百人牛牛

名词解释(或游戏文档)

暂无

协议文件名

cows_protocol.proto
cows_logic.proto
cows_def.proto
msg_type_def.proto

游戏通用结构

盘口ID

0 东
1 南
2 西
3 北

游戏状态

```
enum class game_state
{
        game_state_unknown,     //未开始
        game_state_prepare,     //准备,处理上庄
        game_state_bet,         //下注
        game_state_deal,        //发牌
        game_state_result,      //结果
};
```

房间信息

msg_room_info

参数名	类型	描述
roomid	int32	房间Id

玩家信息

参数名	类型	描述
player_id	int32	玩家ID
player_name	string	玩家昵称
head_frame	int32	头像框
head_custom	string	玩家头像
player_gold	int64	玩家金币
player_sex	int32	玩家性别
vip_level	int32	Vip等级

庄家信息

参数名	类型	描述
player_info	msg_player_info	玩家信息
max_bet_gold	int64	最大下注金额
can_snatch	bool	是否可以抢庄
snatch_gold	string	抢庄花费多少金币
snatch_player_id	int64	抢庄玩家的ID

下注信息

参数名	类型	描述
self_bet_golds	int64数组	自己在每个盘口的下注金额
total_bet_golds	int64数组	每个盘口的下注总金额

牌信息

参数名	类型	描述
pokers	int32数组	五张牌
cards_type	int32	牌型
cards_value	int32	牌点数
match_indexs	int32数组	组成牛的三张牌

结果信息

参数名	类型	描述
banker_win_gold	int64	庄家输赢
other_win_golds	int64数组	四个盘口的闲家输赢
self_win_gold	int64	自己输赢
self_gold	int64	自己金币
self_is_bet	bool	自己是否下注

房间信息

参数名	类型	描述
roomid	int32	房间ID
scene_state	int32	房间状态
count_down	int32	当前状态倒计时
banker_info	msg_banker_info	庄家信息
bet_info	msg_bet_info	下注信息

参数名	类型	描述
cards_infos	msg_cards_info数组	每个盘口的牌信息,0庄家牌1-4闲家牌
result_info	msg_result_info	开奖结果
main_id	int32	局ID,没用到

牌路信息

参数名	类型	描述
is_win	bool数组	每个盘口输赢

参与开奖玩家

参数名	类型	描述
player_id	int32	玩家ID
name	string	昵称
gold	int64	输赢金币

玩家下注信息,通知给GM

参数名	类型	描述
player_id	int32	玩家ID
player_name	string	昵称
bet_gold	int32数组	下注列表
player_gold	int64	携带金币

协议列表

进入房间

说明:现有架构这条协议是不会用到的,进入房间走的是logic的enter_game，进入游戏同时进入房间.
前端请求协议:packetc2l_enter_table

参数名	类型	描述

参数名	类型	描述
packet_id	e_server_msg_type	协议Id
roomid	int32	房间ID

后端返回协议:packetl2c_enter_table_result

参数名	类型	描述
packet_id	e_server_msg_type	协议Id
result	msg_type_def.e_msg_result_def	结果状态

前端逻辑判断

有进入金币限制的游戏,判断玩家金币是否满足,百人游戏,通常不限制

错误码

```
enum e_msg_result_def
{
  e_rmt_unknow = 0; //未知错误
  e_rmt_success = 1; //成功
  e_rmt_fail = 2; //失败
  e_rmt_room_full = 12;          // 房间已满
}
```

