**星际移民PVE战场接口**

1. **、PVE战场交互概述**

进入PVE行星战场有3种方式，1、行星要塞处于正常可攻击状态，通过客户端向游戏服务器（GAME SERVER）发送攻击请求，获取到战斗房间以后登录到房间服务器（ROOM SERVER）进行战斗过程；2、行星要塞已经处于被攻击状态，即已经有其他玩家正在对该行星要塞进行攻击，此时只需要从WEB查询获取战斗房间的详细信息，即可直接连接房间服务器进行联合攻击；3、客户端上次战斗意外退出，使用保存的信息计算房间是否超时，如果没有超时则使用缓存信息的房间服务器信息直接连接房间服务器，进行要塞攻击。本文描述第一种情况，第2、3种情况比较简单不详细描述。

当游戏处在游戏内行星地图版面时，可以点击单个小行星来查看该小行星的当前状态，通过WEB请求获取。如果该小行星当前是可以被攻击的状态，则客户端界面会出现“攻击”字样，此时通过点击“攻击”按钮最后确认是否可以攻击；如果可以，则跳转到战车选择版面。选好战车后点击“进入战场”进入到PVE战场交互部分，包含2个部分：

1. 客户端(CLIENT)与游戏服务器(GAME SERVER)交互

客户端首先向游戏服务器(GAME SERVER)发起TCP连接，成功后发送登录数据包，然后等待游戏服务器返回登录结果；如果登录失败，返回登录结果包含错误信息。如果登录成功，则向服务器发送攻击请求，根据服务器返回的结果进行判断，出错或者包含进行战斗的房间服务器地址、端口以及临时校验信息；过程参考图1所示。目前游戏服务器TCP通信端口暂定为30000，客户端应该使用从查询服务器列表时获取到的各种信息，包游戏服务器、括聊天服务器等。



1. 客户端(CLIENT)与房间服务器(ROOM SERVER)交互

客户端从游戏服务器处获取房间服务器详细信息后，向房间服务器发起TCP连接（通信详细信息包含在从游戏服务器请求结果中），成功后发送登录数据，然后等待房间服务器返回登录结果；如果登录失败，返回登录结果包含错误信息。如果成功，则开始正常的命令互动，直到房间服务器通知游戏结束。

* 在对上述2个服务器进行登录时，当连接成功发送了登录数据包以后，在收到服务器响应登录请求之前，客户端不能向服务器发送任何数据，否则会被服务器无提示断开连接。
* PVE战场用到的命令、错误代码、数据结构以及所有枚举定义都包含在command-def.hpp中；部分数据使用Google protocol buffer进行描述和序列化，参考struct-def.proto中结构定义，该文件可以使用对应的语言工具转化为本地语言类从而实现跨语言的数据持久化。Google protocol buffer结构定义参考具体<http://code.google.com/p/protobuf/>网站的详细说明。
* 如无特殊说明，交互所有数据使用little-endian数据。
* 数据收发必须加入残包、粘包处理机制。
* 本文未列出的数据结构请参考command-def.hpp中定义，命令对应的数据结构为：假设某命令为XXXX\_YYY，则发送该命令时使用的数据结构为HEADER\_XXXX\_YYY。

1. **、PVE战场通信接口详细设计**
   1. 客户端(CLIENT)与游戏服务器(GAME SERVER)

本部分涉及到的命令一共是4个，参考command-def.hpp中的枚举COMMAND\_GAME2CLIENT定义，如下：

GAME2CLIENT\_LOGIN 登录游戏服务器

GAME2CLIENT\_LOGIN\_RESULT 登录游戏服务器结果

GAME2CLIENT\_REQUEST\_ROOM 请求（房间）攻击（行星）

GAME2CLIENT\_REQUEST\_ROOM\_RESULT 请求攻击结果

请求登录服务器时使用如下数据结构：

struct HEADER\_GAME2CLIENT\_LOGIN {

HEADER header;

id\_type whoami;

uint32\_t session\_key\_len;

char session\_key[0];

};

其中HEADER结构是所有通信的前导结构，定义如下：

struct HEADER {

uint32\_t length; //数据包长度，包含sizeof(HEADER)

uint16\_t command; //命令字

uint16\_t result; //命令执行结果，参考ERROR\_CODE\_DEFINE

};

