Frederick Harrington
Fred Barnes
CO600: Standards
14 March 2016

JSON Message Standards

Anon Users:

Direction: Mobile User to Golang Server
Event: "joinlobby"

data: {
    nickname: string
    lobbyid: int}
Server Action: Should trigger a response emit from the golang server to confirm if successful by checking if the username already is in use within the lobby
If successful should add the socket to the lobby namespace.

Direction: Mobile User to Golang Server
Event: “setready"

data: {
    nickname: string
    ready: bool}
Server Action: Set ready bool for AnonUser, and emit updatelobby to all.

Direction: Mobile User to Golang Server
Event: “leavelobby"

data: {none}
Server Action: Lookup anonuser in map and remove from session, emit updatelobby

---

Direction: Mobile User to Golang Server
Event: “msgserver"

data: {
    msg: JSON{} //generic JSON data}
Server Action: Forward onto game server for game processing.

Host Users:

Direction: Desktop User to Golang Server
Event: “hostlobby"

data: {
hostid: int //User's unique ID}

Direction: Desktop User to Golang Server
Event: “start"

data: {none}
Server Action: Forward to game server, server should emit a response to all users

---

Direction: Desktop User to Golang Server
Event: “kick"

data: {
nickname: string //username of anonuser to be kicked
reason: string //optional}
Server Action: Update lobby user data structure, emit updated lobby to all, emit kick message to kicked user

---

Direction: Desktop User to Golang Server
Event: “terminate"

data: {none}
Server Action: Forward to game server, call lobby end functions, emit response to all?

Direction: Desktop User to Golang Server
Event: “msgserver"

data: {
msg: interface{} //generic JSON data - JSON Object}
Server Action: Forward to game server for game processing

Game Server

Direction: Golang Server to Desktop User
Event: “lobbyconnected"

data: {
lobbyid: int
feedback: error}
Front-end Action: Load up lobby and join lobby namespace (lobby id) to listen for new events.

Direction: Golang Server to All Users
Event: “gamestart"

data: {
response: bool
feedback: error}
Front-end Action: Begin loading into the game

Direction: Golang Server to All Users
Event: “gameend"

data: {
response: bool
feedback: error}
Front-end Action: Stop emitting messages to server? (leave lobby)

Direction: Golang Server to Single (Desktop OR Mobile) User
Event: “msgplayer"

data: {
msg: JSON{} //game specific JSON data}
Front-end Action: Process game specific JSON data (msg variable)

Direction: Golang Server to All Users
Event: “msgall"

data: {
msg: JSON{} //game specific JSON data}
Front-end Action: Process game specific JSON data (msg variable)

Direction: Golang Server to All Users
Event: “updatelobby"

data: {
players: []JSON //Array of JSON objects (users)}
Front-end Action: Update the list of players in the lobby and ready status, etc

---

Direction: Golang Server to Single (Mobile) User
Event: "kick"
data: {
reason: string //optional}
Front-end Action: Display kicked message to user and disconnect them from lobby?

Direction: Golang to Mobile User
Event: "joined"
data: {
response: bool
feedback: error}

Game Server to GoLang Server

All messages are sent over TCP in JSON encoded bytes in the following format:
(A JSON object containing a JSON object)

TO GOLANG SERVER:

```
{
        event: "created"
        msg: string}
```

RECEIVED FROM GOLANG:

```
{
        event: "new"
        players: int // user count on start
        maxplayers: int} // max players for session
```

TO A SINGLE USER:

```
{
        event: "msgplayer"
        player: int
        msg: {}(JSON Object)}
```

TO ALL USERS:

```
{
        event: "msgall"
        msg: {}(JSON Object)}
```

RECEIVED FROM USERS:

```
{
        player: int
        msg: {}(JSON Object)}
```