相关配置

表名	字段名	描述	备注
CowRoomConfig	PlayerMaxCount	房间最大人数	包括机器人

获取房间信息

说明:获取房间列表

前端请求协议:packetc2l_get_room_info

参数名	类型	描述
packet_id	e_server_msg_type	协议Id

后端返回协议:packetl2c_get_room_info_result

参数名	类型	描述
packet_id	e_server_msg_type	协议Id
room_list	msg_room_info	房间列表

检测游戏状态

说明:重连之后判断是否在桌子中,如果还在桌子中,需要走断线重连流程
前端请求协议:packetc2l_check_state

参数名	类型	描述
packet_id	e_server_msg_type	协议Id

后端返回协议:packetl2c_check_state_result

参数名	类型	描述
packet_id	e_server_msg_type	协议Id
is_intable	bool	是否在桌子中

请求场景信息

说明:进入房间和断线重连情况下调用,获取整个场景的数据
前端请求协议:packetc2l_get_scene_info

参数名	类型	描述
packet_id	e_server_msg_type	协议Id

后端返回协议:packetl2c_get_scene_info_result

参数名	类型	描述
packet_id	e_server_msg_type	协议Id
scene_info	msg_scene_info	场景信息

下注

说明:收到开始下注通知后，玩家可以开始下注

前端逻辑判断

0.百人牛牛比较特殊，可下注金额,是携带金额的10%,因为最多赔10倍.
1.自己是庄家不能下注.
2.判断当前金币是否大于下注金额.
3.判断当前盘口限红.限红指的是每个玩家在每个盘口的最大下注金额.
4.本局开始时,身上携带的金币,高于下注限制才能下注.
5.盘口ID在范围内.

前端请求协议:packetc2l_ask_bet_info

参数名	类型	描述
packet_id	e_server_msg_type	协议Id
bet_index	int32	盘口ID
bet_count	int64	下注金额

后端返回协议:packetl2c_bet_info_result

参数名	类型	描述
packet_id	e_server_msg_type	协议Id
result	msg_type_def.e_msg_result_def	结果状态
bet_index	int32	盘口ID
bet_count	int64	下注金额

错误码

```
enum e_msg_result_def
{
  e_rmt_unknow = 0; //未知错误
  e_rmt_success = 1; //成功
  e_rmt_fail = 2; //失败
  e_rmt_gold_not_enough = 10;          // 金币不足
  e_rmt_bet_index_error = 21;                    //押注序号不对
  e_rmt_outof_bet_limit = 22;                    //超过房间押注上限
  e_rmt_no_find_table = 23;                      //没有找到桌子
  e_rmt_banker_not_bet = 45;            //庄家不能押注
  e_rmt_other_betgold_is_full = 58;      //超过限红
  e_rmt_betgold_not_enough = 92;     // 本局开始时拥有的金币，低于下注限制
}
```

相关配置

表名	字段名	描述	备注
CowBaseInfoConfig	BetTime	下注时间	下注阶段持续时间
CowRoomConfig	BetLimit	限红	每个玩家每个盘口最大下注金额
CowRoomConfig	BetCondition	下注限制	本局开始时携带的金币,大于下注限制, 才能下注

续压

说明:重复上一局的下注，若当前已下注，不可续压

前端逻辑判断

1.自己是庄家不能续压.

2.已下注不能续压.

3.本局开始时,身上携带的金币,高于下注限制才能下注.

前端请求协议:packetc2l_ask_continue_bet

参数名	类型	描述
packet_id	e_server_msg_type	协议Id

后端返回协议:packetl2c_continue_bet_result

参数名	类型	描述
packet_id	e_server_msg_type	协议Id
result	msg_type_def.e_msg_result_def	结果状态
bet_golds	int64数组	每个盘口下注的金额

错误码

```
enum e_msg_result_def
{
  e_rmt_unknow = 0; //未知错误
  e_rmt_success = 1; //成功
  e_rmt_fail = 2; //失败
  e_rmt_gold_not_enough = 10;          // 金币不足
  e_rmt_outof_bet_limit = 22;                    //超过房间押注上限
  e_rmt_no_find_table = 23;                      //没有找到桌子
  e_rmt_banker_not_bet = 45;            //庄家不能押注
  e_rmt_other_betgold_is_full = 58;      //超过限红
  e_rmt_can_not_bet_hasbet = 85;//玩家下注了不能续压
  e_rmt_betgold_not_enough = 92; // 本局开始时拥有的金币，低于下注限制
}
```

相关配置

表名	字段名	描述	备注
CowBaseInfoConfig	BetTime	下注时间	下注阶段持续时间
CowRoomConfig	BetCondition	下注限制	本局开始时携带的金币,大于下注限制,才能下注

清空下注

说明:清空下注,房间状态为押注状态下可调用

前端逻辑判断

1.下注阶段才能清空下注.

