**Lobby Server Spec**

**1.0 Introduction**

Clients perform **actions** and receive **events**. Actions can update the **Lobby Model** which typically causes several events to be emitted. Actions do not have a return value as such. Failed or non-permitted actions will cause a ***ActionRejected*** event to get emitted.

All actions and events are JSON encoded. All field names are non-delimited lowercase words. All actions have the fields listed in section 3.

**2.0 Lobby Model**

players{

(playerhash) : {

name : string,

gamehash : string,

playerhash : string

}

},

games{

(gamehash) : {

name : string,

password : string,

ownerhash : string,

playerhashes : [string, …],

invited : [string, …],

gamehash : string,

maxplayers : int

}

}

**3.0 Actions**

All actions will have an ‘action’ field which denotes which action it is. The values in the action field are all lowercase, underscore delimited.

Login {username : string, password : string}

* Adds a player to Model.players
* Emits uJoinLobby event to the player joining
* Emits pJoinLobby event to all players

Logout {}

* Removes a player from Model.players
* Removes a player from any game they are part of
* Emits uLogout to the player logging out
* Emits pLeaveLobby to all players
* May emit pLeaveGame

JoinGame {gamehash : string, password : string}

* Adds a player hash to the game Model.games[gamehash].playerhashes
* Sets game hash in Model.players[userhash].game
* Emits uJoinGame event to player joining
* Emits pJoinGame event to all players
* Fails if password doesn’t match
* Fails if maxplayers already in game
* Fails if game doesn’t exist

CreateGame {gamename : string, maxplayers : int, password : string}

* Create a new game at Model.games[gamehash]
* Sets creating player (userhash) to Model.games[gamehash].ownerhash
* Emits newGame event
* Emits uJoinGame event
* Emits pJoinGame event
* Fails if player is already in a game
* Fails if a game of that name already exists

LeaveGame {gamehash : string}

* Removes player from Model.games[gamehash].playerhashes
* If player is owner remove game from Model.games[gamehash]
* Emits uLeaveGame event
* Emits pLeaveGame event
* May emit deleteGame event
* fails if game doesn’t exist
* fails if player is not in game

RemovePlayer {gamehash : string, playername : string}

* Removes playername from Model.games[gamehash].players
* Emits uKicked event
* Emits pLeaveGame event
* fails if game doesn’t exist
* fails if player not in game

StartGame {gamehash : string}

* Starts a game on the game server
* Emits a startGame event (gamehash may be different)
* emits deleteGame event
* emits multiple pLeaveLobby events
* fails if game doesn’t exist
* fails if maxplayers not in game

InvitePlayer {playername : string, gamehash : string}

* adds the player name to Model.games[gamehash].invited
* that player will not require a password to join
* Emits an gameInvite event
* fails if game is full
* fails if player is in a game already
* fails if the game doesn’t exist

RequestPlayers {}

* emits a playerList event

RequestGames {}

* emits a gameList event

RegisterPlayer{playername: string, password: string, email: string}

* Emits action\_sucess event when successful.

DeletePlayer{password : string}

RecoverPassword{playername: string} future implementation

**4.0 Events**

All events will have an ‘event field which denotes which event it is. The values in the action field are all lowercase, underscore delimited.

actionRejected {action : string, message : string}

actionSuccess {action : string }

uJoin {playerhash : string}

pJoinLobby {playername : string}

pLeaveLobby {playername : string}

uLeaveGame {gamehash : string}

uKick {gamehash : string}

pLeaveGame {playername : string, gamehash : string}

newGame {gamename : string, ownername : string, pwrequired : bool}

deleteGame {gamehash : string}

startGame {ip : string, port : int, gamehash : string}

chat {playername : string, message : string}

adminChat {playername : string, text : string}

gameInvite {playername : from, gamehash : string}

playerList {players : [{playername : string}]}

gameList {(gamehash) : [{

name : string,

pwrequired : bool,

ownername : string,

playernames : [string, …],

gamehash : string,

maxplayers : int

}, …]

}

5.0 Connection Algorithm

The JSON-Server only waits for new connections. It immediately passes these connections off to the ConnectionManager. A LobbyConnection is created to hold the Thread that listens for incoming packets.

The ConnectionManager waits for the initial communications:

* Login
* RegisterPlayer
* DeletePlayer
* RecoverPassword

On successful Login the Connection is associated with a username which is added to the lobby model.

All future incoming actions are vetted by the ConnecitonManager via the LobbyConnection object.