阶段信息：

//自己定义的状态，从10开始

**const** (

GAME\_STATUS\_CALL = 10 + iota // 叫分阶段

GAME\_STATUS\_GETMSG //抢地主阶段

GAME\_STATUS\_READ //准备阶段

GAME\_STATUS\_PLAY // 游戏操作阶段

)

游戏指令：

创建房卡：

MSG\_GAME\_FK\_CREATEDESK = 400017 //创建房卡桌子

**type** GFkCreateDesk **struct** { //

Id int32

FkNo string //房间号

FkInfo string //房间规则信息

}

//房卡配置信息 ,处理为josn转string 放入kfinfo

**type** GATableConfig **struct** {

GameModle int //配置-专场 1,娱乐专场（积分模式） 2，土豪专场（金币模式）

PlayerNum int //配置-牌局人数 （二人斗地主只有两个人）

MatchCount int //配置-局数

GameType int //配置-玩法模式 1，正常模式 2,，二人斗地主 3，闪电斗地主 2，癞子模式

PayType int //配置-支付条件 1，房主支付

BaseScore int //配置-底分

CallType int //配置-叫分方式 1，叫分 2，抢地主

Boom int //配置-炸弹 3，3炸 4，4炸 5，5炸 999`，无上限

CanSelect []int //可选， 里面有1， 不可三带一对 2，不可4带两队 3，显示剩余手牌数量 没有选择默认为0

}

创建房卡桌子应答

MSG\_GAME\_FK\_CREATEDESK\_REPLY = 400018 //创建房卡桌子应答

**type** GFkCreateDeskReply **struct** {

Id int32

Result int32

Err string

FkNo string //房间号

GameType int32

RoomType int32

GradeType int32

}

加入房卡场

MSG\_GAME\_FK\_JOIN = 400019 //加入房卡场

**type** GFkJoinToGame **struct** {

Id int32

Account string

Uid int64

Nick string

Sex int32

Head string

Lv int32

Coin int64

Token string

Robot bool

FkNo string //房间号

}

加入房卡场应答

MSG\_GAME\_FK\_JOIN\_REPLY = 400020 //加入房卡场应答

**type** GFkJoinReply **struct** {

Id int32

Result int32 //0成功，其他失败

Err string

}

房卡信息通知

MSG\_GAME\_INFO\_ROOM\_NOTIFY //房间信息通知4100011

//房卡信息返回

**type** GTableInfoReply **struct** {

Id int32

TableId string //房间号

Config GATableConfig //房间配置

IsCanSee int //能否查看剩余手牌 1,可以 2，不可以

}

游戏匹配成功信息返回

MSG\_GAME\_INFO\_AUTO\_REPLY //410014,游戏随机匹配成功的数据

//匹配信息返回

**type** GInfoAutoGameReply **struct** {

Id int32

Seat []GSeatInfo

}

//座位信息

**type** GSeatInfo **struct** {

Uid int64

Nick string

Ready bool

Cid int32 //椅子号

Sex int32

Head string

Lv int32

Coin int64

}

游戏状态改变

MSG\_GAME\_INFO\_STAGE = 410030 //阶段消息

//阶段时间

**type** GStageInfo **struct** {

Id int32

Stage int32

StageTime int32

}

玩家准备

GAME\_STATUS\_READ

//玩家准备

**type** GSPlayerReady **struct** {

Id int

ChairId int32 //座位ID

IsReady int //0.为准备 1.准备

}

玩家准备返回

MSG\_GAME\_INFO\_READY\_REPLY //玩家准备消息返回410002

**type** GSPlayerReady **struct** {

Id int

ChairId int32 //座位ID

IsReady int //0.为准备 1.准备

}

游戏开始

MSG\_GAME\_INFO\_START = 410000 + iota //游戏开始额外信息

//游戏开始

**type** GGameStartNotify **struct** {

Id int32

Round int //回合

}

发牌阶段

