==================================================================

CONSTANTS

==================================================================

100-level codes are reserved for client messages.

200-level codes are reserved for server messages.

300-level codes are reserved for unpaired messages.

Blue - Lobby Protocols

Red - Race Royale and Demolition Derby Protocol

Black - Lobby and Game Protocols

**Request Constants**

CMSG\_AUTH 101

CMSG\_DISCONNECT 102

CMSG\_REGISTER 103

CMSG\_FORGOT\_PASSWORD 104

CMSG\_CREATE\_CHARACTER 105

CMSG\_CHAT 106

CMSG\_MOVE 107

CMSG\_POWER\_UP 108

CMSG\_POWER\_PICKUP 109

CMSG\_HEALTH 110

CMSG\_ENTER\_LOBBY 111

CMSG\_ENTER\_GAME\_LOBBY 112

CMSG\_ENTER\_GAME\_NAME 113

CMSG\_CREATE\_LOBBY 114

CMSG\_PRIVATE\_CHAT 115

CMSG\_INVITE 116

CMSG\_CAR\_CHOICE 117

CMSG\_CAR\_PAINT 118

CMSG\_CAR\_TIRES 119

CMSG\_GARAGE\_PURCHASE 121

CMSG\_RESULTS 122

CMSG\_RANKINGS 123

CMSG\_PRIZES 124

CMSG\_COLLISION 125

CMSG\_DEAD 126

CMSG\_READY 127

REQ\_HEARTBEAT 301

**Response Constants**

SMSG\_AUTH 201

SMSG\_DISCONNECT 202

SMSG\_REGISTER 203

SMSG\_FORGOT\_PASSWORD 204

SMSG\_CREATE\_CHARACTER 205

SMSG\_CHAT 206

SMSG\_MOVE 207

SMSG\_POWER\_UP 208

SMSG\_POWER\_PICKUP 209

SMSG\_HEALTH 210

SMSG\_ENTER\_LOBBY 211

SMSG\_ENTER\_GAME\_LOBBY 212

SMSG\_ENTER\_GAME\_NAME 213

SMSG\_CREATE\_LOBBY 214

SMSG\_PRIVATE\_CHAT 215

SMSG\_INVITE 216

SMSG\_CAR\_CHOICE 217

SMSG\_CAR\_PAINT 218

SMSG\_CAR\_TIRES 219

SMSG\_GARAGE\_PURCHASE 221

SMSG\_RESULTS 222

SMSG\_RANKINGS 223

SMSG\_PRIZES 224

SMSG\_COLLISION 225

SMSG\_DEAD 226

SMSG\_READY 227

SMSG\_RENDER\_CHARACTER 310

SMSG\_REMOVE\_CHARACTER 311

==================================================================

Client Side Protocol (Requests)

