Siege前后端接口

传输方式：POST和Websocket； 数据格式：JSON；

1. **注册**

* 功能：玩家注册
* URL：<http://localhost:8088/register>
* 输入：（key: value，key统一为String类型）
* **“register”: bool** true时进行注册，false不进行注册，默认true
* 返回：
* **"status" String (success 注册成功/ failed 注册失败)**
* **“account” String** 注册成功返回以下数据结构，失败返回空字符串

**“{“address”: “xxx”, “publicKey”: “xxx”, “privateKey”: “xxx”, “privateKeyEncrypted”: false}”** 注册成功

1. **登陆**

* 功能：玩家登陆
* URL：<http://localhost:8088/login>
* 输入：
* **“address”: String** 玩家地址
* **“publicKey”: String** 玩家公钥
* **“privateKey”: String** 玩家私钥
* **“privateKeyEncrypted”: bool** 私钥是否加密，默认为false
* 返回：
* **"status" String (success 登陆成功/ failed 登陆失败)**

1. **进入匹配，开始游戏**

* 功能：将玩家加入匹配池，达到指定人数后开始游戏，初始化数据表
* 使用websocket进行连接：ws://localhost:8088/WebSocket/playersMatch
* 输入：
* **“match”: bool** 是否进入匹配
* **“signature”: String** 玩家签名 数据结构如下：

“{“address”: “xxx”, “publicKey”: “xxx”, “privateKey”: “xxx”, “privateKeyEncrypted”: false}”

* 返回：
* **"match" : string** 匹配结果，类型如下
  + **"match success"** 匹配成功，所有玩家进入下一阶段
  + **"** **already in matching queue"** 该玩家已经在匹配队列中（为了防止用户掉线后重新匹配）
  + **“enter matching queue”** 成功进入匹配队列
  + **"match waiting"** 玩家匹配等待，暂时无法加入匹配队列
  + **"match error"** 玩家匹配出错
  + **"transfer failed"** 缴纳入场费失败
* **“gameId”: int** 匹配成功的游戏id

以下是连接时返回值，为String数据类型。

* **"connect error"** websocket连接失败
* **"connect success"** websocket连接成功
* **"disConnect"** websocket断开连接

1. **竞标阶段**

* 功能：玩家参与竞标，获取城池的归属权
* 使用websocket进行连接：ws://localhost:8088/WebSocket/ bidding/{gameId}，gameId即为匹配成功时获取的值
* 输入：
* **“first”: bool** 是否为第一次发送数据，用于用户信息注册
* **“address”: string** 用户地址
* **“bidding”: bool** 是否处于竞标阶段还是处于缴纳尾款阶段
* **“price”: double** 竞标出价
* **“signature”: string** 用户缴纳尾款时签名
* 竞标各阶段上述参数默认值如下表所示

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **first** | **address** | **bidding** | **price** | **signature** |
| 用户注册 | **true** | **必填** | **可选** | **可选** | **不填** |
| 竞标 | **false** | **必填** | **true** | **必填** | **不填** |
| 缴纳尾款 | **false** | **必填** | **false** | **必填** | **必填** |

* 返回：(JSONObject形式)
* **“stage”: string** 当前阶段，数据如下表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **五种阶段** | **ready** | **bidding** | **publicity** | **pay** |
| **解释** | **用户注册完毕,准备开始接收计时数据** | **30秒倒计时阶段** | **每一轮竞标公示** | **缴纳尾款** |

* **“timer”: int** 当前倒计时，数据如下表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **阶段** | **ready** | **bidding** | **publicity** | **pay** |
| **值** | **30** | **服务器传来的值(竞标30s倒计时)** | **0** | **服务器传来的值(缴纳尾款60s倒计时)** |

* **“round”: int** 当前竞标是第几轮

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **阶段** | **ready** | **bidding** | **publicity** | **pay** |
| **值** | **空** | **服务器传来的值** | **空** | **空** |

