Testing  
Here I will demonstrate that both the Web API and Discord+Twitch Bot function in their desired manner.  
Also that when presented with malicious or malformed data, no fault or breach of security occurs.

**Requestor –** A user that has performed a web request to the web api.  
Code – Indicates the highlighted text represents a small snippet of code  
**AccessToken** – Used to authenticate a **Login**  
**AuthToken** – Used to authenticate a **Bot  
SAMPLEDATA –** Indicates that this data resulted in a success  
SAMPLEDATA – Indicates that something didn’t happen as intended.

## Plan

To test functionality I will use the **cURL** command line utility. Which allows you to make web requests easily from a command line.  
A typical request looks like this curl -X GET 'https://owlcoin.co.uk/webapi/Viewer' -H 'ID: 1' –d ‘{}’  
the –X indicates the request method (GET or POST)  
the ‘https://….’ Is the url to poll  
-H ‘ID: 1’ adds a header with the Key as ID and value of 1  
-d ‘{}’ contains any JSON data that is to be sent with the request

For each key type of request, I will chose a suitable example and will demonstrate the following:

1. Will respond correctly when provided with fully correct data.
2. Will respond with suitable errors when provided with
   1. Non-existent data, like an ID that doesn’t match anything
   2. Boundary data, ie password too short or not strong enough
   3. Erroneous, ie utter gibberish that no sensible user would enter
   4. Lack of authorisation

A youtube video: [**https://youtu.be/yzpHjFKZi\_c**](https://youtu.be/yzpHjFKZi_c)will be viewable, where I will show the **Test Table** commands functioning, along with showing the **Discord+Twitch Bot** side of things working.