==================================================================

|  |  |
| --- | --- |
| **Type and usage** | **Format** |
| **RequestLogin**  Client requests to login with a username and password. The server validates this and responds with ResponseAuth. | Short Constants.CMSG\_AUTH  String Username  String Password |
| **RequestLogout**  Client wishes to log out from the game. No more requests are to be sent after this. The server will respond with ResponseDisconnected, and server will update other users with ResponseRemoveUser. | Short Constants.CMSG\_DISCONNECT |
| **RequestRegistration**  Client registers a new account with the server which includes a username and password. The server also keeps the user’s email for recovering account information. The server validates this and responds with ResponseRegistration. | Short Constants.CMSG\_REGISTER  String Username  String Password  String Email |
| **RequestAccountInformation**  Client requests for their account information to be email to them. The server check the Username and Email to see if they match an account and email the user their account Information. | Short Constants.CMSG\_FORGOT\_PASSWORD  String Username  String Email |
| **RequestCharacterCreation**  Client wishes to creates a new character tied to their account which includes the character name and the faction which their character is loyal to. The server will respond with CharacterCreationResponse. | Short Constants.CMSG\_CREATE\_CHARACTER  String characterName  int classType |
| **RequestChat**  The client chats to other clients through the chat function. The client uses a String for the message being sent. | Short Constants.CMSG\_CHAT  String message |
| **RequestMove**  Client issues a change to their location, and  isMovingflag. This is used when a client wishes to move or stop moving. It is followed by creating a number of ResponseMove and Server will update other users with these ResponseMove  ***Note: This is the way we will handle respawning. When a character’s HP falls to 0 they will be sent to the nearest friendly control point, if none are found then they will be sent to base.*** | Short Constants.CMSG\_MOVE  Double x //location vector  Double y  Double z  Double h //facing direction  String keys |
| **RequestHeartbeat**  Client's state has not changed, but enough time has passed that the client would like an update from the server. Each client now is a thread and associated with a specific username, when request comes, server knows which client is sending the heartbeat, so no user id is required. The server will be able to check the client’s queued response and send all of them out to the client socket. | Short Constants.REQ\_HEARTBEAT |
| **RequestPowerUpUse**  When the client presses a button bound to using a powerup and has a specific powerup which is differentiated through powerId. The server responds by animating their attack through ResponsePowerUpUse. | Short Constants.CMSG\_POWER\_UP  int powerId // which attack am i using |
| **RequestPowerUpPickUp**  When the client picks up a power up. The servers adds the power-up to the car object it has in memory. | Short Constants.CMSG\_POWER\_UP\_PICK\_UP  int powerId |
| **RequestChangeHealth**  When the user damages a character, they report who the damage is being dealt to and how much. The server responds by updating all health bars involved through ResponseChangeHealth. | Short Constants.CMSG\_HEALTH  String Username // this is the person receiving the damage  int healthChange // can be positive or negative |
| **RequestEnterLobby**  When the user attempts to join a lobby, they send their username and lobby id to the server, and the server responds to the users already in the lobby through ResponseEnterLobby | Short Constants.CMSG\_ENTER\_LOBBY  String Username  int lobbyId |