* **“biddingTable”: JSONArray** 排序后竞标表，每30s更新一次，包含以下字段

|  |  |  |  |
| --- | --- | --- | --- |
| **字段** | **price** | **address** | **time** |
| **解释** | **出价** | **用户地址** | **竞标时间戳(服务器时间)** |

* **“deal”: double** 得标者最后成交价，数据如下表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **阶段** | **ready** | **bidding** | **publicity** | **pay** |
| **值** | **空** | **空** | **空** | **成交价** |

* **“transfer”: bool** 缴纳尾款是否成功

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **阶段** | **ready** | **bidding** | **publicity** | **pay** |
| **值** | **空** | **空** | **空** | **true/false** |

* **“frozen”: bool** 告知玩家是否被冻结

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **阶段** | **ready** | **bidding** | **publicity** | **pay** |
| **值** | **空** | **空** | **空** | **true/false** |

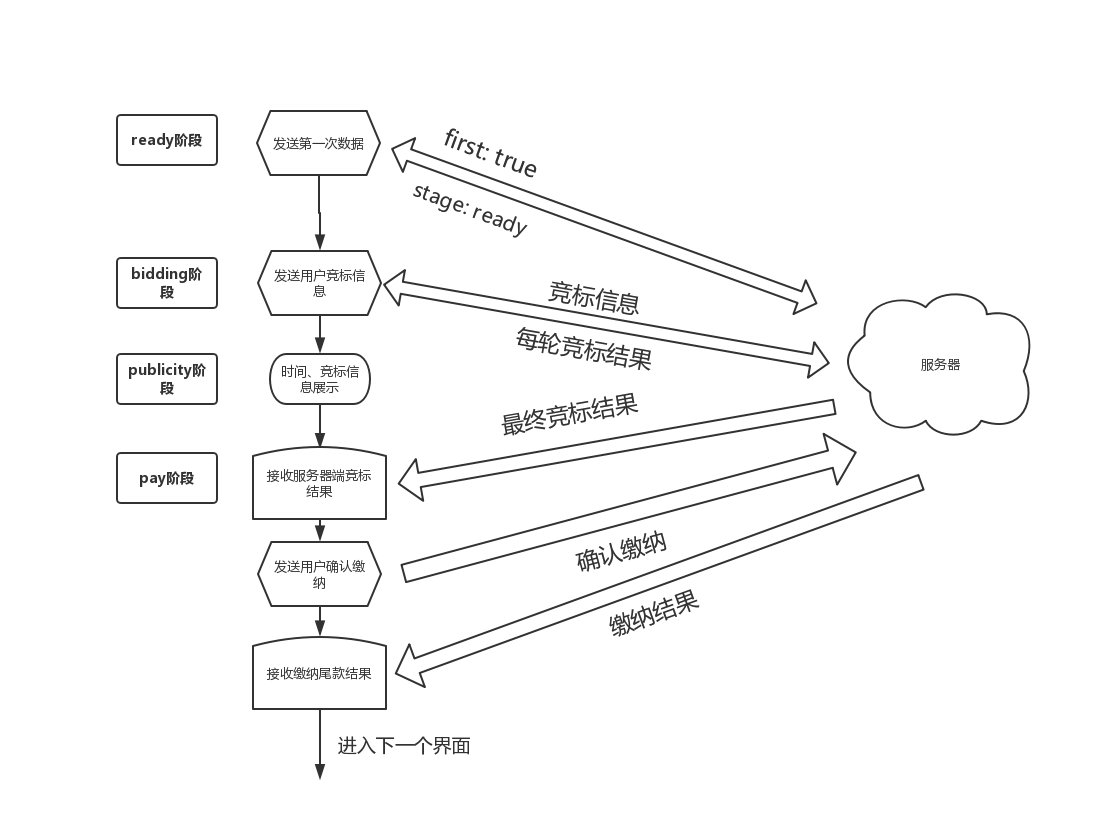
* **“allocate”: bool** 告知玩家分配城池是否完成

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **阶段** | **ready** | **bidding** | **publicity** | **pay** |
| **值** | **空** | **空** | **空** | **true/false** |