前端请求协议:packetc2l_ask_clear_bet

参数名	类型	描述
packet_id	e_server_msg_type	协议Id

后端返回协议:packetl2c_clear_bet_result

参数名	类型	描述
packet_id	e_server_msg_type	协议Id
result	msg_type_def.e_msg_result_def	结果状态

错误码

```
enum e_msg_result_def
{
  e_rmt_unknow = 0; //未知错误
  e_rmt_success = 1; //成功
  e_rmt_fail = 2; //失败
  e_rmt_no_can_bet = 20;//押注时间已过
}
```

相关配置

表名	字段名	描述	备注
CowRoomConfig	BetTime	下注时间	下注阶段持续时间

申请上庄

说明:申请加入上庄列表
前端请求协议:packetc2l_ask_apply_banker

参数名	类型	描述
packet_id	e_server_msg_type	协议Id

后端返回协议:packetl2c_apply_banker_result

参数名	类型	描述
packet_id	e_server_msg_type	协议Id
result	msg_type_def.e_msg_result_def	结果状态

前端逻辑判断
1.携带的金币是否高于上庄金额

错误码

```
enum e_msg_result_def
{
  e_rmt_unknow = 0; //未知错误
  e_rmt_success = 1; //成功
  e_rmt_fail = 2; //失败
  e_rmt_gold_not_enough = 10;          //金币低于上庄金额
  e_rmt_banker_is_full = 46;                    //上庄列表已满
  e_rmt_has_in_banker_list = 52;                //已经在上庄列表中
  e_rmt_now_is_banker = 53;                     //现在是庄家
}
```

相关配置

表名	字段名	描述	备注
CowRoomConfig	BankerCondition	上庄金额	

申请离开上庄列表

说明:申请离开上庄列表
前端请求协议:packetc2l_leave_list_banker

参数名	类型	描述
packet_id	e_server_msg_type	协议Id

后端返回协议:packetl2c_leave_list_banker_result

参数名	类型	描述
packet_id	e_server_msg_type	协议Id
result	msg_type_def.e_msg_result_def	结果状态

错误码

```
enum e_msg_result_def
{
  e_rmt_unknow = 0; //未知错误
  e_rmt_success = 1; //成功
  e_rmt_fail = 2; //失败
  e_rmt_now_is_banker = 53;                     //现在是庄家
}
```

申请下庄

说明:申请下庄
前端请求协议:packetc2l_ask_leave_banker

参数名	类型	描述
packet_id	e_server_msg_type	协议Id
force	bool	强制下庄

后端返回协议:packetl2c_leave_banker_result

参数名	类型	描述
packet_id	e_server_msg_type	协议Id
result	msg_type_def.e_msg_result_def	结果状态
cost_ticket	int32	强制消耗

错误码

```
enum e_msg_result_def
{
  e_rmt_unknow = 0; //未知错误
  e_rmt_success = 1; //成功
  e_rmt_fail = 2; //失败
}
```

抢庄

说明:花钱抢庄,若当前庄家不在上庄保护期
前端请求协议:packetc2l_ask_snatch_banker

参数名	类型	描述
packet_id	e_server_msg_type	协议Id

后端返回协议:packetl2c_snatch_banker_result
说明:申请抢庄成功

参数名	类型	描述
packet_id	e_server_msg_type	协议Id

参数名	类型	描述
result	msg_type_def.e_msg_result_def	结果状态

前端逻辑判断

1.携带的金币是否高于上庄金额

错误码

```
enum e_msg_result_def
{
  e_rmt_unknow = 0; //未知错误
  e_rmt_success = 1; //成功
  e_rmt_fail = 2; //失败
  e_rmt_gold_not_enough = 10;          // 金币不足
  e_rmt_now_is_banker = 53;                        //现在是庄家
  e_rmt_now_is_you = 63;                             // 庄家已经是自己了
}
```

相关配置

表名	字段名	描述	备注
CowBaseInfoConfig	PminBankerCount	上庄保护局数	上庄保护期间不能被抢庄
CowRoomConfig	FirstBankerCost	抢庄花费	
CowRoomConfig	AddBankerCost	每次抢庄竞价累计值	本局每次抢庄,价格提升

抢庄成功

后端通知协议:packetl2c_banker_success
说明:通知抢庄成功的玩家，抢到了庄，花了多少金币

参数名	类型	描述
packet_id	e_server_msg_type	协议Id
gold	int32	抢庄消耗的金币

抢庄广播

后端通知协议:packetl2c_bc_snatch_banker
说明:广播通知所有玩家有人抢庄成功

参数名	类型	描述
packet_id	e_server_msg_type	协议Id
snatch_player_id	int32	抢庄玩家ID
snatch_gold	int32	抢庄消耗的金币

