SEER介绍

一，账户体系

SEER的账号名有普通账号名和高级账号名之分，账号名带元音aeiouy但又不带数字和点(.)和横(-)的账号名属于高级账号名，注册费会高于普通账号名。

SEER的账号又分普遍会员和终身会员：

新注册的用户均为普通会员，可消费SEER币升级为终身会员

终身会员特权：

1，使用SEER功能的手续费打折（6折）

2，有权创建见证人、理事会成员、其他系统角色

3，邀请并注册新账号的推广激励

SEER的账户体系组成：

1，资金权限

2，账户权限

3，备注密钥

其中权限的组成为：

1，阀值

2，授权列表含各自权重。

对于以上#2授权的种类为：

1，账户id

2，公钥

3，地址

当授权的权重达到阀值时，则可执行该权限。

举例一：

资金权限的阀值为1

账号seer-dev和公钥SEER646RGdL4gncz7y834wfGfcHECnKdbdVWd6gh9aEYdn3HWyhBjB的权重均达到了阀值

则：seer-dev和SEER646RGdL4gncz7y834wfGfcHECnKdbdVWd6gh9aEYdn3HWyhBjB 两者均可以动用该账号的资金。

示例二：

账户权限的阀值为50

账号seer-dev和seer-dev1、seer-dev2权重各为25

则：要想成功修改该账号，至少需要seer-dev、seer-dev1、seer-dev2三人中的其中2个人的授权（签名），使权重为25+25=50，达到阀值要求的50。

修改账户时注意：最好将备注密钥设成成资金密钥相同

二，SEER见证人

SEER的见证人分成两部分:

1，主力见证人：具备打包出块权限的见证人。该见证人必须配置并运行节点程序以打包出块。

入选条件：抵押排名前21名，并且前2天没有丢块超过50%的。

入选收益：打包出块收入，每出一个块，奖励3 SEER，实时发放至“待领取余额”中。

2，获息见证人：具备获取利息的见证人。

获息见证人包含主力见证人。

入选条件：抵押总量排名前101名即可。

入选收益：抵押的利息收入。

收益发放：新加坡时间每天8：00。

见证人竞选更新时间：新加坡时间每天8：00

利息计算公式：

当天总利息支出： (资金池余额-622080000)/5400

当天见证人的利息收入: 当天总利息支出 * 个人抵押金/获息见证人总抵押金

初始资金池余额为20亿

资金池的消耗：见证人的出块支出和利息支出均从资金池中扣除

资金池的收入：用户在SEER上支付的手续费，60% 注入资金池

利率说明:
排名对利息的利率有细微影响;第一名比第N名高0.02%*(N-1)。
第1名的利率比第101名高0.02%*(101-1) = 2%。
假设第一名的日利率是0.1%
那么
第2名是0.09998%
第3名是0.09996%
第4名是0.09994%
.....
第101名是0.098%

三、用户发行的资产

Seer允许用户创建各种自定义资产(UIA)。自定义资产的应用场景数不胜数。比如，UIA可被用来代替简单的活动门票，存入合格用户的手机钱包中，在进入活动现场时进行实时验证。同样，UIA可被用来进行众筹、所有权追踪，甚至是代表公司的股权。

资产的属性列表如下：
1，名字
2，精度（几位小数）
3，发行人（创建者）
4，最大供应量（总量上限）
5，市场交易手续费率（百分比比如0.1%）
6，市场交易手续费上限
7，资产开通的权限
8，资产已激活的权限
9，跟SEER主资产的手续费汇率（手续费池汇率）
10，持有人白名单
11，持有人黑名单
12，市场交易授权名单（只可以跟什么资产交易）
13，市场交易黑名单（禁止跟什么资产交易）
14，动态参数对象，其包括：当前供应量；隐私交易量；收集到的手续费；手续费池余额

不同使用场景的法律法规可能天差地别，所以Seer提供了资产管理功能来帮助创建者来合规发行和管理资产，功能清单如下：
1，收取交易手续费
2，要求资产持有人预先加入白名单
3，发行人可将资产收回
4，所有转账必须通过发行人审核同意
5，禁止隐私交易

