ISSUE\_T、SYS\_REGION\_T、PLATFORM\_T关联，根据ISSUE\_T.SYSTEM\_ISSUE\_END\_TIME在00:00:00~23:59:59范围，且省份、游戏、期、平台不存在RPT\_MIDDLE\_ABANDON\_T表记录为条件，统计为无效数据（DATA\_TYPE=0）并记录至RPT\_ABANDON\_FUND\_STAT\_T;

* 向“报表核查表”（RPT\_COMMON\_CHECK\_T）插入当前日期该报表计算结果。