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** |

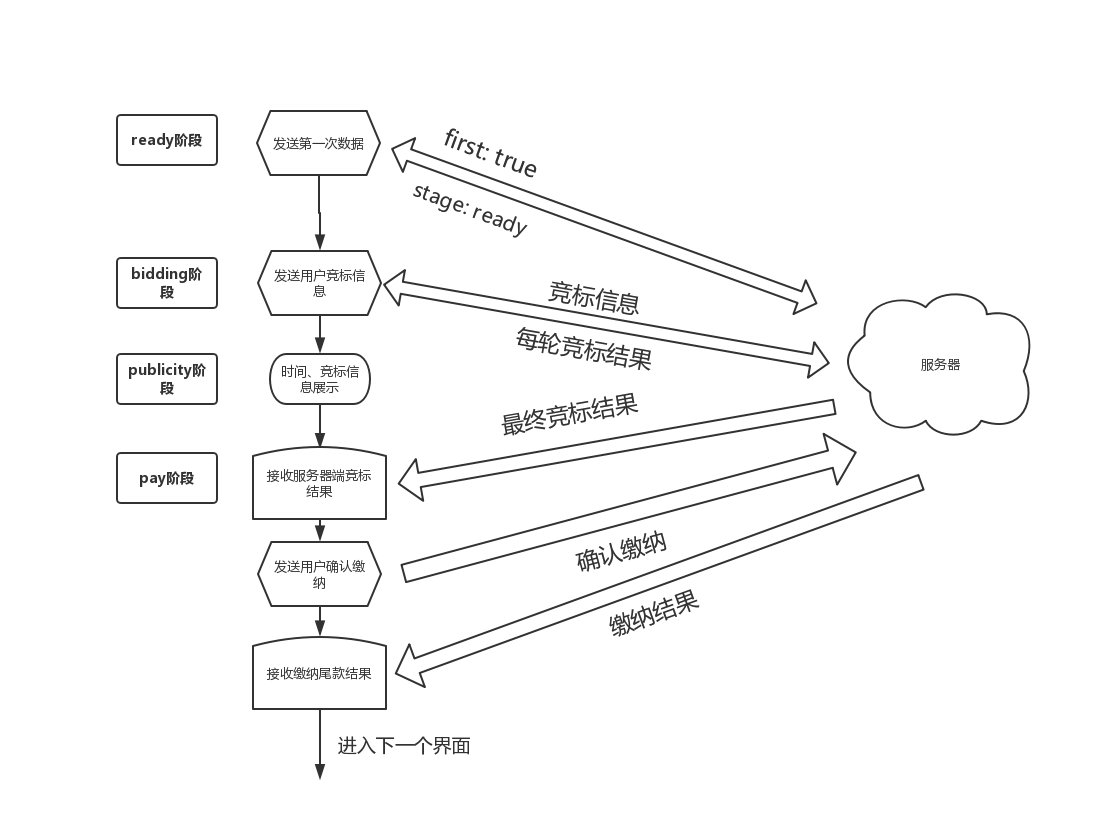
* **“allocate”: bool** 告知玩家分配城池是否完成

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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**

* 功能：城池界面，玩家可以进攻和防御
* 使用websocket进行连接：ws://localhost:8088/WebSocket/cityMap/{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（必选）
* **“address”: string** 用户地址（必选）
* **“operation”: String** 玩家操作
* **“cityId”: int** 用户操作的城池id
* **“target”: String** 目标（对手）地址
* **“choice”: int** 防御者选择
* **“price”: double** 城池的价格
* **“bonus”: double** 城池产生的收益
* **“signature”: String** 用户签名

阶段选择

|  |  |  |
| --- | --- | --- |
| **变量** | **first** | |
| **状态** | **true** | **false** |
| **解释** | **只进行注册操作** | **进行后续operation操作** |

operation选择

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **operation** | **occupy** | **leave** | **attack** | **defense** |
| **解释** | **玩家占领空城** | **玩家离开城池** | **玩家进攻** | **玩家防御** |
| **cityId** | **该城池id** | **该城池id** | **进攻的城池id** | **自己城池的id** |
| **target** | **空** | **空** | **防御者的地址** | **进攻者的地址** |
| **choice** | **空** | **空** | **空** | **0弃城;1防御** |
| **price** | **该城池价格** | **空** | **空** | **空** |
| **bonus** | **空** | **该城池产生的分红** | **空** | **该城池产生的分红** |
| **signature** | **缴纳费用需要** | **空** | **空** | **空** |

* 返回：
* **“stage”: string** 游戏阶段
* **“timer”: int** 该阶段倒计时，每秒更新
* **“produceRate”: double** 实时产出率
* **“bonusPool”: double** 奖池总金额
* **“cityBonus”: JSONArray** 各个城池目前的分红(以上3项每十秒更新)
  + **“cityId”: int** 城池id
  + **“producedBonus”: double** 目前的分红

（以上为城池全局信息，每个玩家都要收到）

* **“cityId”: int** 城池id
* **“cityName”: string** 城池名称(固定)
* **“defenseIndex”: double** 城池防御指数(固定)
* **“realtimePrice”: double** 城池当前价格
* **“ifBeOccupied”: bool** 城池是否被占领
* **“belongPlayer”: string** 城池的拥有者地址
* **“producedBonus”: double** 城池目前的分红