| **RequestEnterGameLobby**  When the user attempts to join a game lobby, they send their username and lobby id to the server, and the server responds to the users already in the lobby through ResponseEnterGameLobby | Short Constants.CMSG\_ENTER\_GAME\_LOBBY  String Username  int gameId  int lobbyId |
| **RequestEnterGameName**  When the user attempts to join a custom lobby, they send their username and lobby name to the server, and the server responds with a accepted or failed to enter the game room through ResponseEnterGameName | Short Constants.CMSG\_ENTER\_GAME\_NAME  String Username  String GameLobby |
| **RequestCreateLobby**  When the user creates a lobby, they send the gamemodeid, groupName,and status to the server, and the server responds to all online users with the information to display on the lobby list for that gamemode through ResponseCreateLobby | Short Constants.CMSG\_CREATE\_LOBBY  String groupName  int gamemodeId  int status  // 0 is open  // 1 is closed/invite only |
| **RequestPrivateChat**  The client chats to a single client through the chat function. The client uses a String for the message being sent. The server sends ResponsePrivateChat to that user. | Short Constants.CMSG\_PRIVATE\_CHAT  String username  String message |
| **RequestInvite**  When the client sends a friend request or group invite, they send the username of the person they want to invite, the inviteType, and the groupName if it is a group invite. | Short Constants.CMSG\_INVITE  String username  int InviteType  // 0 is friend  // 1 is group  int groupName |
| **RequestCarChoice**  When the user chooses their car choice, it is sent to the server and the other clients are notified throughResponseCarChoice | Short Constants.CMSG\_CAR\_CHOICE  String Username  Int carType |
| **RequestCarPaint**  When the user chooses the color of their car, it is sent to the server and the other clients are notified through ResponseCarPaint | Short Constants.CMSG\_CAR\_PAINT  String Username  Int color |
| **RequestCarTires**  When the user chooses their car’s tires, it is sent to the server and the other clients are notified through ResponseCarTires | Short Constants.CMSG\_CAR\_TIRES  String Username  Int tireType |
| **RequestGaragePurchase**  When the user attempts to purchase a part for their car, this request is sent to the server to check if the user can make the purchase. A response is given through ResponseGaragePurchase | Short Constants.CMSG\_GARAGE\_PURCHASE  String Username  Int Purchase |
| **RequestResults**  When the client ends the race or dies they request the results to see where the placed in the gamemode. This calls the response for the user and all of the other users who have finished the race with ResponseResults | Short Constants.CMSG\_RESULTS  int gameId |
| **RequestRankings**  The client requests the status for all the other players in the lobby. | Short Constants.CMSG\_RANKINGS  int gameId |
| **RequestPrizes**  At the end of the gamemode the user requests a prize from the server and the server responses with a prize based on the user’s rank. | Short Constants.CMSG\_PRIZES  String Username |
| **RequestDead**  The client tells the server that it’s has died and the server response to all the other users in the lobby with ResponseDead | Short Constants.CMSG\_DEAD  int gameId |
| **RequestCollision**  The client sends the player that it hit and the damage calculated through the damage modifier to the server, and responses the damage to the user that was hit with ResponseCollision  Damage=(base\_damage×weight\_modifier×speed modifier×inflicting\_part)/(receiving\_part) | Short Constants.CMSG\_COLLISION  int playerId //  int damage // result of damage formula |

