此文档用来说明智能合约首次部署以及更新的步骤。

# 首次部署

## 购买内存

cleos --wallet-url http://127.0.0.1:6666 system buyram useraaaaaaag useraaaaaaaj "10 SYS"

## 部署合约

Bac游戏合约部署：

cleos --wallet-url http://127.0.0.1:6666 set contract gamemallards ./ mallard.wasm mallard.abi

Sic游戏合约部署：

cleos --wallet-url http://127.0.0.1:6666 set contract game12lizard ./ lizard.wasm lizard.abi

## 发币（可选）

cleos --wallet-url http://127.0.0.1:6666 set contract useraaaaaaaj /home/eos/eosio.contract/build/eosio.token/ eosio.token.wasm eosio.token.abi

cleos --wallet-url http://127.0.0.1:6666 push action useraaaaaaaj create '["useraaaaaaai","100000000.0000 TES"]' -p useraaaaaaaj

cleos --wallet-url http://127.0.0.1:6666 push action useraaaaaaaj issue '["useraaaaaaab","100000.0000 TES","xxx"]' -p useraaaaaaai

cleos --wallet-url http://127.0.0.1:6666 push action useraaaaaaaj issue '["useraaaaaaac","10000.0000 TES","xxx"]' -p useraaaaaaai

cleos --wallet-url http://127.0.0.1:6666 push action useraaaaaaaj issue '["useraaaaaaad","10000.0000 TES","xxx"]' -p useraaaaaaai

cleos --wallet-url http://127.0.0.1:6666 get currency balance useraaaaaaaj useraaaaaaab "TES"

cleos --wallet-url http://127.0.0.1:6666 get currency balance useraaaaaaaj useraaaaaaac "TES"

cleos --wallet-url http://127.0.0.1:6666 get currency balance useraaaaaaaj useraaaaaaad "TES"

## 初始化添加合约支持币种

cleos --wallet-url http://127.0.0.1:6666 push action gamemallards initsymbol '["useraaaaaaaj","TES","0.1000 TES"]' -p gamemallards

cleos --wallet-url http://127.0.0.1:6666 push action gamemallards initsymbol '["eosio.token","SYS","0.1000 SYS"]' -p gamemallards

# 更新合约

## 备份原合约

为防止误操作，部署了新结构合约无法恢复，更新前要备份原合约的wasm、abi。

## 调用更新接口

SC::upgrading(1)

修改一个标志位upgrading为true，默认false。（当upgrading为true时，非托管下dealerseed以及托管下的serverseed两个合约均不可被执行，也就是说，当标志位upgrading为true时，等待所有桌本轮结束，均不可开启新一轮）

## 校验tablesinfo

校验所有table的status是否只有两种状态：ROUND\_END或PAUSED

## 导出tablesinfo表到本地

Server端mysql储存当前所有链上状态表数据，通过getTableRows。

## 删除tablesinfo

Tablesinfo数据全部导出成功后，调用clear12cache接口删除所有链上状态表ROUND\_END或PAUSED数据项（理想状态下，应该所有的表都属于这两种状态，如果不是的，可以等待轮次结束变成ROUND\_END）。

## 部署新结构合约

部署包含数据结构修改的新合约

## 导入数据到新结构

调用import12data，按照参数列表传入所有桌相关字段（table fields），轮次字段数据不需要，因为上一轮已经结束了，新的一轮未开始。

**注意**：导入时，每桌字段upgrading标志位写死的仍旧是true，以保证全部数据正常以后，手动控制结束更新。

### import12data

**一、定义：**ACTION import12data(uint64\_t tableId, uint64\_t tableStatus, uint64\_t cardBoot, name dealer, bool trusteeship, bool isPrivate, asset dealerBalance, asset oneRoundMaxTotalBet\_BP, asset minPerBet\_BP, asset oneRoundMaxTotalBet\_Tie, asset minPerBet\_Tie, asset oneRoundMaxTotalBet\_Push, asset minPerBet\_Push, asset oneRoundDealerMaxPay, asset minTableDeposit, float commission\_rate\_agent, float commission\_rate\_player, bool upgradingFlag, extended\_symbol amountSymbol, std::vector<uint16\_t> validCardVec);

**二、参数列表：**tablesinfo中所有table相关的属性，tablesinfo中的字段可分为table属性以及round属性，其中table属性包括：

1. std::vector<uint16\_t> validCardVec; // newtable init & new round check.
2. uint64\_t tableId; // table fix.
3. uint64\_t tableStatus; // round stage.
4. uint64\_t cardBoot;
5. name dealer; // table owner.
6. bool trusteeship; // table flag.
7. bool isPrivate; // table flag.
8. asset dealerBalance; // table field.
9. asset oneRoundMaxTotalBet\_BP;
10. asset minPerBet\_BP;
11. asset oneRoundMaxTotalBet\_Tie;
12. asset minPerBet\_Tie;
13. asset oneRoundMaxTotalBet\_Push;
14. asset minPerBet\_Push;
15. asset oneRoundDealerMaxPay;
16. asset minTableDeposit;
17. extended\_symbol amountSymbol;
18. float commission\_rate\_agent;
19. float commission\_rate\_player;
20. bool upgradingFlag;
21. string redundancy;

**三、调用方式：**push action game12lizard import12data '[180,0,"useraaaaaaab","0","0","100.0000 SYS","2.0000 SYS","3.0000 SYS","4.0000 SYS","5.0000 SYS","6.0000 SYS","7.0000 SYS","8.0000 SYS","9.0000 SYS","10.0000 SYS","11.0000 SYS","12.0000 SYS","13.0000 SYS","14.0000 SYS","15.0000 SYS","16.0000 SYS","17.0000 SYS",0.003,0.002,"0",{"symbol":"4,TES","contract":"useraaaaaaaj"}]'

**四、权限：**useraaaaaaak

## 更新完成

SC::upgrading(0)，将所有桌的upgrading标志位置为false，更新完成，可以正常运行了。