## Test Table

|  |  |  |  |
| --- | --- | --- | --- |
| Command function | Command | Sample Data | Returned Data |
| This command will create a **Login** with the given  **UserName** / **Email** and **Password**.  Only when provided with a **UserName** / **Email** where it isn’t already attached to another **Login.**  And where the **Password** is longer than 8 characters and contains a Capital, Number and Special. | curl -X POST 'https://owlcoin.co.uk/webapi/signup' -H 'UserName: USERNAME'  -H 'Password: PASSWORD' -d '' | USERNAME = Test  PASSWORD = test | {  "Data": null,  "Code": 400,  "Message": "Bad Request, User already exists"  } |
| USERNAME = Test@ | {  "Data": null,  "Code": 400,  "Message": "Bad Request, Username is not AlphaNumeric"  } |
| USERNAME = TestAccount  PASSWORD = test | {  "Data": null,  "Code": 400,  "Message": "Bad Request, Password too short"  } |
| USERNAME = TestAccount  PASSWORD = testtesttest | {  "Data": null,  "Code": 400,  "Message": "Bad Request, Password requires at least 1 Capital, 1 Number, 1 Special"  } |
| USERNAME = TestAccount  PASSWORD = testtest1 | SAME AS ABOVE |
| USERNAME = TestAccount  PASSWORD = testtest1! | SAME AS ABOVE |
| USERNAME = TestAccount  PASSWORD = TestTest1! | {  "Data": null,  "Code": 200,  "Message": "The requested task was performed successfully"  } |
| curl -X POST 'https://owlcoin.co.uk/webapi/signup' -H 'Email: EMAIL'  -H 'Password: TestTest1!' -d '' | EMAIL = testmail | {  "Data": null,  "Code": 400,  "Message": "Bad Request, Email is not valid"  } |
| EMAIL = testmail@gmailcom | SAME AS ABOVE |
| EMAIL = testmailgmail.com | SAME AS ABOVE |
| EMAIL = testmail@gmail.com | {  "Data": null,  "Code": 200,  "Message": "The requested task was performed successfully"  } |
| This command allows us to obtain an **AccessToken** to be used for authentication purposes.  To do this we must send the **UserName** or **Email** for the account. Along with the **Password.**  If the **UserName** / **Email** correspond to a **Login** and the **Password** matches, then we will receive said **AccessToken**.  If not we will receive an error, indicating what went wrong. | curl -X POST 'https://owlcoin.co.uk/webapi/login'  -H 'UserName: USERNAME'  -H 'Password: PASSWORD' -d '' | USERNAME = TestAccount1  PASSWORD = TestTest1! | {  "Data": null,  "Code": 400,  "Message": "Bad Request, UserName does not correspond to an existing user"  } |
| USERNAME = TestAccount  PASSWORD = TestTest1 | {  "Data": null,  "Code": 400,  "Message": "Bad Request, Password does not match"  } |
| USERNAME = TestAccount  PASSWORD = TestTest1! | {  "Data": {  "UserName": "TestAccount",  "HashedPassword": null,  "AccessToken": "ACCESSTOKEN",  "Email": "",  "LastLoginDateTime": "2019-04-02T18:23:26.8010768+01:00",  "ID": 39  },  "Code": 200,  "Message": "The requested task was performed successfully"  } |
| USERNAME = testaccount  PASSWORD = TestTest1! | SAME AS ABOVE |
| This will fetch the **Login** with the given **ID**.  If an **AccessToken** is provided, that is valid for the target **Login**, extra information such as the **Email** will be returned inside of the object. Otherwise said information will be withheld for privacy purposes. | curl -X GET 'https://owlcoin.co.uk/webapi/login'  -H 'ID: ID' | ID = -1 | {  "Data": null,  "Code": 400,  "Message": "Bad Request, ID does not match an existing object"  } |
| ID = dd | {  "Data": null,  "Code": 400,  "Message": "Bad Request, Malformed ID"  } |
| ID = 39 | {  "Data": {  "UserName": "TestAccount",  "HashedPassword": null,  "AccessToken": null,  "Email": null,  "LastLoginDateTime": "2019-04-02T18:23:26",  "ID": 39  },  "Code": 200,  "Message": "The requested task was performed successfully"  } |
| curl -X GET 'https://owlcoin.co.uk/webapi/login'  -H 'ID: 39'  -H 'AccessToken:ACCESSTOKEN' | ACCESSTOKEN: thdfyui | {  "Data": {  "UserName": "TestAccount",  "HashedPassword": null,  "AccessToken": null,  "Email": null,  "LastLoginDateTime": "2019-04-02T18:23:26",  "ID": 39  },  "Code": 400,  "Message": "Bad Request, AccessToken doesnt match"  } |
| ACCESSTOKEN: (A Valid **AccessToken** for this **Login**) | {  "Data": {  "UserName": "TestAccount",  "HashedPassword": null,  "AccessToken": null,  "Email": "othertestmail@gmail.com",  "LastLoginDateTime": "2019-04-02T18:23:26",  "ID": 39  },  "Code": 200,  "Message": "The requested task was performed successfully"  } |
| With a valid **AccessToken** we are now able to create a new **Currency** for this **Login**.  All we need do is authenticate ourselves, and a **Currency** will be created with the default **Command Configuration**. | curl -X POST 'https://owlcoin.co.uk/webapi/currency' -H 'LoginID: LOGINID'  -H 'AccessToken: ACCESSTOKEN' -d '' | LOGINID = -1  ACCESSTOKEN = dfgdfg | {  "Data": null,  "Code": 400,  "Message": "Bad Request, ID does not correspond to an existing user"  } |