参考定义中的注释，后续不再描述该结构。

登录结构中whoami是客户端玩家编号，为无符号64位整数，玩家从客户端登录时可以从WEB请求结果中获取。Session\_key\_len指定后续变量session\_key的长度，目前版本为恒定长度40个字节。

登录结果结构为HEADER结构，如果result字段为0即RESULT\_SUCCESS则表明登录成功，否则登录失败，参考错误定义枚举获取详细错误信息。

登录成功后客户端构造攻击请求数据包（请求房间），定义如下：

struct HEADER\_GAME2CLIENT\_REQUEST\_ROOM {

HEADER header;

id\_type whoami; //攻击者id

id\_type chariot; //攻击者tank\_id

id\_type who; //防守者id

id\_type where; //要塞id

};

其中whoami同登录时，chariot为将用来参加战斗的战车编号，该编号在进入战场前选择战车时获取；who为将要被攻击的行星要塞所有者玩家编号，如果该行星要塞还没有被人占领(地图中的白色星球)则该字段应该为0；where是将要被攻击的行星要塞的编号，该编号在行星地图浏览时可以从WEB请求中获取。

请求攻击结果返回如下：

struct HEADER\_GAME2CLIENT\_REQUEST\_ROOM\_RESULT {

HEADER header;

uint32\_t gid;

ROOM\_CONTEXT room[0];//if success

};

Gid为后续登录房间时将使用到的信息，为玩家在战场中的组号，用以进行敌我识别；

Room则详细描述用来连接房间服务器的信息，包括IP地址、端口、类型、编号以及临时校验口令，其中编号将在连接房间服务器时向大厅服务器指定要登录的房间编号。

* 1. 客户端(CLIENT)与房间服务器(ROOM SERVER)

