Game Function API

API Specifications

Game Function API

Game Function API

1.0 Login API (HTTP POST)

URL	https://rocket-api.aposcb.org/api/user/get-settings-demo	
Method	POST	
Params	host_id	0e83088027d4c42c8e9934388480c996
	access_token	demo01
	game_code	195
	is_promotion	0 or 1
Example	https://rocket-api.aposcb.org/api/user/get-settings-demo?host_id=0e83088027d4c42c8e9934388480c996&access_token=demo01&game_code=195&is_promotion=0	

- **host_id** and **access_token** will never change
- **game_code** will be hardcoded into game itself
- Store Response received into game memory and use in socket

Game Function API

jQuery Login API Example:

```
var user_data;
var get_user_data = function(){
    var settings = {
        "url": "https://rocket-bo.aposcb.org/api/user/get-settings-demo",
        "method": "POST",
        "timeout": 0,
        "headers": {
            "Content-Type": "application/x-www-form-urlencoded",
            "Origin": "https://st-dslot.apollogames.co",
            "Cookie": "PHPSESSID=nn88raavdkcqj3fhp4o2lnqft1"
        },
        "data": {
            "host_id": host_id,
            "access_token": access_token,
            "game_code": "195",
            "is_promotion": "0",
            "device_type": "Desktop",
            "country_code": "MY"
        }
    };
    $.ajax(settings).done(function (response) {
        user_data = response.data;
    });
};
```

Game Function API

2.0 Establish Socket Connection

The system uses a Socket API (Node.js) to support real-time communication.
During game development, it is necessary to establish a socket connection using the user data returned by the Login API.

Socket Endpoint: https://rocket-socket-binary.aposcb.org

```
const socket = io('https://rocket-socket-binary.aposcb.org', { transports :
['websocket'],  upgrade: false });
```

Game Function API

3.0 Socket Event Names

Reference events for API Socket connection. Game result and Player’s wallet to be displayed according to Response Data.

Event Name	Description
connect	Event fired on connected to API Socket (NodeJS)
subscribe	Emit user_data to API Socket
onSubscribeDone	Event fired after passing user data from Login API
kick-user	Force game session end, display error message to user
kick-user-maintenance	Force game end due to game maintenance, display error message to user
kick	Game error occured, display error message to user
balance	Update and display player balance in game

binary-bet	Emit bet data to API Socket:
	bet_data = {
	    denominator: 1,
	    bet_stake: 10, // bet amount
	    timestamp: now,
	    signature: …
	};

binary-result	Win/lose bet result for Event binary-bet

Game Function API

3.1 Socket.emit(“subscribe”)

Param	Data Type	Note	Example
host_id	String	Demo Host ID	0e83088027d4c42c8e9934388480c996
access_token	String	Demo token	demo01
game_code	Int	Demo game id	195
user_id	Int	Apply value from jQuery Login API	63011
credits	Int		10000
username	String		demo01
is_demo	Bool		1
rtp	Int		96
currency	String		MYR
is_apollo	Bool		1
default_bet	Int	Default Number to bet	1
ip_address	String	Player’s IP address	192.168.110.1
device_type	String	Chrome: get data from navigator.userAgent	Desktop
os_type	String		Windows
browser_type	String		Chrome
user_agent	String		Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.0.0 Safari/537.36
event_type	Int	event_type = 1 is promotion	0
timestamp	Int		
signature	String		

Example:

```
var now = Date.now();
var data = {
```

Game Function API

```
        host_id: "0e83088027d4c42c8e9934388480c996",
        access_token: "demo01",
        user_id: user_data.user_id,
        game_code: 195,
        credits: user_data.balance,
        username: user_data.username,
        device_type: "Mobile",
        ip_address: my_ip,
        is_demo: user_data.is_demo,
        rtp: user_data.rtp,
        currency: user_data.currency,
        is_apollo: user_data.is_apollo,
        default_bet: 0,
        device_type: getDeviceType(),
        os_type: getOS(),
        browser_type: getBrowser(),
        user_agent: window.navigator.userAgent,
        event_type: 0,
        timestamp: now,
        signature: ...
};
var data_encrypt = ecryptContoller.encrypt(JSON.stringify(data));
socket.emit('subscribe', data_encrypt);
```

3.2 Socket.on(“onSubscribeDone”)

Param	Data Type	Note	Example
status_code	Int	Default 0	0
timestamp	Int		
signature	String		

Example:

```
{
      status_code: 0
      signature: "981033da7a361b88c09e7b8c69d42bdf"
      timestamp: 1764652387197
}
```

3.3 Socket.emit(“binary-bet”)

Param	Data Type	Note	Example
denominator	Int	Denom for previous slot games, value = 1 for now	1
bet_stake	Int	Total bet amount	10
timestamp	Int		
signature	String		

Example:

```
var bet_stake = 10;
var now = Date.now();
var data = {
      denominator: 1,
      bet_stake: bet_stake,
      timestamp: now,
      signature: ...
};
var data_encrypt = ecryptContoller.encrypt(JSON.stringify(data));
socket.emit('binary-bet', data_encrypt);
```

3.4 Socket.on(“binary-result”)

Param	Data Type	
data	Object	Result Data
timestamp	Int	
signature	String	

Example:

```
{
        data: {
                result: 'win',
                amount: 10
        }
        signature: "981033da7a361b88c09e7b8c69d42bdf"
        timestamp: 1764652387197
}
```

3.5 Socket.on("balance")

Param	Data Type	Note
data	Object	Balance Data
timestamp	Int	
signature	String	

Example:

```
{
        data: {
                balance: 10010
        },
        signature: "abe43c762784ab7a5f3223520fc1a0d1",
        timestamp: 1764652387197
}
```

Game Function API

3.6 Socket.on(“kick-user”)

Param	Data Type	
status	Int	Status Code
message	String	Error Message

Example result:

```
{
	message: "Invalid signature."
	status_code: 1014
}
```

Game Function API

4.0 Security: Encryption and Signature

Required for every socket emit and listener event. Regular event signature formula as follow:
Javascript MD5(“event name” + timestamp + host_id + access_token)

binary-bet, **binary-result** and **balance** signature as follow:
Javascript MD5(“event name” + amount + timestamp + host_id + access_token)

Use custom JS ecrypt file for encryption.
Download here :
https://drive.google.com/file/d/17ZB1zNgQ4E-d8b5pugQp9L0jh68G0EBq/view?usp=sharing

Example:

```
import * as ecryptContoller from 'ecrypt';
socket.on('onSubscribeDone', (res) => {
      var data = JSON.parse(ecryptContoller.decrypt(res));
      if(checkSignature('onSubscribeDone', data.timestamp, host_id, access_token,
data.signature)){
      // signature is valid
      } else {
      // signature is invalid
      };
});

var data = {
      ...
      timestamp: now,
      signature: getSignature('subscribe', now, host_id, access_token)
};
var data_encrypt = ecryptContoller.encrypt(JSON.stringify(data));
socket.emit('subscribe', data_encrypt);

function getSignature(method, timestamp, host, session){
      return md5(method.toString() + timestamp.toString() + host.toString() +
session.toString());
}
function checkSignature(method, timestamp, host, session, signature){
      var my_signature = md5(method.toString() + timestamp.toString() +
host.toString() + session.toString())
      return (signature == my_signature) ? true : false;
}
```