需要注意的事，以上权限也是可以关闭的。
一旦关闭某项权限，便不可重新打开。
权限需要激活才会生效。

四、SEER相关程序说明

下载网址：
witness_node/cli_wallet：https://github.com/seer-project/seer-core-package/releases
Seer-UI安装包:https://github.com/seer-project/seer-UI-package/releases

1，witness_node
命令行方式运行
witness_node为重钱包
运行witness_node的计算机可从p2p网络中同步完整的区块链数据，同步完成后该计算机成为一个全数据节点（具有完整的区块数据）。

运行示例:
A,windows
witness_node.exe --p2p-endpoint=0.0.0.0:1888 --rpc-endpoint=0.0.0.0:8002

B,ubuntu
./witness_node --p2p-endpoint=0.0.0.0:1888 --rpc-endpoint=0.0.0.0:8002

其中8002为提供给cli_wallet和GUI钱包连接的端口

2,cli_wallet
witness_node为轻钱包
命令行方式运行
运行cli_wallet连接至witness_node，可进行各种操作（纯粹的功能比GUI前端钱包丰富），包括注册用户、升级会员、创建资产、升级资产、投票、查询余额、查询账号历史、查询区块、查询区块参数.....

运行示例：
A,windows
cli_wallet.exe -s ws://127.0.0.1:8002

B,ubuntu
./cli_wallet -s ws://127.0.0.1:8002

其中127.0.0.1为witness_node程序的IP，这里为本机，8002为witness_node开放的端口

3,Seer-UI
GUI版本轻钱包，可至https://wallet.seer.best 或下载GUI安装包进行体验

五、SEER预测类型介绍
1，LMSR

LMSR类型的预测，用户的参与量可以是负数，负数即卖出。

LMSR参与量的单位为“份”，价格是需要根据份数即时计算的，比如卖出1份价格为1，但卖出2份价格未必是2，前端可对相应数量的买卖价格做初步计算。

LMSR玩法的优势：参与预测的过程可以伴随自由的买卖，用户可以在预测结果出来之前卖出获利或者止损。预测的参与量的买卖可以使预测倾向流动加快，即更快的向概率最大的选项倾斜。

2，PVP
用户自由参与预测，该预测类型没有庄家，预测参与资金全部分给预测正确者。

3，Advanced
高级预测类型
A，房主可设置各个选项的赔率
B，房主可随时修改赔率
C，用户预测以参与时间点的赔率计算中奖回报
D，房间有预设资金池的概念
E，在预测进程过程中，房主可往资金池添加资金，不可提取资金
F，预测未开启时，房主可添加或提取资金池资金
G，用户参与时，若当前总资金（含资金池和用户参与资金）可能不够派奖，则参与失败

