

CA110 Space API

David Gray
Version 0.1

26th February 2015

Contents

1 HTTP Requests	2
2 3-D Coordinates	2
3 Responses	2
4 Discovery API	4
4.1 <code>getServers</code> request	4
5 Authentication API	5
5.1 <code>register</code> request	5
5.2 <code>authenticate</code> request	5
5.3 <code>version</code> request	5
6 Trade API	6
7 Game API	7
7.1 <code>Connection Handshake</code>	7
7.2 <code>Chatting</code>	9

1 HTTP Requests

For the HTTP/JSON APIs, all requests use HTTP GET.

2 3-D Coordinates

A **3-D Coordinates** object is a JSON object with the following fields:

x: A real number.

y: A real number.

z: A real number.

For example:

```
{
  "x": 42363.5374374,
  "y": 3947394796.215,
  "z": 846.26732
}
```

3 Responses

All HTTP/JSON requests respond with a JSON object containing a boolean `success` field that has one of two values:

1. `true`: The request has been successful and the JSON object will have other fields containing the results of the request; see §??.
2. `false`: The request has failed and the JSON object will have two other fields:

code: A integer value as described in Table 1.

errorText: A string giving a text message for the error.

For example,

```
{
  "success": false,
  "code": 101,
  "errorText": "Missing id parameter in http request"
}
```

Code	Reason
100	Other error
101	Missing parameter in request
102	Unknown parameter in request
103	Unknown request
104	Server not ready or busy
105	Not players turn
106	Authentication failure
107	Sender not the originator of chat message
108	Unknown player
	...

Table 1: Error Codes

4 Discovery API

4.1 getServers request

4.1.1 Parameters

None

4.1.2 Response fields

addresses: A JSON object containing the URLs of the three servers:

```
authServer : URL string
gameServer : URL string
tradeServer : URL string
```

4.1.3 Semantics

The addresses of the servers to be used.

4.1.4 Example Exchange

Request: `http://???.getServers`

Response:

```
{
  "success":true,
  "addresses": {
    "authServer": "https://1.1.1.1:3000",
    "gameServer": "https://2.2.2.2:3001",
    "tradeServer": "https://2.2.2.2:3002"
  }
}
```

5 Authentication API

5.1 register request

Allows a user to register with the API.

5.2 authenticate request

Allows an existing user to obtain an authentication token.

5.3 version request

5.3.1 Parameters

None

5.3.2 Response fields

major: The major version of the API.

minor: The minor version of the API.

5.3.3 Semantics

API version information

5.3.4 Example Exchange

Request: `http://???/version`

Response:

```
{
  "success":true,
  "major":0,
  "minor":2
}
```

6 Trade API

TBD

7 Game API

The Game API is implemented over a bidirectional, stream-oriented connection¹ over which messages are transferred. Each message has a **name** and a **content**.

1. A message's name is a string that identifies the **category** of message.
2. A message's content is a JSON object².

7.1 Connection Handshake

We could use a single response message with a boolean flag, but using two different message categories should facilitate neater code. Please comment.

When a connection is established³ the client must send a **start** message and *must not send any further message until it receives an **accepted** message from the server*. If the sever returns a **rejected** messages then the client should terminate.

7.1.1 start message

The JSON object for a **start** message has the following fields:

token: The user's authentication token.

For example:

```
{
  "token": "98786vs8g5bsg875w6g57gdg"
}
```

¹The connection must support the transfer of JSON objects belonging to different categories. For example, `engine.io.protocol` over TCP is a suitable protocol.

²Primitive JSON types and arrays cannot be used as a message's content.

³`engine.io.protocol` has its own handshake protocol used when a TCP connection is established.

7.1.2 accepted message

The JSON object for an **accepted** message has the following fields:

major: The major version of the API.
minor: The minor version of the API.
position: A 3-D Coordinates object.
direction: A 3-D Coordinates object.

For example:

```
{
  "major":0,
  "minor":2
  "position": {
    "x":42363.5374374,
    "y":3947394796.215,
    "z":846.26732
  },
  "direction": {
    "x":1.457,
    "y":0.525,
    "z":0.2546
  }
}
```

7.1.3 rejected message

The JSON object for an **rejected** message has the following fields:

code: A integer value as described in Table 1.
errorText: A string giving a text message for the error.

For example:

```
{
  "code":106,
  "errorText":"Expired Token"
}
```


7.2 Chatting

I have made some changes to chat. In particular, I'm proposing that chat messages are not acknowledged, unless there is an error and that the name of the originator of messages is included by the originator. Please comment.

There are two categories of chat message:

1. **gchat** - global chat messages.
2. **pchat** - private chat messages.

Both categories of chat message are sent from an originator's client to the server and from the server to the recipient(s). The server does not acknowledge chat messages unless there is an error, in which case, the server sends a **failedChat** message to the originator.

Once a connection has been accepted, the client may send chat messages to the server and should be prepared to accept chat message from the server.

7.2.1 gchat message

The JSON object for a **gchat** message has the following fields:

id : Unique ID for this message.

originator: The originator's name.

errorText: The chat text.

For example:

```
{
  "id":3478,
  "originator":"Master Yoda",
  "text":"Welcome to Dagobah"
}
```

7.2.2 pchat message

The JSON object for a **pchat** message has the following fields:

id : Unique ID for this message.

originator: The originator's name.

recipient: The recipient's name.

text: The chat text.

For example:

```
{
  "id":3479,
  "originator":"Master Yoda",
  "recipient":"Hans Solo",
  "text":"Welcome to Dagobah"
}
```

7.2.3 chatFailure message

The JSON object for a **chatFailure** message has the following fields:

code: A integer value as described in Table 1.

message: A string giving a text message for the error.

original: A copy of the original chat message.

For example:

```
{
  "code":107,
  "text":"Sender not originator",
  "original": {
    "id":3479,
    "originator":"Master Yoda",
    "recipient":"Hans Solo",
    "errorText":"Welcome to Dagobah"
  }
}
```