MSG\_GAME\_INFO\_SEND\_NOTIFY //发牌410003

//发牌通知

**type** GGameSendCardNotify **struct** {

Id int32

Rount int

Cid int32 //此人开始叫地主（抢地主）

HandsCards []byte

}

叫地主（叫分制状态）

GAME\_STATUS\_CALL

//阶段时间

**type** GStageInfo **struct** {

Id int32

Stage int32

StageTime int32

}

叫分返回

MSG\_GAME\_INFO\_CALL\_REPLY //410005

//叫地主应答

**type** GCallMsgReply **struct** {

Id int32 //协议号

Cid int32 //哪个玩家叫的分

Coins int32 //叫几分

End int32 //叫分是否结束

}

抢地主状态

GAME\_STATUS\_GETMSG //抢地主阶段

//抢地主

**type** GGetMsg **struct** {

Id int32

GetMsg int32 //是否抢（叫）地主 1，叫地主 2，抢地主 3，不叫 4，不抢

}

抢地主应答

MSG\_GAME\_INFO\_GETMSG\_REPLY //抢地主应答 410020

//抢地主应答

**type** GGetMsgReply **struct** {

Id int32

Cid int32 //轮到哪一个玩家叫分

CallOrGet int32 //接下来是抢地主 还是 叫地主 1,抢地主 2，叫地主

End int32 //抢地主是否结束 1,未结束 2，结束

}

发送定庄通知

MSG\_GAME\_INFO\_BANKER\_NOTIFY //定庄410006

//定庄通知

**type** GBankerNotify **struct** {

Id int32

Banker int32

DiPai []byte

Double int32

}

游戏开始阶段

GAME\_STATUS\_PLAY // 游戏操作阶段

//阶段时间

**type** GStageInfo **struct** {

Id int32

Stage int32

StageTime int32

}

出牌阶段

MSG\_GAME\_INFO\_OUTCARD //出牌410007

//出牌

**type** GGameOutCard **struct** {

Id int32 //协议号

Cards []int32 //牌

}

出牌响应

MSG\_GAME\_INFO\_OUTCARD\_REPLY //410008

//出牌应答

**type** GGameOutCardReply **struct** {

Id int32 //协议号

Cid int32 //谁出的牌

Type int32 //出牌类型

Cards []byte //牌

Max byte //最大牌

Double int32 //倍数,炸弹翻倍

}

过牌

MSG\_GAME\_INFO\_PASS //过410009

过牌应答

MSG\_GAME\_INFO\_PASS\_REPLY //4100010

//过应答

**type** GGamePassReply **struct** {

Id int32 //协议号

Cid int32

}

断线重连

MSG\_GAME\_RECONNECT = 400010 //400010断线重连

断线重连应答

MSG\_GAME\_RECONNECT\_REPLY = 400011 //断线重连应答

//短线重连应答

**type** GReConnectReply **struct** {

Id int32

Result int32

Err string

}

托管

MSG\_GAME\_INFO\_TUOGUAN //410012托管

**type** GTuoGuan **struct** {

Id int32 //协议号

Ctl int32 //1托管，2取消托管

}

托管应答

MSG\_GAME\_INFO\_TUOGUAN\_REPLY //410013

**type** GTuoGuanReply **struct** {

Id int32 //协议号

Ctl int32

Result int32 //结果

Err string

Cid int32 //谁托管

}

断线

MSG\_GAME\_DISCONNECT = 400013 //400013断线

断线应答

MSG\_GAME\_ONLINE\_NOTIFY = 400012 //400012离线和在线通知

MSG\_GAME\_LEAVE\_REPLY = 400008 //游戏离开应答

//游戏离开应答

**type** GLeaveReply **struct** {

Id int32

Result int32 //0成功，其他失败

Cid int32

Uid int64

Err string

Token string

Robot bool // 是否机器人

}

//上线和掉线通知

**type** GOnLineNotify **struct** {

Id int32

Cid int32

State int32 //1上线，2掉线

}