PVE战斗主要的交互逻辑是客户端与房间服务器的交互，在成功登录房间服务器以后不能再次发送登录命令，否则会被服务器无提示断开。

详细命令定义参考枚举COMMAND\_ROOM2CLIENT，定义如下：

enum COMMAND\_ROOM2CLIENT {

ROOM2CLIENT\_LOGIN = 0, //玩家者登陆

ROOM2CLIENT\_PING, //延迟计算

ROOM2CLIENT\_MOVING, //攻击者对象移动

ROOM2CLIENT\_FIRE, //开火

ROOM2CLIENT\_ATTACKED, //报告伤害对象情况

ROOM2CLIENT\_FIX, //防守者调整对象恢复速度

ROOM2CLIENT\_AIRFORCE\_HELP, //防守者呼叫空中支援

ROOM2CLIENT\_SURRENDER, //攻击者投降、撤离

ROOM2CLIENT\_REQUEST\_CONTROL, //攻击者请求控制防守方攻击对象

ROOM2CLIENT\_RELEASE\_CONTROL, //攻击者主动释放已控制的对象

ROOM2CLIENT\_VOTE, //要塞归属投票

ROOM2CLIENT\_BOARDCAST\_MESSAGE, //玩家自定义消息

ROOM2CLIENT\_OBJECT\_REQUEST\_RELIVE, //请求复活对象，当前仅对战车有效

ROOM2CLIENT\_REQUEST\_BUFFER, //请求获得增益效果

//以下是room服务器发送给客户端的命令

ROOM2CLIENT\_LOGIN\_RESULT = 1000, //

ROOM2CLIENT\_BOARDCAST\_STATUS, //广播受到伤害对象当前

ROOM2CLIENT\_FINISH, //游戏结束

ROOM2CLIENT\_FINISH\_TIMEOUT, //游戏超时结束

ROOM2CLIENT\_MOVING\_HISTORY, //可移动对象移动历史

ROOM2CLIENT\_REQUEST\_CONTROL\_RESULT, //请求控制结果

ROOM2CLIENT\_TIMEOUT\_RELEASE\_CONTROL, //被控制对象超时被释放

ROOM2CLIENT\_PLAYER\_ENTER, //新玩家进入房间

ROOM2CLIENT\_VOTE\_STARTUP, //开始投票

ROOM2CLIENT\_VOTE\_RESULT, //投票结果

ROOM2CLIENT\_SHUTDOWN, //房间即将关闭

ROOM2CLIENT\_OBJECT\_DYING,

ROOM2CLIENT\_OBJECT\_REQUEST\_RELIVE\_RESULT, //请求复活结果

ROOM2CLIENT\_FORTBUIL编号NG\_CHANGING\_G编号,

//

ROOM2CLIENT\_BUFFER\_GENERATED, //增益产生

ROOM2CLIENT\_REQUEST\_BUFFER\_RESULT, //增益获取结果

ROOM2CLIENT\_BUFFER\_EATING,

ROOM2CLIENT\_NPC\_CHARIOT\_ENTER, //NPC战车加入战场

ROOM2CLIENT\_OBTAIN\_HONOR, //击毁建筑（炮台）获得荣誉

ROOM2CLIENT\_UPDATE\_OBJECT, //更新对象属性

};

1. ROOM2CLIENT\_LOGIN与ROOM2CLIENT\_LOGIN\_RESULT、ROOM2CLIENT\_PLAYER\_ENTER命令

该命令在客户端向房间服务器成功发起连接后发送，命令结构定义如下：

struct HEADER\_ROOM2CLIENT\_LOGIN {

HEADER header;

id\_type whoami; //玩家编号

id\_type chariot\_id; //战车编号

uint32\_t gid; //分组id = ENUM\_PLAYER\_GROUP

uint32\_t room\_index; //房间号

uint32\_t passport; //房间临时通信证

uint32\_t version; //系统版本号

};

Whoami是玩家编号，chariot\_id是将用来进行战斗的战车编号，gid、roomv\_index、passport是从GAME SERVER获取到的信息或者从WEB请求中获取到的信息，version是一个常量，定义在command-def.hpp中，一般不会修改这个定义。

登录结果通过如下数据结构进行返回,定义服下：

struct HEADER\_ROOM2CLIENT\_LOGIN\_RESULT {

HEADER header;

uint32\_t room\_startup; //当前时间

uint32\_t room\_will\_shutdown\_at; //房间预计超时时间

char data[0]; //如果登陆成功，此处数据为 USER\_DATA 的序列化之后的二进制数据

};

如果成功，HEADER中的result为RESULT\_SUCCESS并且data携带当前战场中所有的玩家详细数据，失败则data不携带任何信息即data无数据。

其中vroom\_startup为服务器当前时间，配合room\_will\_shutdown\_at可以计算出该房间超时关闭的剩余时间，目前设定PVE战场最大为3分钟（或者5分钟）；超时结束的战斗会判定攻击失败，所有参与攻击的战车都会被损毁。

Data数据为google protocol buffer 结构 USER\_DATA 的序列化数据，该结构是一个循环嵌套结构，其中包含了所有玩家，玩家基本信息，玩家战车，战车基本信息，战车所有挂件信息以及玩家要塞基本信息以及要塞所包含的要塞建筑详细信息，详细信息参考struct-def.proto

如果战场中已经有别的玩家存在时，先前存在的玩家会收到ROOM2CLIENT\_PLAYER\_ENTER命令，数据结构同HEADER\_ROOM2CLIENT\_LOGIN\_RESULT，只是该结构的data中仅包含新进入玩家的详细数据。此时客户端应该销毁所有的NPC飞行器，服务器端也会做相应的处理。



1. ROOM2CLIENT\_PING命令（废弃）
2. ROOM2CLIENT\_MOVING和ROOM2CLIENT\_MOVING\_HISTORY命令

该命令涉及的定义如下：

struct POINT {

float x;

float y;

};

struct MOVING {

id\_type whoami; //对象编号，当前应该为chariot编号

POINT from; //出发点坐标

float angle; //角度

float speed; //速度

POINT to; //目标点坐标

uint32\_t duration; //客户端发送移动可以不填写，服务器返回的结构此处表示距离此次移动已过去的时间ms

};

struct HEADER\_ROOM2CLIENT\_MOVING {

HEADER header;

MOVING moving; //详细移动信息

char data[0]; //客户端携带信息扩展保留

};

