James Grant
Fred Barnes
CO600: Design
Message Data Specification

# Platform Socket.IO Events and JSON Message Format Specification

## Document Outline:

This document contains the specification for Socket.IO events used in communicating specific functionality with the back-end Go engine. The functionality for the Mobile and Desktop users defined here is specifically used for the platform development by the ANEXD team, and these Socket.IO events are not intended to be exposed to application developers on the platform. This was done by creating generic functions to the ANEXD front end API, i.e. anexd.send() to execute a "msg" Socket.IO event.

The Application Server specification is available as an option to connect your application server to a lobby instance using a Socket.IO **client** library. This is available as an alternative to more standard methods that are also supported such as TCP and WebSockets.

All three Application Server connection types (TCP, WS, Socket.IO) receive messages in the same JSON format, as defined by the specification in this document.

## Document Contents:

# 1. Mobile User Specification:

**Use Case: On successful socket.io connection (should be an automatic emit by respective client).**

| **Sent by: Mobile User** | | **Response to: Mobile User** | |
|---|---|---|---|
| Socket.IO Event: | "client" | Socket.IO Event: | "client" |
| Data: | "mobile" (string literal) | Data: | boolean (true/false success) |

**Use Case: Joining lobby by lobby ID (Mobile Users can join existing open lobbies).**

| **Sent by: Mobile User** | | **Response to: Mobile User** | |
|---|---|---|---|
| Socket.IO Event: | "joinlobby" | Socket.IO Event: | "joinlobby" |
| Data: | JSON Object<br>{<br>lobbyid: (i.e. 6-digit number in STRING format)<br>username: string (entered nickname)<br>} | Data: | boolean (true/false success) |

**Use Case: Setting status as ready in lobby.**

| **Sent by: Mobile User** | |
|---|---|
| Socket.IO Event: | "setready" |
| Data: | boolean |

**Use Case: Attempting to soft leave a lobby.**

| **Sent by: Mobile User** | |
|---|---|
| Socket.IO Event: | "leave" |
| Data: | n/a |

**Use Case: Getting the App ID (ID stored in the Database)**

| **Sent by: Mobile User** | | **Response to: Mobile User** | |
|---|---|---|---|
| Socket.IO Event: | "getappid" | Socket.IO Event: | "getappid" |
| Data: | n/a | Data: | Number (app id from database) |

**Use Case: Send an application specific message to the game server.**

| **Sent by: Mobile User** | | **Response:** |
|---|---|---|
| Socket.IO Event: | "msg" | *None – any issues and failures such as packet loss are ignored (messages are real-time)* |
| Data: | JSON Object | |

**Use Case: Receive an application specific message from the game server.**

| | |
|---|---|
| | **Received by: Mobile User**<br>Socket.IO Event: "msg"<br>Data: JSON Object |

**Use Case: Confirmation that you have been removed from the room (soft leave)**

| | |
|---|---|
| | **Received by: Mobile User**<br>Socket.IO Event: "leave"<br>Data: n/a |

**Use Case: Mobile User was kicked by the host.**

| | |
|---|---|
| | **Received by: Mobile User:**<br>Socket.IO Event: "kicked"<br>Data: string (reason entered by host) |

## 2. Desktop User Specification:

**Use Case: On successful socket.io connection (should be an automatic emit by respective client).**

| | |
|---|---|
| **Sent by: Desktop User**<br>Socket.IO Event: "client"<br>Data: "desktop" (string literal) | **Response to: Desktop User**<br>Socket.IO Event: "client"<br>Data: boolean (true/false success) |

**Use Case: Joining lobby by lobby ID (Desktop User joins to become host user in a new lobby).**

| | |
|---|---|
| **Sent by: Desktop User**<br>Socket.IO Event: "hostlobby"<br>Data: JSON Object<br>{<br>lobbyid: (i.e. 6-digit number in STRING format)<br>username: string (scope var?)<br>} | **Response to: Desktop User**<br>Socket.IO Event: "hostlobby"<br>Data: boolean (true/false success) |

**Use Case: Setting status as ready in lobby.**

| | |
|---|---|
| **Sent by: Desktop User**<br>Socket.IO Event: "setready"<br>Data: boolean | |

**Use Case: Kicking a Mobile User from the lobby as the host.**

| **Sent by: Desktop User** | **Response to: Desktop User** |
|---|---|
| Socket.IO Event: "kick"<br>Data: JSON Object<br>{<br>username: string (nick in lobby)<br>reason: string (entered by host)<br>} | Socket.IO Event: "kick"<br>Data: JSON Object<br>{<br>response: boolean (t/f success)<br>feedback: string (why t/f)<br>} |

**Use Case: Initiating connection with the game server (attempting to start the lobby with the current users within).**

| **Sent by: Desktop User** | **Response to: Desktop User (potentially multiple responses as some are used for feedback updates, do not take action until either of the *complete* or *failed* Booleans equal true)** |
|---|---|
| Socket.IO Event: "start"<br>Data: n/a | Socket.IO Event: "start"<br>Data: JSON Object<br>{<br>complete: boolean<br>failed: boolean<br>feedback: string<br>} |

**Use Case: Once Desktop and Anon Users have started the respective application, inform the game server that it has successfully loaded and is ready to send/receive messages.**

| **Sent by: Desktop User** | |
|---|---|
| Socket.IO Event: "launch"<br>Data: n/a | |

**Use Case: Send an application specific message to the game server.**

| **Sent by: Desktop User** | **Response:** |
|---|---|
| Socket.IO Event: "msg"<br>Data: JSON Object | *None – any issues and failures such as packet loss are ignored (messages are real-time)* |

## 3. All Users Specification:

The following events require functionality implemented on both Desktop and Mobile users upon these global events being triggered in the Socket.IO lobby Room.