（以上为城池数据，在玩家连接成功时发送给玩家初始化城池数据，然后每10s更新）

* **“operation”: string** 玩家操作
* **“status”: bool** 玩家操作是否成功
* **“situation”: string** 玩家当前面临的状况

（以上为玩家进行操作之后需要的字段）

各操作返回的数据如下表所示

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 玩家发起的操作 | occupy | leave | attack | defense/0 | defense/1 |
| **返回字段** | | | | |  |
| **operation** | **occupy** | **leave** | **attack** | **defense** | **defense** |
| **status(**是否接收到城池修改信息，若为false，以下数据均不发送**)** | **true/false** | **true/false** | **true/false** | **true/false** | **true/fasle** |
| **situation(**告知对手状态**)** | **空** | **空** | **beAttackRequest** | **getCity** | **beforeBattle** |
| **opponent** | **空** | **空** | **对手address** | **空** | **空** |
| **cityId（**标志性信息，收到之后代表需要更新城池map**）** | **占领的城池id** | **离开的城池id** | **空** | **离开的城池id** | **空** |
| **realtimePrice** | **占领时费用** | **空** | **空** | **空** | **空** |
| **ifBeOccupied** | **true** | **false** | **空** | **true** | **空** |
| **belongPlayer** | **占领者地址** | **0x0** | **空** | **占领者地址** | **空** |
| **producedBonus** | **0.** | **0.** | **空** | **0.** | **空** |

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. **战斗准备和战斗界面**

* 功能：玩家购买卡牌、进行PVP对战
* 使用websocket进行连接：ws://localhost:8088/WebSocket/cityMap/{gameId}/{battleId}，gameId即为匹配成功时获取的值，battleId形式为:

address1&&address2&&cityId，其中address1是进攻者的地址，address2为防御者的地址，cityId为目标城池id。

* 输入：
* **“first”: bool** 是否为第一次发送数据，用于用户信息注册（必选）
* **“address”: string** 用户地址（必选）
* **“operation”: String** 玩家操作
* **“type”: int[]** 购买的士兵集合数组(如：[1, 2, 3, 4])
* **“price”: double** 购买士兵总价
* **“quantity”: int**购买士兵的数量
* **“signature”: String** 用户签名
* **“soldier”: int**选择的士兵种类

阶段选择

|  |  |  |
| --- | --- | --- |
| **变量** | **first** | |
| **状态** | **true** | **false** |
| **解释** | **只进行注册操作** | **进行后续operation操作** |

operation选择

|  |  |  |  |
| --- | --- | --- | --- |
| **operation** | **buySoldiers** | **departure** | **pickSoldier** |
| **解释** | **购买士兵操作** | **集结出发** | **出牌** |
| **type** | **购买的士兵数组** | **购买完成的士兵数组** | **空** |
| **price** | **购买士兵的总价** | **购买完成的士兵总价** | **空** |
| **quantity** | **购买的士兵数量** | **购买完成的士兵数量** | **空** |
| **signature** | **必须** | **空** | **空** |
| **soldier** | **空** | **空** | **玩家挑选的士兵** |

* 返回：
* 第一阶段：初始化士兵商店（JSONArray）
* **“type”: int** 士兵种类
* **“description”: string** 士兵介绍
* **“price”: double** 士兵战力
* 第二阶段：购买士兵倒计时
* **“stage”: string “buySoldiers”** 购买士兵阶段
* **“positive”: bool** 该阶段是否正常，错误直接退出
* **“timer”: int** 60s购买士兵倒计时 （以上两项为倒计时）
* **“operation”: string “buySoldiers”**
* **“status”: bool** 支付是否成功
* 第三阶段：集结出发阶段
* **“operation”: string “departure”**
* **“status”: bool** 操作是否成功
* **“ready”: bool** 双方是否都已经集结出发？（以上三项为必须）
* **“opponent”： string** 对手地址
* **“myPoint”: double** 我的士兵战力
* **“opponentPoint”: double** 对方的士兵战力
* **“myQuantity”: int** 我的士兵数量
* **“opponentQuantity” int** 对方士兵数量
* 第四阶段：对战阶段
* **“stage” : string “battle”** 每轮对战倒计时
* **“positive”： bool** 该阶段是否正常，错误直接退出
* **“round”: int** 本轮轮数
* **“timer”: int** 20s倒计时
* **“isOver”: bool** 对战是否结束？（以上五项为必须）
* **“winner”: string** 整局游戏获胜者地址
* **“loser”: string** 整局游戏失败者地址
* **“operation”: string “pickSoldier”** 对战阶段
* **“status”: bool** 操作是否成功
* **“situation”: string** 情况选择
* **“opponentPick”** 对方已经出牌
* **“wait”** 对方未出牌，己方等待
* **“judge”** 本轮判定阶段
* **“result”: string** 战斗结果
* **“win”** 获胜
* **“lose”** 失败
* **“tie”** 平手
* **“opponent”： string** 对手地址
* **“myPoint”: double** 我的士兵战力
* **“opponentPoint”: double** 对方的士兵战力
* **“myQuantity”: int** 我的士兵数量
* **“opponentQuantity” int** 对方士兵数量

=========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