参考定义中的注释，不细述；其中data为客户端数据扩展，即该部分数据类型、长度由客户端自己定义，服务器不解析该字段，但是进行转发的时候会携带该信息，注意HEADER中length要包含data的实际长度，否则会发生错帧错误。该消息会进行广播但排除发送者本身，并且消息内容会被更新到玩家移动历史记录，该记录会在新玩家登录时进行发送。

移动历史命令数据结构定义如下：

struct HEADER\_ROOM2CLIENT\_MOVING\_HISTORY {

HEADER header;

uint32\_t nb\_history;

MOVING moving[0];

};

在登录成功时房间服务器会发送除登录者之外的所有玩家最后一次移动消息，新玩家可以更具这个信息对画面进行同步。

1. ROOM2CLIENT\_FIRE命令

开火命令数据结构定义如下：

struct HEADER\_ROOM2CLIENT\_FIRE {

HEADER header;

id\_type identifier\_firing; //正在射击的chariot对象id

char data[0]; //客户端携带信息扩展保留

};

其中identifierv\_firing为战车编号，data数据由客户端自行扩展，意义同移动命令中的data定义。

服务器端会对该命令进行校验，开火对象必须是玩家的战车或者是被玩家控制的对象，该命令会被广播给包括发送者的所有人。

* 1. ROOM2CLIENT\_ATTACKED和ROOM2CLIENT\_BOARDCAST\_STATUS命令

该命令使用的数据结构定义如下：

enum ENUM\_OBJECT\_TYPE {

OBJECT\_TYPE\_CHARIOT = 0,//战车

OBJECT\_TYPE\_FORTBUILDING,//建筑

OBJECT\_TYPE\_TANKPART,//挂件

OBJECT\_TYPE\_NB,

};

struct ATTACKED\_OBJECT {

id\_type id\_take\_attack; //受到伤害的目标id

uint32\_t type; //0 chariot 1 building

uint32\_t attack\_type; //攻击类型

POINT pt; //受到伤害的目标当前位置

};

struct HEADER\_ROOM2CLIENT\_ATTACKED {

HEADER header;

POINT explode\_pos; //爆炸点

id\_type attacker\_uid;

uint32\_t attacker\_gid; //攻击者gid

float current\_min\_attack; //# 当前最小基础攻击

float current\_max\_attack; //# 当前最小基础攻击

float current\_attack\_area; //# 当前爆炸范围

uint32\_t count; //此次爆炸受影响的对象个数

ATTACKED\_OBJECT attack[0]; //详细列举具体对象信息

};

参考定义中的注释，不再细述。结构中的攻击数据包含在登录成功的返回PB数据中，按照具体的攻击对象抽取。服务器通过计算后，使用ROOM2CLIENT\_BOARDCAST\_STATUS命令进行响应，但响应不是必须的，如某个对象已经在前次伤害中被摧毁，这个是由于设计模式导致的。

该数据的产生可以从前次接收到的开火命令中提取，并计算开火所造成的自己的战车以及自己所有控制对象的伤害原始数据，发送到服务器进行计算。注意，通知服务器对象遭受攻击时，仅计算被自己控制的对象包括自己的战车以及NPC飞行器等，忽略未控制对象以及可能被其他玩家控制的对象。

具体定义如下：

struct HEALTH\_BOARDCAST {

uint32\_t type; //HBC\_TYPE\_CHARIOT=0 chariot HBC\_TYPE\_FORTBUILDING=1 building

id\_type id; //

float current\_endurance;

float current\_shield;

};

struct HEADER\_ROOM2CLIENT\_BOARDCAST\_STATUS {

HEADER header;

uint32\_t count; //

HEALTH\_BOARDCAST health[0];

};

通过上述定义，服务器按照实际返回本次攻击收到影响的对象目前的耐久、护盾值。结果是对所有玩家进行广播，客户端解析数据后对在场的收影响对象进行数字的修正并反映到界面上。

1. ROOM2CLIENT\_FIX（未使用）
2. ROOM2CLIENT\_AIRFORCE\_HELP（未使用）
3. ROOM2CLIENT\_SURRENDER

使用的数据结构定义如下：

struct HEADER\_ROOM2CLIENT\_SURRENDER {

HEADER header;

id\_type uid;