六、预测业务流程说明

1，创建平台

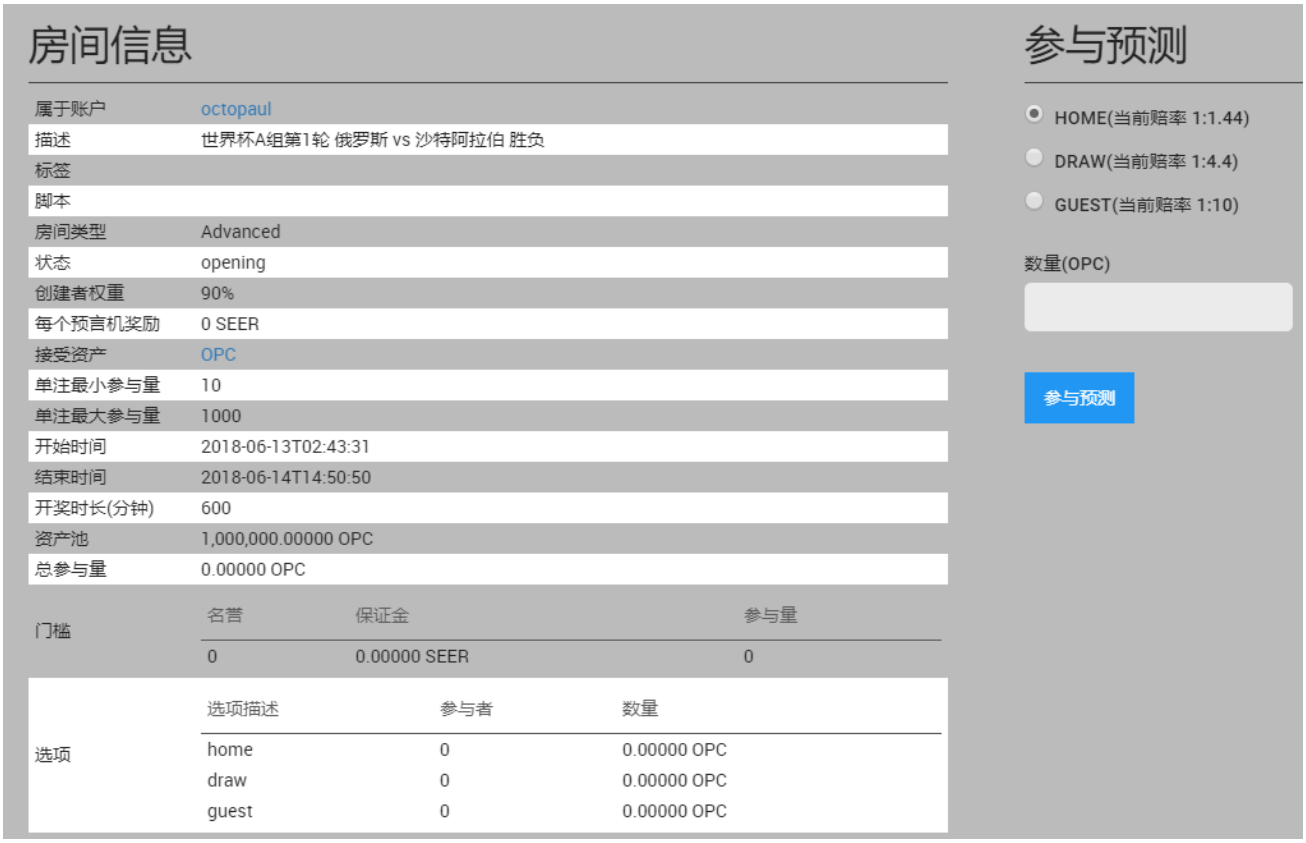
前置条件:钱包当前账号为平台账号

操作入口:钱包右上角菜单->平台

说明：保证金的金额关系到可同时创建的房间的数量。当前为：每10万保证金可创建1个房间。

操作者：房主

2，创建房间/更新房间



前置条件:钱包当前账号为自己的账号

操作入口: 钱包右上角菜单->平台/房间->创建/更新房间

注意事项：

A，关闭状态时，可随时更新房间

B，接受资产一旦设定即不可修改

C，房间类型一旦设定即不可修改

D，如果房间类型为Advanced，任何时间均可修改赔率

操作者：房主

3,开启预测

当房间状态为“关闭”状态时,可开启预测

前置条件:钱包当前账号为平台账号

操作入口:钱包右上角菜单->平台->对指定房间点“开启”