**Use Case: Update the list of players displayed in the lobby.**

| | |
|---|---|
| | **Received by: All Users**<br>Socket.IO Event: "updatelobby"<br>Data: []JSON (array of the following JSON Objects):<br>{<br>nickname: string<br>ready: boolean<br>} |

**Use Case: Game server has successfully connected and instantiated the lobby session, and is ready for the users to load into the web application.**

| | |
|---|---|
| | **Received by: All Users (*occurs only when the value 'complete' is TRUE*)**<br>Socket.IO Event: "start"<br>Data: JSON Object:<br>{<br>complete: boolean<br>failed: boolean<br>feedback: string<br>} |

## 4. Application Server Specification.

### a. Using Socket.IO to connect your application to ANEXD:

**Use Case: On successful socket.io connection (should be an automatic emit by respective client).**

| **Sent by: Game Server** | **Response to: Game Server** |
|---|---|
| Socket.IO Event: "client"<br>Data: "server" (string literal) | Socket.IO Event: "client"<br>Data: boolean (true/false success) |

**Use Case: Joining lobby by lobby ID (Game Server joins to associate socket instance with respective lobby).**

| **Sent by: Game Server** | | **Response to: Game Server** | |
|---|---|---|---|
| Socket.IO Event: | "connectlobby" | Socket.IO Event: | "connectlobby" |
| Data: | string (lobbyid) | Data: | boolean (true/false success) |

**Use Case: Receiving a generic JSON structure for application specific processing. As defined in the JSON Events Specification.**

| | **Received by: Game Server** | |
|---|---|---|
| | Socket.IO Event: | "in" |
| | Data: | JSON Object |

**Use Case: Sending a generic JSON structure for application specific processing. As defined in the JSON Events Specification.**

| **Sent by: Game Server** | | |
|---|---|---|
| Socket.IO Event: | "out" | |
| Data: | JSON Object | |

## b. JSON Events Specification:

The ANEXD platform aims to provide a generic JSON structure that does not restrict the potential of the applications developed on it. However, applications developed on ANEXD require the events in this section implemented into their server.

All messages sent to the application server will be in JSON format (in a raw byte stream), and may need to be reconstructed on the server, most commonly if implemented with a TCP/WS connection. Some Socket.IO implementations decode the JSON automatically.

An example JSON message received from a user would look like:

```
{
        event: "msg"
        player: 1
        msg: JSON Object
}
```

## Inbound Events (Go Engine sending to Application Server):

**Use Case: Receiving application specific JSON data from a player.**

| **Event Value: "msg"**<br>**JSON Variables:** | **Action:** |
|---|---|
| event: string<br>player: Number (sender)<br>msg: JSON Object (application data) | Process the JSON Object stored in 'msg' as an action from the user 'player'. |

**Use Case: Create a new instance of a lobby session.**

| **Event Value: "new"**<br>**JSON Variables:** | **Action:** |
|---|---|
| event: string<br>players: Number (number of users at start)<br>maxplayers: Number (max number of users allowed in total) | Instantiate a new lobby instance on the server with the sent parameters, and send a 'created' event back to the engine. |

**Use Case: Desktop user has loaded the application up and is ready to begin sending/receiving messages.**

| **Event Value: "launch"**<br>**JSON Variables:** | **Action:** |
|---|---|
| event: string | Begin communicating using 'msgall' and 'msgplayer' events. |

**Use Case: Safely ending an instance of a lobby session.**

| **Event Value: "end"**<br>**JSON Variables:** | **Action:** |
|---|---|
| event: string | Stop communication and safely close down the lobby instance. |

## Outbound Events (Application Server sending to Go Engine):

**Use Case: Confirm that the lobby session has been successfully created and is ready to send/receive messages.**

| **Event Value: "created"**<br>**JSON Variables:** | |
|---|---|
| event: string | |

**Use Case: Send a message to 1 specific player.**

| | |
|---|---|
| **Event Value: "msgplayer"**<br>**JSON Variables:**<br>event: string<br>player: Number (player number in lobby, 0 for the desktop host user)<br>msg: JSON Object (application data) | |

**Use Case: Send a message to all players in the lobby.**

| | |
|---|---|
| **Event Value: "msgall"**<br>**JSON Variables:**<br>event: string<br>msg: JSON Object (application data) | |