};

Uid为玩家编号，同前述whoami。发送该命令后，客户端可以立即断开到房间服务器的连接，同时该命令会被广播给除发送者之外的所有玩家。如果是最后一名玩家头像则房间会执行清理过程，如果仍然有玩家存在则将来玩家还可以返回到该房间进行的战斗中。

1. ROOM2CLIENT\_REQUEST\_CONTROL和ROOM2CLIENT\_RELEASE\_CONTROL命令

使用的数据结构定义如下：

struct HEADER\_ROOM2CLIENT\_REQUEST\_CONTROL {

HEADER header;

id\_type target; //炮台id

};

其中target为要塞建筑（炮塔）编号。

目前对建筑的控制按照谁认为自己离炮台最近的原则来控制，如果建筑已经被控制，则当前被控制玩家将会收到HEADER\_ROOM2CLIENT\_RELEASE\_CONTROL命令，表示该建筑被其他人抢夺控制权。目前请求控制建筑恒成功，也返回ROOM2CLIENT\_REQUEST\_CONTROL\_RESULT命令，客户端可以在收到命令后把对象加入控制列表。

1. ROOM2CLIENT\_VOTE和ROOM2CLIENT\_VOTE\_STARTUP命令

使用的数据结构如下：

enum ENUM\_VOTE\_TYPE {

VOTE\_TYPE\_GIVEUP=0,

VOTE\_TYPE\_NEED,

VOTE\_TYPE\_BUY,

VOTE\_TYPE\_NB,

};

struct VOTE\_CONTEXT {

id\_type uid;

uint32\_t type; //VOTE\_TYPE\_BUY/VOTE\_TYPE\_NEED/VOTE\_TYPE\_GIVEUP

uint32\_t value;

};

struct HEADER\_ROOM2CLIENT\_VOTE {

HEADER header;

VOTE\_CONTEXT vote;

};

在多玩家对同一个行星要塞进行攻击，并且战斗胜利结束时（行星要塞的标志建筑要塞中心被击毁）在场的玩家超过至少一个，房间服务器会发送ROOM2CLIENT\_VOTE\_STARTUP命令到每个玩家，测试玩家在固定时间（10～15秒依赖服务器配置值）内发送ROOM2CLIENT\_VOTE到服务器以决定该要塞的归属。数量代表的值是暗物质，客户端需要校验输入越界。Type表示放弃、需求、购买，超时未收到投票视为放弃。

所有投票结束后（无有效投票使用放弃票替代）房间服务器将把所有投票信息进行广播，客户端可以用以展示。

投票期间，客户端只能向房间服务器发送投票命令，其他命令将被忽略。

1. ROOM2CLIENT\_BOARDCAST\_MESSAGE

该命令为客户端自定义广播数据扩展，即某客户端想广播数据给其他客户端时，可以自行定义命令结构，通过服务器端进行广播，服务器端不对数据进行解释。

1. ROOM2CLIENT\_OBJECT\_REQUEST\_RELIVE

玩家战车被击毁后，客户端询问是否使用暗能水晶进行现场复活，如果玩家选择复活则发送该命令到房间服务器，服务器确认可以复活后对该数据包进行广播，其他客户端需要呈现该对象复活。数据结构定义如下：

struct HEADER\_ROOM2CLIENT\_OBJECT\_DYING {

HEADER header;

// id\_type uid;

uint32\_t type; //HBC\_TYPE\_CHARIOT=0 chariot HBC\_TYPE\_FORTBUILDING=1 building

id\_type identify;

uint32\_t timeout;

};

Type配合identify说明欲复活对象，目前只有战车复活。

1. ROOM2CLIENT\_REQUEST\_BUFFER、ROOM2CLIENT\_BUFFER\_GENERATED和ROOM2CLIENT\_REQUEST\_BUFFER\_RESULT命令

房间服务器按照游戏策略会在房间初始时、建筑被击毁时产生增益效果，产生时会广播ROOM2CLIENT\_BUFFER\_GENERATED命令，客户端应该记录该BUFFER，在适当的时候（如距离靠近）发送获取buffer的请求；请求成功服务器会删除该增益并广播该获取增益请求，其他客户端此时应该从本地保存增益列表中删除，由于不同不，获取增益请求有可能失败。