请求玩家列表

说明:请求当前房间的玩家列表
前端请求协议:packetc2l_ask_playerlist

参数名	类型	描述
packet_id	e_server_msg_type	协议Id

后端返回协议:packetl2c_playerlist_result

参数名	类型	描述
packet_id	e_server_msg_type	协议Id
player_infos	msg_player_info数组	玩家列表

请求上庄列表

说明:请求当前房间的上庄列表
前端请求协议:packetc2l_ask_bankerlist

参数名	类型	描述
packet_id	e_server_msg_type	协议Id

后端返回协议:packetl2c_bankerlist_result

参数名	类型	描述
packet_id	e_server_msg_type	协议Id

参数名	类型	描述
player_infos	msg_player_info数组	上庄列表

请求牌路

说明:请求当前房间的牌路
前端请求协议:packetc2l_ask_history_info

参数名	类型	描述
packet_id	e_server_msg_type	协议Id

后端返回协议:packetl2c_history_info

参数名	类型	描述
packet_id	e_server_msg_type	协议Id
total_count	int32	总局数
win_counts	int32数组	每个盘口赢次数
lose_counts	int32数组	每个盘口输次数
history_infos	msg_history_info数组	牌路列表

广播准备状态

说明:开始游戏时,广播倒计时，庄家信息
后端返回协议:packetl2c_bc_scene_prepare_into

参数名	类型	描述
packet_id	e_server_msg_type	协议Id
count_down	int32	倒计时
banker_info	msg_banker_info	庄家信息

通知开始下注

说明:后端主动通知前端
后端通知协议:packetl2c_bc_scene_bet_into

参数名	类型	描述
packet_id	e_server_msg_type	协议Id
count_down	int32	下注阶段倒计时

同步下注状态

说明:下注阶段时,每秒同步下注
后端通知协议:packetl2c_bc_scene_bet_into

参数名	类型	描述
packet_id	e_server_msg_type	协议Id
bet_golds	int64数组	每个盘口的下注

广播发牌状态

说明:进入发牌阶段通知
后端通知协议:packetl2c_bc_scene_deal_into

参数名	类型	描述
packet_id	e_server_msg_type	协议Id
count_down	int32	发牌阶段倒计时
cards_infos	msg_cards_info数组	牌列表,0庄家1-4闲家

广播结算状态

说明:进入结算阶段通知
后端通知协议:packetl2c_bc_scene_result_into

参数名	类型	描述

参数名	类型	描述
packet_id	e_server_msg_type	协议Id
result_info	msg_result_info	结算信息
player_list	msg_player_award数组	结算玩家列表

游戏控制

说明:控制客户端+超级账号，可以控制游戏结果
前端逻辑判断
1.当前账号是超级账号，并且有控制模块，才显示控制界面

前端请求协议:packetl2c_debug

参数名	类型	描述
packet_id	e_server_msg_type	协议Id
types	int32数组	每个区域的结果
other_type	int32	控制类型 0精确控制 3普通控制

后端返回协议:packetl2c_debug_result

参数名	类型	描述
packet_id	e_server_msg_type	协议Id
result	msg_type_def.e_msg_result_def	结果状态

错误码

```
enum e_msg_result_def
{
  e_rmt_unknow = 0; //未知错误
  e_rmt_success = 1; //成功
  e_rmt_fail = 2; //失败
}
```

通知场景信息给GM

说明:超级账号才会收到这条协议,由后端主动推送

后端协议:packetl2c_notify_sceneinfo

参数名	类型	描述
packet_id	e_server_msg_type	协议Id
main_id	int32	牌局id
banker_type	int32	庄家类型 1系统小庄2机器人3玩家
banker_name	string	庄家名字
player_betinfos	msg_player_betinfo数组	下注玩家列表
earn_gold	int64	抽水
stock_gold	int64	库存
banker_gold	int64	庄家金币

