Glico CP API document

KEMBO CO.,LTD. 上海賢房信息技術有限公司

test host http://glicocp.kembo88.com

2016.9.1

グループID生成

GET /api/get_gid

*QR不要

Request

```
GET /api/get_gid  HTTP/1.1
```

Paramater 参数

なし

Response json

```
HTTP/1.1 200

{
  "message": "success",
     "gid": "1
}
```

Respons

name	value
gid	グループID
qrHtml	QRコード

画像アップロード

POST /api/uploads

Example

Request

画像URLは固定で下記の使用で、フロント側でgidとuidをもとに生成する。
/uploads/{{geme_id}}/{{user_id}}/original.jpg　# 各ユーザの顔画像の path
/uploads/{{geme_id}}/share/original.jpg #シェーア画像の　path

Base64の変換をサーバでjpg変更する

```
POST /api/uploads  HTTP/1.1
```

Paramater 参数

name	value
file	file
user	1： リーダ　2: ユーザ 3:Share
gid	グループID (リーダ以外)

Response json

```
HTTP/1.1 200

{
  "message": "success",
     "gid": "1",
     "uid": "1"
     "qrHtml": "<div id="qr"><table><tr><td class="black"></td>
     ......
     <td class="white"></td></tr></table></div>"
}
```

Respons

name	value
gid	グループID

uid	ユーザID
qrHtml	QRコード　＊リーダのみ

share は successのみ

GET /api/message

Users.

Example

Request

```
POST /api/message
HTTP/1.1
```

Paramater 参数

name	value
gid	グループID
message	message_id

Response

```
HTTP/1.1 200
{
  "status": "success",
  "data": {
    "message": "success",
  }
}
```

Respons

user 用户

name	value
status	“”

APIエラー処理

code: 200　正常
code: 400　エラー
code: 500 サーバ例外

Response json

```
HTTP/1.1 400

{
  "status": "error",
  "message": "nothing user id"
 }
```

Response json

```
HTTP/1.1 500

{
  "status": "error",
  "message": "server exception  message"
 }
```

ウェブソケット

client to websocket　サンプル

```
App.drift.move("paramater 1(status)",paramater 2...);
```

websocket to client サンプル

```
App.drift = App.cable.subscriptions.create({
    channel: "DriftChannel",
    user_id: $("#drift_boat").data('user-id');

}, {
  connected: function() {},
  disconnected: function() {},
  received: function(data) {
    console.log(data);
  }
});

data:{
  "status": "user_upload_success",
   "uid":  "2"
}
```

paramater

name	value	type
gid	グループID	int
uid	ユーザID	int
turn_id	順番	int
status	ステータス	string
total_users	全員の数	int

各ユーザアップロード完了

user_upload_success

WS => リーダー

Json

```
data:{
  "status": "user_upload_success",
    "uid":  "2"
}
```

通信速度

user_network

ユーザ,リーダ => ws

```
App.drift.move("user_network",network);
```

アップロード締め切り <--追加

user_upload_close

リーダ=> ws

script

```
App.drift.move("user_upload_close");
```

アップロード締め切り通知 <--追加

user_upload_close_success

ws=> ユーザ、リーダ

Json

```
{
  "status": "user_trun_success"
}
```

N順通知

user_turn

ユーザ => ws

Json

```
App.drift.move("user_turn",user_id);
```

順番OK

user_turn_success

user turn の戻り値に自分の順番が帰る。全員に通知
リーダは全員が揃ってから通知が飛ぶ

ws => ユーザ、リーダ

json

```
{
  "status": "user_turn_success",
    "message": "success",
    "trun_id": "1〜10",
    "total_users": "1〜10"
}
```

順番リセット

user_turn_reset

リーダ => ws

Script

```
App.drift.move("user_turn_reset");
```

ローディングスタート

loading_start

パラメータに動画のトータル時間（ミリ秒）と同期間隔（ミリ秒）追加
リプレイ用のフラグ追加

リーダ=>ws

Script

```
App.drift.move("loading_start",total_ms,update_ms,replay);
```

ローディングスタート

loadting_start_success

リプレイ用のフラグ追加
ws => ユーザ、リーダ

Json

```
{
  "status": "loading_start_success",
  "network": "1-4",
  "message_id": "1〜10",
  "replay": "0 〜"
}
```

ダウンロード完了

user_download_success

ユーザ 、リーダ=> ws

script

```
App.drift.move("user_download_success");
```

動画スタート

animation_start

ws=> ユーザ,リーダ

Json

on

```
{
  "status": "animation_start"
}
```

動画同期

animation_update

ws=> ユーザ,リーダ

Json

loading_startで指定された間隔で発行される
経過時間（ミリ秒）
on

```
{
  "status": "animation_update",
  "elapsed": 100
  }
```

リーダShare ボタン押下

share_ok

リーダ => ws

最後のシェアボタン
パラメータに種別（1-2）追加

js

```
App.drift.move("share_ok","1,2");
```

Share

sahre_ok_success

ws=> ユーザ

最後のシェアを表示するため

json

```
{
  "status": "share_ok_success",
  "type": {{1,2}}
}
```