说明：

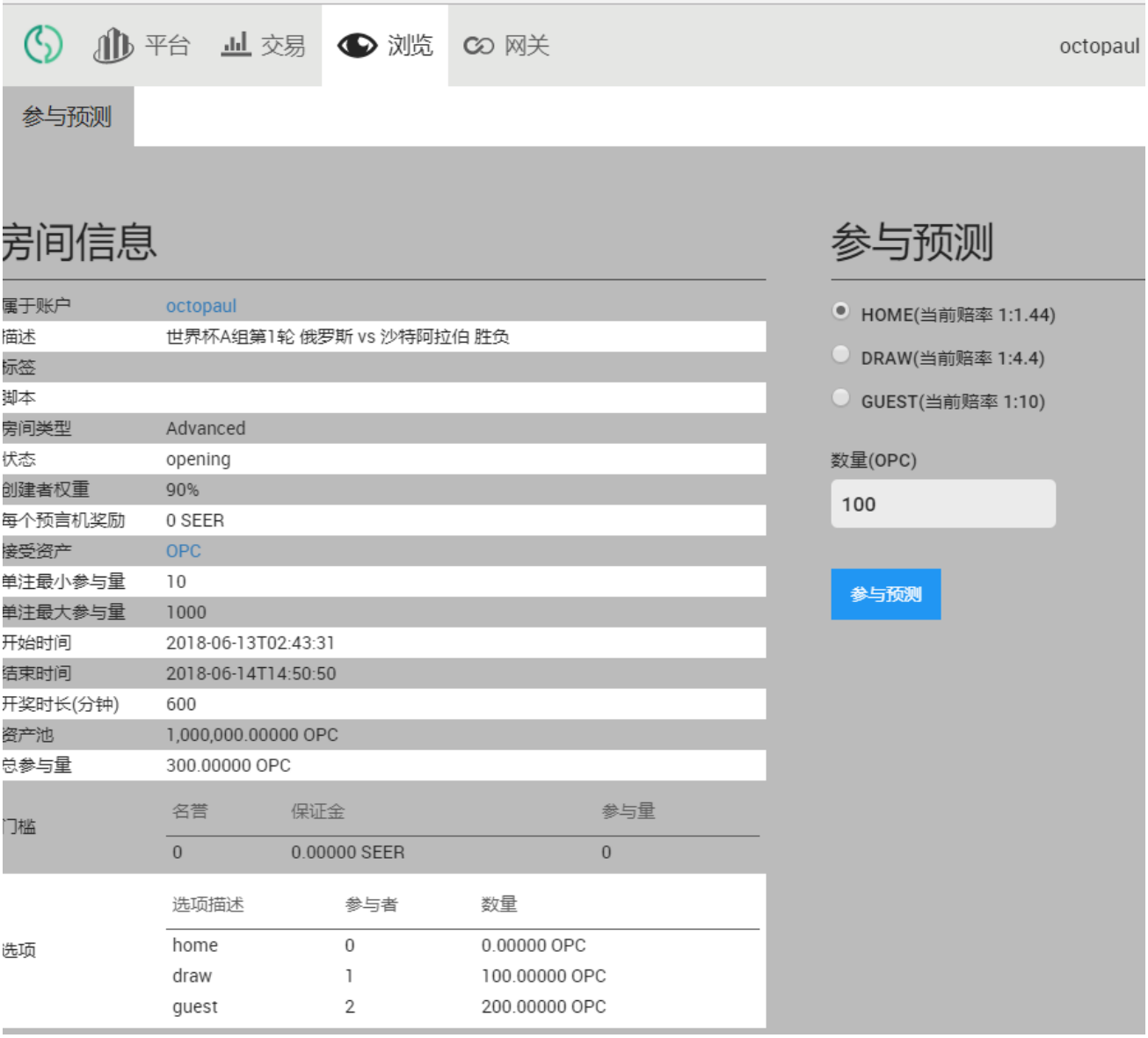
开始时间/结束时间：玩家只可以在该时间段内参与预测

结束时间/结束时间+开奖时长：该时间内房主/预言机可对该预测输入结果

结束时间+开奖时长---：在此时间之后，任何人均可以触发该预测的结算和派奖

操作者：房主

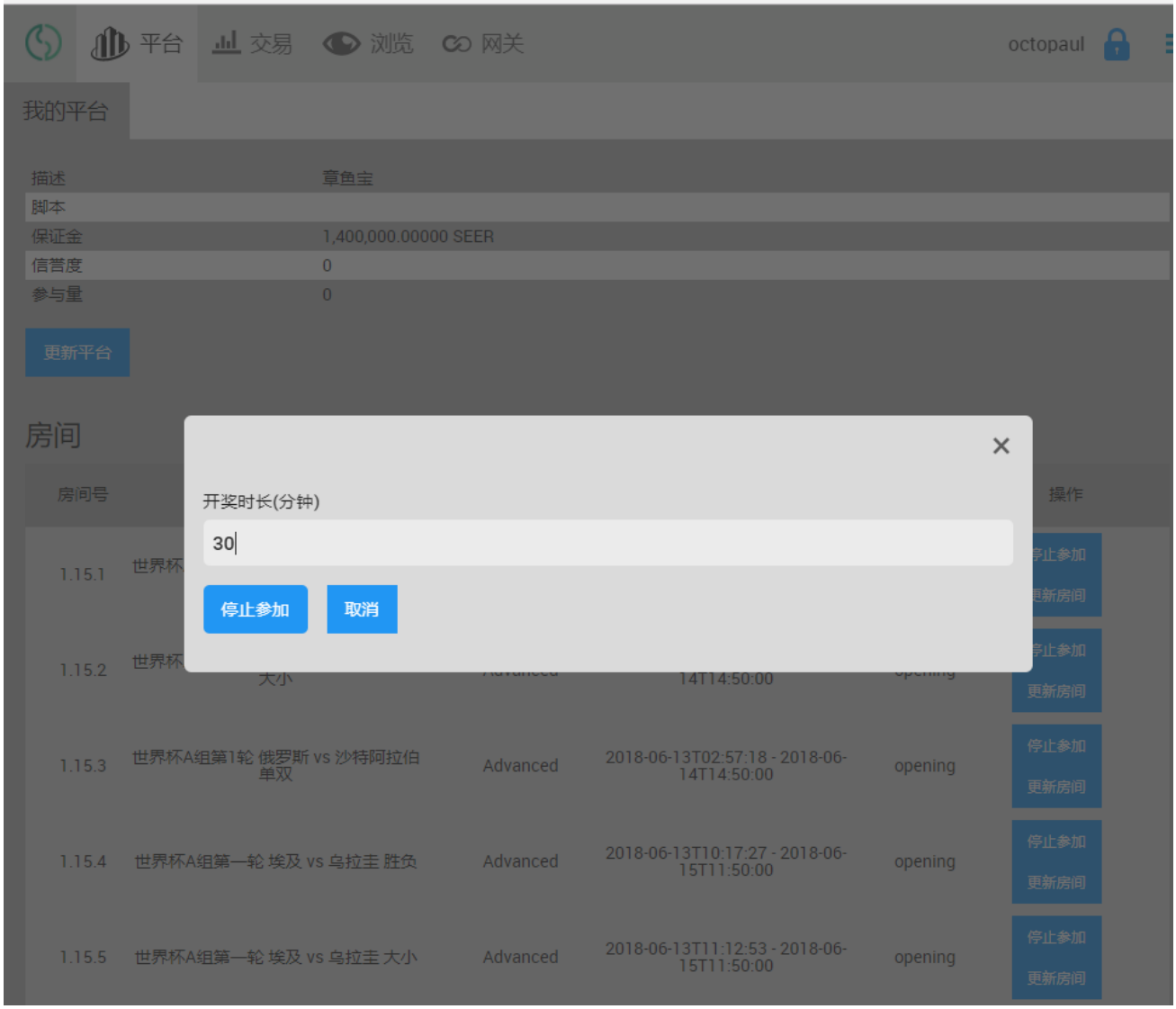
4，参与预测



前置条件:钱包当前账号为SEER用户账号

操作入口:钱包首页的平台->选择平台->选择房间->参与预测
操作者：任何SEER用户

5，提前停止预测



前置条件:钱包当前账号为平台账号
操作入口:钱包右上角菜单->平台->对指定房间点“停止参加”

房主可提前停止预测，使用户无法再参与预测，此时可重新设置“开奖时长”。
操作者：房主

6，输入预测结果
操作入口:钱包右上角菜单->平台->对指定房间点“输入结果”/
预言机输入:平台->进入指定平台->对指定房间点"Oracle输入"

当时间到达结束时间，并在开奖时长范围内时，可以对房间输入结果。

操作者：预言机和房主

7，结算
前置操作:当前账号为自己账号
房主操作入口:钱包右上角菜单->平台->对指定房间点“结算"
普通用户操作入口:平台->进入指定平台->对指定房间点"结算"

可结算前提：开奖时间结束以后，并且有人输入了预测结果（房主及预言机）
所做操作：计算预言机和房主的输入统计出最终结果，并惩罚作恶的预言机/奖励诚实的预言机

操作者：任何SEER用户

8，派奖

前置操作:当前账号为自己账号

房主操作入口:钱包右上角菜单->平台->对指定房间点“派奖"

普通用户操作入口:平台->进入指定平台->对指定房间点"派奖"