1. ROOM2CLIENT\_FINISH、ROOM2CLIENT\_FINISH\_TIMEOUT和ROOM2CLIENT\_SHUTDOWN命令

战斗结束时，如果分出胜负即要塞中心被击毁，则会发送ROOM2CLIENT\_FINISH附带一个结果表明胜利或者失败；如果是超时未分出胜负（要塞中心未被击毁）则服务器广播ROOM2CLIENT\_FINISH\_TIMEOUT命令，客户端应该自行摧毁所有战车并离开战场。无论任何情况当客户端收到ROOM2CLIENT\_SHUTDOWN时表明服务器即将关闭，此时可以立即断开到房间服务器的连接，该命令也附加一个RESOURCE\_OBTAIN\_IN\_BATTLE结构，表明该玩家在此次战斗中的所有收益；如果是胜利也包括胜利的战斗奖励，客户端可以根据这个命令在战斗胜利时展示战斗结果。

1. ROOM2CLIENT\_NPC\_CHARIOT\_ENTER命令

房间服务器根据策略会定时产生NPC飞行器，产生时通过这个命令广播给所有玩家，玩家应该按照统一的原则对NPC飞行器进行运动、攻击以及销毁。各种逻辑本地执行，不同步给其他玩家。

1. ROOM2CLIENT\_OBTAIN\_HONOR命令

玩家击毁一个要塞建筑后，会收到该命令，表明通过击毁获得的荣誉/声望数值，由客户但呈现获取效果，数据结构定义如下：

struct HEADER\_ROOM2CLIENT\_OBTAIN\_HONOR {

HEADER header;

id\_type who; //who obtain honor

id\_type generate\_by; //generate by fortbuilding only

int honor; //how much of honor

};

其中generate\_by是产生要塞的编号，效果可以产生在该建筑的位置，honor是具体荣誉/声望数值。

1. ROOM2CLIENT\_OBJECT\_DYING未使用
2. ROOM2CLIENT\_FORTBUILIDNG\_CHANGING\_GID未使用
3. ROOM2CLIENT\_BUFFER\_EATING未使用
4. ROOM2CLIENT\_UPDATE\_OBJECT命令

数据结构定义如下：

enum PARAMETER\_TYPE {PT\_STRING = 0, PT\_FLOAT, PT\_INT32, PT\_UINT32, PT\_INT64, PT\_UINT64};

struct OBJECT\_UPDATE\_CONTEXT {

uint32\_t offset;

uint32\_t value\_type; //ENUM\_VALUE\_TYPE

uint32\_t value;

};

struct HEADER\_ROOM2CLIENT\_UPDATE\_OBJECT {

HEADER header;

uint32\_t type; //HBC\_TYPE\_CHARIOT=0 chariot HBC\_TYPE\_FORTBUILDING=1 building

id\_type who; //which will be updated

uint32\_t count;

OBJECT\_UPDATE\_CONTEXT contexts[0];

};

目前count恒为0，且无后续数据，客户端收到该命令后，应该去掉要塞中心的攻击力，耐久度从别的渠道广播。该命令只有在所有要塞中攻击型建筑被击毁后从服务器发送。

1. ROOM2CLIENT\_NPC\_CHARIOT\_ENTER命令

数据结构定义如下：

struct HEADER\_ROOM2CLIENT\_NPC\_CHARIOT\_ENTER {

HEADER header;

id\_type generated\_by; //0基地产生、非0是从该对象当前点产生，仅在目标类型为CHARIOT\_TYPE\_NPC\_FLYING\_BOMB有意义；目标类型为CHARIOT\_TYPE\_NPC\_MOTHERSHIP时，始终在基地产生

char player1\_pb[0]; //PLAYER1结构pb数据，仅有chariots/gid有意义，type只能为CHARIOT\_TYPE\_NPC\_MOTHERSHIP/CHARIOT\_TYPE\_NPC\_FLYING\_BOMB

};

参考定义注释，其中player\_pb为PB序列化数据。