通知坐庄用户离开

说明:连庄次数超过限制,会给该玩家发出这条协议

后端协议:packetl2c_attention_needLeave

参数名	类型	描述
packet_id	e_server_msg_type	协议Id

协议ID枚举

```
enum e_server_msg_type
{
        //客户端到服务端
    e_mst_start_c2l = 10000;
        e_mst_c2l_get_room_info = 10001;//得到房间
        e_mst_c2l_enter_room = 10002;//进入房间
        e_mst_c2l_leave_room = 10003;//离开房间
        e_mst_c2l_add_bet = 10004;//下注
        e_mst_c2l_repeat_bet = 10005;//重复押注
        e_mst_c2l_clear_bet = 10006;//押注清零
        e_mst_c2l_get_room_scene_info = 10007;//获得桌子内信息
        e_mst_c2l_check_state = 10008;//检测游戏状态
        e_mst_c2l_gm = 10009;//GM
        e_mst_c2l_ask_for_banker = 10010;//上庄
        e_mst_c2l_leave_banker = 10011;//下庄
        e_mst_c2l_ask_first_for_banker = 10012;//抢庄
        e_mst_c2l_ask_player_list = 10013;//请求玩家列表
        e_mst_c2l_ask_banker_list = 10014;//请求上庄列表
        e_mst_c2l_ask_history_list = 10015;//请求牌路

    e_mst_c2l_control_change_result = 10016;//控制客户端提示服务器是否放分

        // 服务端到客户端--------------------------------------------------------------------------

        e_mst_start_l2c = 15000;//客户端到服务端
    e_mst_start_c2l = 10000;
        e_mst_c2l_get_room_info = 10001;//获取房间信息
        e_mst_c2l_enter_table = 10002;//进入房间
        e_mst_c2l_leave_table = 10003;//离开房间
        e_mst_c2l_check_state = 10004;    //检测状态

        e_mst_c2l_get_scene_info = 10010;        //获取场景信息
        e_mst_c2l_ask_bet_info = 10011;          //请求下注
        e_mst_c2l_ask_apply_banker = 10012;              //申请上庄
        e_mst_c2l_ask_leave_banker = 10013;              //请求下庄
        e_mst_c2l_ask_bankerlist = 10014;                //请求庄家列表
        e_mst_c2l_ask_history_info = 10015;      //请求历史结果
        e_mst_c2l_ask_continue_bet = 10016;              //请求续压
        e_mst_c2l_ask_clear_bet = 10017;                 //请求取消下注
        e_mst_c2l_ask_playerlist = 10018;                //请求玩家列表
        e_mst_c2l_ask_snatch_banker = 10019;     //请求抢庄

        e_mst_c2l_leave_list_banker = 10020;//下庄列表

        e_mst_c2l_debug = 10100;                                    //控制
        // 服务端到客户端--------------------------------------------------------------------------

        e_mst_start_l2c = 15000;
        e_mst_l2c_get_room_info_result = 15001;//返回房间信息
        e_mst_l2c_enter_table_result = 15002;//进入房间返回
        e_mst_l2c_leave_table_result = 15003;//离开房间返回
        e_mst_l2c_check_state_result = 15004;//返回状态

        e_mst_l2c_get_scene_info_result = 15010;         //场景信息返回
```

```
        e_mst_l2c_bet_info_result = 15011;                      //下注结果
        e_mst_l2c_apply_banker_result = 15012;          //申请上庄结果
        e_mst_l2c_leave_banker_result = 15013;          //离开庄家
        e_mst_l2c_bankerlist_result = 15014;            //庄家列表
        e_mst_l2c_history_info = 15015;                         //历史牌型
        e_mst_l2c_banker_info = 15016;                          //上庄下庄提示
        e_mst_l2c_continue_bet_result = 15017;          //请求续压结果
        e_mst_l2c_clear_bet_result = 15018;                     //取消下注结果
        e_mst_l2c_playerlist_result = 15019;            //玩家列表结果
        e_mst_l2c_snatch_banker_result = 15020;         //抢庄结果
        e_mst_l2c_banker_success = 15021;       //抢庄成功

        e_mst_l2c_bc_scene_prepare_into = 15050;        //场景准备状态
        e_mst_l2c_bc_scene_bet_into = 15051;            //场景下注状态
        e_mst_l2c_bc_sync_scene_bet_into = 15052;       //场景下注同步
        e_mst_l2c_bc_scene_deal_into = 15053;           //场景发牌状态
        e_mst_l2c_bc_scene_result_into = 15054;         //场景结果状态
        e_mst_l2c_bc_snatch_banker = 15055;                     //抢庄结果

        e_mst_l2c_notify_sceneinfo = 15101;                     //通知场景信息
        e_mst_l2c_debug_result = 15102;                 //控制返回

        e_mst_l2c_leave_list_banker = 15103;//下庄列表

    e_mst_attention_needLeave = 15104;

        e_mst_clend_index = 20000;
}
```