可结算前提：已经完成结算

所做操作：给中奖者派发奖励并将剩余余额返给房间创建者

操作者：任何SEER用户

七、前端开发接口说明

1，查询数据接口

前端通过websocket连接至witness_node,已经提供了一个开发包为seerjs，前端开发者只需要调用该接口即可。

示例一：

```
var Apis = require("seerjs-ws").Apis;
Apis.instance().db_api().exec("get_seer_room", [this.props.params.room_id, 0, 500]).then(r => {
  this.setState({room: r});
});
```

说明：*Apis*.instance().db_api().exec为执行接口，get_seer_room为底层提供的API名字，[]内数据为参数集

示例二：

```
Apis.instance().db_api().exec("get_oracles", [this.state.room.option.allowed_oracles]).then(houses => {
  var ret = [];
  houses.forEach(function(item,index){
    ret.push(item.owner);
  });
  this.setState({oracles:ret});
});
```

2，交易操作的接口

绝大多数操作的接口均进行了初步封装，开发者可至以下目录参考

Seer-UI\app\actions\

代码示例：

```
import SeerActions from "../../actions/SeerActions";
let args = {
  issuer: user_id,
  room: this.state.room.id,
  type: 0,
  input: [this.state.checked_item],
  input1: [],
  input2: [],
  amount: parseInt(this.state.amount * this.state.precision)
};
SeerActions.participate(args);
```

而在SeerActions.js中

```
participate(args) {
  let tr = WalletApi.new_transaction();
  tr.add_type_operation("seer_room_participate", args);
  return (dispatch) => {
    return WalletDb.process_transaction(tr, null, true).then(result => {
```

```
            dispatch(true);
        }).catch(error => {
            console.log("seer_room_participate error ----->", error);
            dispatch(false);
        });
    };
}
```

对不同操作的数据序列化，均在seerjs完成，应用开发者可以不用关心这一层级，只需要在将各种操作(如创建房间、转账、创建资产、创建预测、开启预测等等.....)所需参数传递进去即可

3，常用查询接口

A，读取指定用户的平台和房间

```
Apis.instance().db_api().exec("get_house_by_account", [this.props.account.get("id")]).then((results) => {
    this.setState({house: results,rooms:[]});
    if(results)
        results.rooms.forEach(room => {
            Apis.instance().db_api().exec("get_seer_room", [room, 0, 10]).then(r => {
                this.state.rooms.push(r);
                this.forceUpdate();
            });
        });
});
```

4，常用操作参数列表

A，创建平台

```
_createHouse() {
    let guaranty= parseInt(this.state.guaranty*100000)
    let args = {
        issuer: this.props.account.get("id"),
        guaranty: guaranty,
        description: this.state.description,
        script: this.state.script
    };
    SeerActions.createHouse(args);
}
```

B，更新平台

```
_updateHouse() {
    let core_asset = ChainStore.getAsset("1.3.0");
    let guaranty = parseInt(this.state.guaranty) * Math.pow(10, core_asset.get("precision"));
    let args = {
        issuer: this.props.account.get("id"),
        guaranty: guaranty,
        claim_fees: 0,
        description: this.state.description,
        script: this.state.script,
        house: this.props.house.get("id")
    };
    SeerActions.updateHouse(args);
}
```

C,创建房间

```
_createRoom() {
    let args = {
        issuer: this.props.account.get("id"),
        label: this.state.label.filter(l => {return l.trim() != ""}),
        description: this.state.description,
        script: this.state.script,
```

```
      room_type: this.state.room_type,
      option: {
        result_owner_percent: parseInt(this.state.result_owner_percent*100),
        reward_per_oracle: parseInt(this.state.reward_per_oracle*100000),
        accept_asset: this.state.accept_asset,
        minimum: parseInt(this.state.min*this.state.accept_asset_precision),
        maximum: parseInt(this.state.max*this.state.accept_asset_precision),
        start: new Date(),
        stop: new Date(),
        input_duration_secs: 60,
        filter: {
          reputation: this.state.reputation,
          guaranty: parseInt(this.state.guaranty*100000),
          volume: this.state.volume
        },
        allowed_oracles:[],
        allowed_countries:[],
        allowed_authentications:[]
      },
      initial_option:{
        room_type: this.state.room_type,
        selection_description: this.state.selections,
        range: this.state.selections.length
      }
    };
    if (this.state.room_type == 0) {
      args.initial_option.lmsr = {
        L: parseInt(this.state.L)
      };
    } else if (this.state.room_type == 2) {
      args.initial_option.advanced = {
        pool: parseInt(this.state.pool),
        awards: this.state.awards.map(a => {return parseInt(a*10000);})
      };
    }
    SeerActions.createRoom(args);
}
```