==================================================================

Server Side Protocol (Responses)

==================================================================

|  |  |
| --- | --- |
| **Type and usage** | **Format** |
| **ResponseRegistration**  Affirmative response to RequestRegistration. flag = 1 means registration was successful. | Short Constants.SMSG\_REGISTER  int flag (0 invalid, 1 valid) |
| **ResponseAuth**  Affirmative response to RequestLogin. It is followed by creating a number of ResponseRenderCharacter. All ResponseRenderCharacter will be queued up into all OTHER users’ update queue.  If flag is 1: send character list to the client | Short Constants.SMSG\_AUTH  int flag (0 invalid, 1 valid)    *IF flag is 1:*  int characterListSize  *for each character in list:*  String characterName  int id  int type |
| **ResponseLogout**  Client sends out request to disconnect. Server replies and removes user from current list also inform other clients to remove the user. Followed by creating a number of ResponseRemoveUser. All ResponseRemoveUser will be queued up into all OTHER users’ update queue. | Short Constants.SMSG\_DISCONNECT |
| **ReesponseAccountInformation**  Client requests for their account information to be email to them. If the server finds the account based on the provided information it will respond with a flag.  0 - Tells the user that the information they provided was invalid.  1 - Tells the user that the information they provided was valid and that an email has been sent with their account information. | Short Constants.SMSG\_FORGOT\_PASSWORD  int flag (0 invalid, 1 valid) |
| **ResponseCharacterCreation**  Affirmative response to RequestCharacterCreation.  flag = 1 means that the character was successfully created.    If flag is 1: send character list to the client | Short Constants.SMSG\_CHARACTER\_CREATE  int flag (0 invalid, 1 valid)    *IF flag is 1:*  int characterListSize  *for each character in list:*  String characterName  int id  int type |
| **ResponseRenderCharacter**  Create one character in client game world representing one existing user who already logs in.  The new character in the world will be associated with username. | Short Constants.SMSG\_RENDER\_CHARACTER  String username  int carType  int carPaint  int carTires |
| **ResponseRemoveUser**  One client disconnects. Server informs other clients to remove the user in their game worlds. | Short Constants.SMSG\_REMOVE\_CHARACTER  String username |
| **ResponseChat**  The server responds to the chat function from client through Requeschat. The server uses an int for which faction that user belongs to and a String for the message being sent. | Short Constants.SMSG\_CHAT  String username  String message |
| **ResponseMove**  server response a change to their location, and  isMovingflag. This is used when server moves as the cilent wishes to move or stop moving. Server will update other users with the RequestMove from the client. | Short Constants.SMSG\_MOVE  String username  Double x //location vector  Double y  Double z  Double h //facing direction |
| **ResponsePowerUpUse**  The server tells all users that username must run the animation for powerId. | Short Constants.SMSG\_POWER\_UP  String username  int powerId |
| **ResponsePowerUpPickUp**  Response from the server after RequestPowerUpPickUp, if the server validates the pick-up ‘validate’ is equal to 1 | Short Constants.SMSG\_POWER\_UP\_PICK\_UP  int validate // 1 or 0 |
| **ResponseChangeHealth**  The server increases or decreases carID’s health. | Short Constants.SMSG\_HEALTH  String carID  int healthChange |
| **ResponseEnterLobby**  Response from the server after RequestEnterLobby if the server validates if the user is able to enter the lobby. | Short Constants.SMSG\_ENTER\_LOBBY  String Username  int validate |
| **ResponseEnterGameLobby**  Response from the server after RequestEnterGameLobby if the server validates if the user is able to enter the specified game lobby. | Short Constants.SMSG\_ENTER\_GAME\_LOBBY  String Username  int validate |
| **ResponseEnterGameName**  Response from the server after RequestEnterGameName if the server validates if the user is able to enter the lobby given. | Short Constants.SMSG\_ENTER\_GAME\_NAME  String Username  int validate |
| **ReponseCreateLobby**  Response from the server after RequestCreateLobby if the server validates the users created lobby | Short Constants.SMSG\_CREATE\_LOBBY  String LobbyName  int validate |
| **ReponsePrivateChat**  Response from the server after RequestPrivateChat to send a private message to a specific user | Short Constants.SMSG\_PRIVATE\_CHAT  String UsernameFrom  String UsernameTo  String message |
| **ResponseInvite**  Response from the server after RequestInvite to send the invite to the specified user. | Short Constants.SMSG\_INVITE  String UsernameFrom  String UsernameTo  String GameLobby |
| **ResponseCarChoice**  Response from the server after RequestCarChoice to confirm car choice and allow other clients to render the proper car for opponents. | Short Constants.SMSG\_CAR\_CHOICE  String Username  Int carType |
| **ResponseCarPaint**  Response from the server after RequestCarPaint to confirm color choice and allow other clients to render the proper car color for opponents. | Short Constants.SMSG\_CAR\_PAINT  String Username  Int paintType |
| **ResponseCarTires**  Response from the server after RequestCarTires to confirm car choice and allow other clients to render the proper car tires for opponents. | Short Constants.SMSG\_CAR\_TIRES  String Username  Int tireType |
| **ResponseGaragePurchase**  Response from the server after RequesGaragePurchase to confirm user can make the purchase. | Short Constants.SMSG\_GARAGE\_PURCHASE  String Username  Int Purchase  int validate |
| **ResponseResults**  The server responses with either what place they finished on the race, or where the placed in the destruction derby, also the results of all the other players that have finished in their lobby. | Short Constants.SMSG\_RESULTS  int place  int number\_of\_players  // list of ‘number\_of\_players’ players  [String username, int place] |
| **ResponseRankings**  The server response with the list of players and their ranking in the lobby. | Short Constants.SMSG\_RANKINGS  Short type  int number\_of\_players  // list of ‘number\_of\_players’ players  [String username, int score] |
| **ResponsePrizes**  Based on the player’s rank during the game. The server returns an itemId for what prize the user has unlocked. | Short Constants.SMSG\_Prizes  int itemId |
| **ResponseDead**  The server responses to all of the other players in the lobby, once a player has lost all of his health. | Short Constants.SMSG\_DEAD  int playerid |
| **ResponseCollision**  The damage from the server request is sent to the player that was hit. | Short Constants.SMSG\_COLLISION  int damage // result of damage formula |