Interfaces in Json

# begin\_to\_ game

## Request Description

### Precondition & Background:

When a user opens the game, CLIENT should compose this message and send it first. In this game, “login” is not necessary. CLIENT can give a user name and corresponding uuid instead of “login”.

### Fields

**req**: begin\_to\_game

**user\_name**: A random string is generated by CLIENT as user name.

**uuid**: Universally Unique Identifier, it is also generated by CLIENT.

## Response Description

### Precondition & Background

After receiving “begin\_to\_game” request, SERVER will query a user list to check whether the uuid had already existed. If it does, SERVER can find its score and other information to reply to CLIENT. Otherwise, Server can give a note indicates it is a new user in this game.

In addition, SERVER will choose a map in game.

### Fields

**rep**: begin\_to\_game

**result**: “success” or “failure” will be filled in this field. It shows the request is success or not.

**failure\_note**: if the “result” is “failure”, this field will contain a statement to explain what reason to the failure is. Otherwise, this filed should be null.

**info**: it is a sub directory containing score and other information for the user. If the uuid doesn’t exist, this filed will be null.

**max\_score**: a key in the “info”, it is the max score the user played before.

**ava\_score**: a key in the “info”, it is the average score the user played before.

**level**: a key in the “info”, it represents the user’s playing level.

**map**: A index number is here, indicating which map is chosen in user’s game.

# begin to match

## Request Description

### Precondition & Background

If a user intends to play with other users via internet, this message should send first. Some network information should be included in this message in order to establish a connection between two players.

### Fields

**req:** begin\_to\_match

**uuid**: Universally Unique Identifier, it is also generated by CLIENT. It should be same as “begin\_to\_game”.

**level**: it represents the user’s playing level. The value is got from “begin\_to\_game” response message.

**ip**: user’s ip address.

**port**: user’s port number.

## Response Description

### Precondition & Background

This message should reply according to the request. SERVER will add the user in request to “match\_user\_list”. And SERVER will find a suited opponent from “match\_user\_list” to reply.

### Fields

**rep**: begin\_to\_match

**result**: “success” or “failure” will be filled in this field. It shows the request is success or not.

**failure\_note**: if the “result” is “failure”, this field will contain a statement to explain what reason to the failure is. Otherwise, this filed should be null.

**opponent**: it is a sub-directory, storing opponent’s information.

**uuid**: opponent’s uuid.

**ip**: opponent’s ip addres.

**port**: opponent’s port number.

**level**: opponent’s level.

# step

## Request Description

### Precondition and Background

This interface will gather the current step and current status of chessboard. These information can be applied to generate history and judge victory or not.

### Fields

**req**: step

**uuid**: Universally Unique Identifier, it is also generated by CLIENT. Its value should be same as “begin\_to\_game”

**direct\_vector**: it an array, having 2 elements. The first represents increment of x axis, the second is for y axis. So all the directories can be shown as [1,0], [-1,0], [0, 1], [0, -1], [1, -1], [1, 1], [-1, 1] and [-1, -1].

## Response Description

### Precondition and Background

SERVER will store current step of the user and judge whether the user is victory. If the user is victory, SERVER will record the score and adjust level of the user.

### Fields

**rep**: step

**result**: “success” or “failure” will be filled in this field. It shows the request is success or not.

**failure\_note**: if the “result” is “failure”, this field will contain a statement to explain what reason to the failure is. Otherwise, this filed should be null.

**victory**: if the user is victory after this step, “true” will be filled in this field. Otherwise, “false” is filled instead.

# end\_game

## Request Description

### Precondition and Background

When a game is ended by user or the winner is determined, this message should be sent to SERVER.

### Fields

req: end\_game

uuid: Universally Unique Identifier, it is also generated by CLIENT. Its value should be same as “begin\_to\_game”.

## Response Description

### Precondition and Background

SERVER will end the game and reply to CLIENT.

### Fields

**rep**: end\_game

**result**: “success” or “failure” will be filled in this field. It shows the request is success or not.

**failure\_note**: if the “result” is “failure”, this field will contain a statement to explain what reason to the failure is. Otherwise, this filed should be null.