D,更新房间

更新高级玩法的赔率：

```
_updateRoomAward() {
    let args = {
      issuer: this.props.account.get("id"),
      room:this.props.room.get("id"),
      new_awards: (this.props.room.get("room_type") == 2)?this.state.awards.map(a => {return parseInt(a*10000);}):null
    };

    SeerActions.updateRoom(args);
}
```

房间未开启时可更新房间

```
_updateRoom() {
    let args = {
      issuer: this.props.account.get("id"),
      room:this.props.room.get("id"),
      description: this.state.description,
      script: this.state.script,
      option: {
        result_owner_percent: parseInt(this.state.result_owner_percent*100),
        reward_per_oracle: parseInt(this.state.reward_per_oracle*100000),
        accept_asset: this.state.accept_asset,
        minimum: parseInt(this.state.min*this.state.accept_asset_precision),
        maximum: parseInt(this.state.max*this.state.accept_asset_precision),
```

```
            start: new Date(),
            stop: new Date(),
            input_duration_secs: 60,
            filter: {
                reputation: this.state.reputation,
                guaranty: parseInt(this.state.guaranty*100000),
                volume: this.state.volume
            },
            allowed_countries:[],
            allowed_authentications:[]
        },
        initial_option:{
            room_type: this.state.room_type,
            selection_description: this.state.selections,
            range: this.state.selections.length
        }
    };

    if (this.state.room_type == 0) {
        args.initial_option.lmsr = {
            L: parseInt(this.state.L)
        };
    } else if (this.state.room_type == 2) {
        args.initial_option.advanced = {
            pool: parseInt(this.state.pool),
            awards: this.state.awards.map(a => {return parseInt(a*10000);})
        };
    }

    SeerActions.updateRoom(args);
}
```

E,参与预测

```
onSubmit() {
    let obj = ChainStore.getAccount(AccountStore.getState().currentAccount);
    if (!obj) return;
    let id = obj.get("id");
    let args = {
        issuer: id,
        room: this.state.room.id,
        type: 0,
        input: [this.state.checked_item],
        input1: [],
        input2: [],
        amount: parseInt(this.state.amount * this.state.precision)
    };
    SeerActions.participate(args);
}
```

F,输入预测结果

```
onSubmit() {
    let args = {
        issuer: this.props.account.get("id"),
        room: this.state.room.id,
        input: [this.state.input]
    };
    SeerActions.inputRoom(args);
}
```

G，结算

```
finalRoom(room) {
    var args = {
        issuer: this.props.account.get("id"),
```

```
        room: room.id,
    };
    SeerActions.finalRoom(args);
}
```

H，派奖

```
settleRoom(room) {
    var args = {
        issuer: this.props.account.get("id"),
        room: room.id,
    };
    SeerActions.settleRoom(args);
}
```

I,创建预言机

```
_createOracle() {
    let args = {
        issuer: this.props.account.get("id"),
        guaranty: parseInt(this.state.guaranty*100000),
        description: this.state.description,
        script: this.state.script
    }
    SeerActions.createOracle(args);
}
```

J，更新预言机

```
_updateOracle() {
    let args = {
        issuer: this.props.account.get("id"),
        guaranty: parseInt(this.state.guaranty*100000),
        description: this.state.description,
        script: this.state.script,
        oracle: this.props.oracle.get("id")
    };
    SeerActions.updateOracle(args);
}
```

K，预言机输入结果

```
onSubmit() {
    let args = {
        issuer: this.props.account.get("id"),
        oracle: this.state.oracle.id,
        room: this.state.room.id,
        input: [this.state.input]
    };
    SeerActions.inputOracle(args);
}
```