

Connect 4 Web Connection Protocols

The many nuances of trying to work with other people

Lewis Miller| 15/01/18 | Noah Stride

# Capitalization

While capitalization standards between the two coders are understood to be different, with Lewis {hitherto referred to as L} primarily using PascalCase and Noah {N} primarily using camelCase, the two understand that message type names from both Client -> Server and Server -> Client will utilize PascalCase, while data structures contained within these will utilize camelCase. This allows for the easy differentiation of message types and their data.

# Communication

## Client -> Server

### MessageShell [type: “” \ agent: “” \ data: {}]

Used when the server has not assigned a JavaScript Web Token to the client. This is to be used only once to register a client with the server. All subsequent communications should utilize the *AuthenticatedMessageShell* for data security purposes. The *agent* property allows the server to identify which client is in use.

### Registration [type: “Registration” \ agent: “” \ data: {username: “”}]

Utilizes the *MessageShell* class. Takes the user’s chosen username and registers it with the server.

### AuthenticatedMessageShell [type: “” \ agent: “” \ jwt: “” \ data: {}]

Used for all communication subsequent to the original registration request. This class provides a JWT that allows communications with the server to be secure.

### MatchRequest [type: “MatchRequest” \ agent: “” \ jwt: “” \ data: {}]

Requests a place in the queue for an online game. Sent after receiving a *RegistrationReturn* from the server.

### ChatMessage [type: “ChatMessage” \ agent: “” \ jwt: “” \ data: {message: “”}]

Sends a chat to the server. This will be distributed to both clients.

### PlayPosition [type: “PlayPosition” \ agent: “” \ jwt: “” \ data: {column: 123}]

Sends details of your move to the server. The result will be distributed to both clients.

## server -> client

### RegistrationReturn [type: “RegistrationReturn” \ agent: “” \ data: {jwt: “”}]

Returns a JWT to the client who requested it. This will be used in all subsequent communication from the client.

### ChatMessageReturn [type: “ChatMessageReturn” \ agent: “” \ data: {sender: “” \ message: “”}]

Distributes a chat message between two clients, with the name of the player who sent it. This allows for simpler coding as no logical decisions need to be made

### MatchRequestReturn [type: “MatchRequestReturn” \ agent: “” \ data: {opponent: “” \ localNum: 123 \ currentPlayer: 123}]

Begins a game. Contains details of the opponent’s name, the local client’s assigned number, and the number of the client that will begin the game {This value is randomly chosen}

### MatchUpdate [type: “MatchUpdate” \ agent: “” \ data: {board: [6][5] \ currentPlayer: 123}]

Sends the contents of the board and the current player’s number to both clients.

### MatchEnd [type: “MatchEnd” \ agent: “” \ data: {winner: 123 \ type: 123}]

Informs the clients that a victory has been reached.

Type: 1 = Standard victory [4 in a row]

Type: 2 = Victory by opponent disconnect