Protocol Team:

Madhulika Indukuru

Daanyaal Syed

Jonathan Young

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

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..

**Request Constants**

CMSG\_AUTH 101

CMSG\_DISCONNECT 102

CMSG\_REGISTER 103

CMSG\_CREATE\_CHARACTER 104

CMSG\_CHAT 105

CMSG\_MOVE 106

CMSG\_ATTACK 107

CMSG\_HEALTH 108

CMSG\_CONTROL\_POINT\_STATE 111

CMSG\_CONTROL\_POINT\_CAP 112

REQ\_HEARTBEAT 301

**Response Constants**

SMSG\_AUTH 201

SMSG\_DISCONNECT 202

SMSG\_REGISTER 203

SMSG\_CREATE\_CHARACTER 204

SMSG\_CHAT 205

SMSG\_MOVE 206

SMSG\_ATTACK 207

SMSG\_HEALTH 208

SMSG\_RESOURCE 209

SMSG\_CONTROL\_POINT\_STATE 211

SMSG\_CONTROL\_POINT\_CAP 212

SMSG\_RENDER\_CHARACTER 310

SMSG\_REMOVE\_CHARACTER 311

SMSG\_SPAWN\_GUARDS 312

SMSG\_DESTROY\_NPC 313

SMSG\_SPAWN\_GOLEMCP 321

SMSG\_DESTROY\_GOLEMCP 322

SMSG\_SPAWN\_GOLEM\_NPC 323

SMSG\_GOLEM\_PIECE 324

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

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  String message |
| **RequestRegistration**  Client registers a new account with the server which includes a username and password. The server validates this and responds with ResponseRegistration. | Short Constants.CMSG\_REGISTER  String Username  String Password |
| **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  String factionId  int classType |
| **RequestChat**  The client chats to other clients through the chat function. The client uses an int for which faction that user belongs to and a String for the message being sent. | Short Constants.CMSG\_CHAT  int factionId // which faction to broadcast to  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 | Short Constants.CMSG\_MOVE  Float x //location vector  Float y  Float z  Float h //facing direction  int isMoving // 0 or 1 |
| **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 |
| **RequestCharacterAttack**  When the client presses a button bound to a particular attack which is differentiated through attackId. The server responds by animating their attack through ResponseCharacterAttack. | Short Constants.CMSG\_ATTACK  int attackId // which attack am i using |
| **RequestCharacterChangeHealth**  When the client receives or deals damage through an attack or receives healing, the health of each player affected is updated. The server responds by updating all health bars involved through ResponseCharacterChangeHealth. | Short Constants.CMSG\_HEALTH  int healthChange // can be positive or negative |
| **RequestSetControlPointState**  When the client is interacting with a control point the control point changes colors based on if it’s being taken or contested. The server responds with ResponseSetControlPointState. | Short Constants.CMSG\_CONTROL\_POINT\_STATE  int controlPointId // which control point  int controlPointState // 1 red, 2 blue, 3 purple |
| **RequestControlPointCapture**  When the client successfully takes a control point they begin to gain resources and have access to healing. The server responds with ResponseControlPointCapture. | Short Constants.CMSG\_CONTROL\_POINT\_CAP  int controlPointId  int factionId |

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

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. | Short Constants.SMSG\_AUTH  int flag (0 invalid, 1 valid) |
| **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 |
| **ResponseCharacterCreation**  Affirmative response to RequestCharacterCreation.  flag = 1 means that the character was successfully created. | Short Constants.SMSG\_CHARACTER\_CREATE  int flag (0 invalid, 1 valid) |
| **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 characterName  int factionId |
| **ResponseRemoveUser**  One client disconnects. Server informs other clients to remove the user in their game worlds. | Short Constants.SMSG\_REMOVE\_CHARACTER  String characterName |
| **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  int factionId  String message |
| **ResponseCharacterMovement**  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 Requestcharactermove from the client. | Short Constants.SMSG\_MOVE  String username  Float x //location vector  Float y  Float z  Float h //facing direction  int isMoving // 0 or 1 |
| **ResponseCharacterAttack**  The server tells all users that characterName must run the animation for attackId. | Short Constants.SMSG\_ATTACK  String characterName  int attackId |
| **ResponseCharacterChangeHealth**  The server increases or decreases characterName’s health. | Short Constants.SMSG\_HEALTH  String characterName  int healthChange |
| **ResponseControlPointState**  The server informs all clients that a control point has changed state. | Short Constants.SMSG\_CONTROL\_POINT\_STATE  int controlPointId // which control point  int controlPointState // 1 red, 2 blue, 3 purple |
| **ResponseControlPointCapture**  When the server successfully takes a request from client, they begin to gain resources and have access to healing. | Short Constants.SMSG\_CONTROL\_POINT\_CAP  int controlPointId  int factionId |
| **ResponseChangeResourcePoints**  Modifies factionId’s resource points by resourceAmount. | Short Constants.SMSG\_RESOURCE  int factionId  int resourceAmount |
| **ResponseSpawnGuards**  When the client successfully takes a control point five guards of type guardId spawn at control point controlPointId. They are aligned to factionId. | Short Constants.SMSG\_SPAWN\_GUARDS  int controlPointId  int factionId  int guardId |
| **ResponseSpawnGolemControlPoint**  Spawn the golem control point at locationId (each of the possible golem control point spawn locations have a locationId). | Short Constants.SMSG\_SPAWN\_GOLEMCP  int locationId |
| **ResponseDestroyGolemControlPoint**  Destroy the golem control point at locationId (locationId is needed in case there are multiple golem control points spawned). | Short Constants.SMSG\_DESTROY\_GOLEMCP  int locationId |
| **ResponseSpawnGolem**  Create the golem NPC. It should spawn at the control point identified by controlPointId. | Short Constants.SMSG\_SPAWN\_GOLEM\_NPC  int controlPointId |
| **ResponseGolemPiece**  Adds a golem piece to factionId’s golem piece counter (Golem spawns when number of golem pieces is 3). | Short Constants.SMSG\_GOLEM\_PIECE  int factionId |
| **ResponseDestroyNPC**  When the hit points of a certain NPC falls to zero, remove that NPC from the server. | Short Constants.SMSG\_DESTROY\_NPC  int npcId // which NPC to destroy |