以下是连接时返回值，为String数据类型。

* **"connect error"** websocket连接失败
* **"connect success"** websocket连接成功
* **"disConnect" websocket**断开连接

该阶段数据发送与接收逻辑图如下



1. **主战场界面**

=========下面是关于区块链上数据表查询的前后端接口=========

1. 查询玩家信息表

* 功能：返回玩家游戏数据
* URL：<http://localhost:8088/getPlayerTable>
* 输入：（key: value，key统一为String类型）
* **“playerAddress”: String** 玩家地址
* **“signature”: String** 签名，可以是玩家的签名，也可以是项目方的签名，数据结构同上
* 返回：（key: value，key统一为String类型）返回数据结构可能是JSONObject或者是JSONArray
* 暂时保存在我电脑上，等电脑修好①

1. 查询游戏中城池信息表

* 功能：返回每一场匹配中城池数据
* URL：<http://localhost:8088/getCitiesTable>
* 输入：
* **“gameId”: long** 游戏匹配的Id
* **“cityId”: long** 城池Id
* **“signature”: String** 签名，可以是玩家的签名，也可以是项目方的签名
* 返回：
* 待定

（后续部分还没实现）

=========下面是关于区块链上用户资产查询、操作的前后端接口=========

1. 创建游戏道具

* 功能：项目方创建游戏资产，当前版本权利只交给项目方，玩家不可以创建资产，但是可以获取游戏资产
* URL：<http://localhost:8088/createGameItem>
* 输入：
* **“initialSupply”: long** 资产发行量
* **“uri”: String** 统一资源标志符，用于解析链上资产，同类资产仅有一个id和uri（后续资产和uri的对应表我会定下来，参考模板为“uri://siege-class-name-id”）
* **“signature”: String** 项目方签名
* 返回
* 待定

1. 分发游戏资产

* 功能：玩家无法创建游戏资产，通过游戏可以获取游戏资产，由项目方进行发放。
* URL：<http://localhost:8088/issueGameItem>
* 输入：
* **“id”: long** 资产在链上的Id
* **“to”: String** 接收方地址
* **“value”: long** 数量
* **“signature”: String** 项目方签名，资产直接从项目方地址转至to地址
* 返回：
* 待定

1. 游戏资产游戏内转移

* 功能：将玩家A中的资产1转至玩家B地址中。
* URL：<http://localhost:8088/transferGameItem>
* 输入：
* **“from”: String** 玩家A地址
* **“to”: String** 玩家B地址
* **“id”: long** 资产在链上的Id
* **“value”: long** 数量
* **“data”: String** 交易附言，可为空
* **“signature”: String** 地址from的签名
* 返回
* 待定

1. 查询某一资产数量

* 功能：查询指定id资产的余额
* URL：<http://localhost:8088/balanceOfGameItem>
* 输入：
* **“owner”: String** 拥有者地址
* **“id”: long** 资产在链上的Id
* **“signature”: String** 签名，可以是玩家的签名，也可以是项目方的签名
* 返回：
* 待定

1. 查询用户所有资产

* 功能：查询某一用户所拥有的所有资产
* URL：<http://localhost:8088/propertyOfGameItem>
* 输入：
* **“owner”: String** 拥有者地址
* **“signature”: String** 签名，可以是玩家的签名，也可以是项目方的签名
* 返回：
* 待定

1. 获取资产uri

* 功能：通过资产id获取对应的uri
* URL：<http://localhost:8088/getUriGameItem>
* 输入：
* **“id”: long** 资产在链上的id
* **“signature”: String** 签名，可以是玩家的签名，也可以是项目方的签名
* 返回：
* 待定

1. 获取资产id

* 功能：通过资产uri获取对应的id
* URL：<http://localhost:8088/getIdGameItem>
* 输入：
* **“uri”: String** 资产uri
* **“signature”: String** 签名，可以是玩家的签名，也可以是项目方的签名
* 返回：
* 待定