| LOGINID = sdf  ACCESSTOKEN = dffgffg | {  "Data": null,  "Code": 400,  "Message": "Bad Request, Malformed LoginID"  } |
| LOGINID = 39  ACCESSTOKEN = ffgdffsd | {  "Data": null,  "Code": 400,  "Message": "Bad Request, AccessToken is invalid"  } |
| LOGINID = 39  ACCESSTOKEN = (A Valid **AccessToken** for this **Login**) | {  "Data": {  "OwnerLogin": null,  "LoginConfig": null,  "CommandConfig": { COMMAND CONFIGURATION },  "ID": 237  },  "Code": 200,  "Message": "The requested task was performed successfully"  } |
| We can look at the **Currencies** tied to our **Login** by performing this command. Which will return a list, containing all of our **Currencies**.  Where we receive “Unknown Outcome, It is not known if the LoginID matches an object”,  This is because we do not check if the **LoginID** matches a **Login**. Instead we just return all **Currencies** where the **OwnerID** is the **LoginID**.  The **LoginConfig** will always be **NULL** unless it is fetched by a **SuperBot**. | curl -X GET 'https://owlcoin.co.uk/webapi/currency' -H 'LoginID: LOGINID' | LOGINID = -1 | {  "Data": [],  "Code": 200,  "Message": "Unknown Outcome, It is not known if the LoginID matches an object"  } |
| LOGINID = sdfsdf | {  "Data": null,  "Code": 400,  "Message": "Bad Request, Malformed LoginID"  } |
| LOGINID = 39 | {  "Data": [  {  "OwnerLogin": null,  "LoginConfig": null,  "CommandConfig": { COMMAND CONFIGURATION },  "ID": 237  },  {  "OwnerLogin": null,  "LoginConfig": null,  "CommandConfig": { COMMAND CONFIGURATION },  "ID": 238  }  ],  "Code": 200,  "Message": "Unknown Outcome, It is not known if the LoginID matches an object"  } |
| Now we have a **Currency** for our viewer, we can create a **Bot** to use on it.  To do this we need to provide our authentication details for the **Login.**  Upon creation we will receive the **Bot**s details along with the **Access** and **Refresh** **Tokens.**  Which will be used for authenticating the **Bot** in future.  If desired the **BotName** header can be included, to make the **Bot** easier to pick out from a list of **Bots.** | curl -X POST 'https://owlcoin.co.uk/webapi/bot'  -H 'LoginID: LOGINID'  -H 'AccessToken: ACCESSTOKEN' -d '' | LOGINID = -1  ACCESSTOKEN = rfgs | {  "Data": null,  "Code": 400,  "Message": "Bad Request, ID does not correspond to an existing user"  } |
| LOGINID = fsdfsd  ACCESSTOKEN = fsdf | {  "Data": null,  "Code": 400,  "Message": "Bad Request, Malformed LoginID"  } |
| LOGINID = 39  ACCESSTOKEN = fdsf | {  "Data": null,  "Code": 400,  "Message": "Bad Request, AccessToken is invalid"  } |
| LOGINID = 39  ACCESSTOKEN = (A Valid **AccessToken** for this **Login**) | {  "Data": {  "Currency": null,  "AccessToken": "ACCESSTOKEN",  "RefreshToken": "REFRESHTOKEN",  "BotName": "No Name Given",  "TokenRefreshDateTime": "2019-04-03T16:51:30",  "OwnerLogin": null,  "IsSuperBot": false,  "ID": 88  },  "Code": 200,  "Message": "The requested task was performed successfully"  } |
| Once we have a **Bot**, to make it useful we need to bind it to a **Currency**.  To do this, any **Currency** owner may provide their authorisation details along with the target **Bot**s **ID** and the **CurrencyID** of the **Currency** they want it to bind too.  This will then allow the bot to edit said **Currency**; when required authorisation is provided. | curl -X POST 'https://owlcoin.co.uk/webapi/bot'  -H 'LoginID: 39'  -H 'AccessToken: ACCESSTOKEN'  -H 'BotID: BOTID'  -H 'CurrencyID: CURRENCYID' -d '' | BOTID = 1  CURRENCYID = 1  ACCESSTOKEN = fsdfds | {  "Data": null,  "Code": 400,  "Message": " Bad Request, BotID doesnt match any bot"  } |
| BOTID = sdfsdf  CURRENCYID = 1  ACCESSTOKEN = faddfs | {  "Data": null,  "Code": 400,  "Message": "Bad Request, Malformed BotID"  } |
| BOTID = 83  CURRENCYID = 1  ACCESSTOKEN = fsdfds | {  "Data": null,  "Code": 400,  "Message": "Bad Request, Bot is already bound to a currency"  } |
| BOTID = 88  CURRENCYID = 1  ACCESSTOKEN = dfdsf | {  "Data": null,  "Code": 400,  "Message": "Bad Request, AccessToken is not allowed to edit that currency"  } |
| BOTID = 88  CURRENCYID = ffsdf  ACCESSTOKEN = sdfsd | {  "Data": null,  "Code": 400,  "Message": "Bad Request, Malformed CurrencyID"  } |
| BOTID = 88  CURRENCYID = 237  ACCESSTOKEN = fsdfsd | {  "Data": null,  "Code": 400,  "Message": "Bad Request, AccessToken is invalid"  } |
| BOTID = 88  CURRENCYID = 237  ACCESSTOKEN = (A Valid **AccessToken** for this **Login**) | {  "Data": null,  "Code": 200,  "Message": "The requested task was performed successfully"  } |
| Before we can use this **Bot** to modify this **Currency**, we must acquire a valid **AuthToken**. To do this, we will send the **RefreshToken** that we acquired when creating the **Bot.**  Upon refresh, we will receive a new **Refresh** and **Access** **Token** along with details about the **Bot** that we have refreshed. | curl -X POST 'https://owlcoin.co.uk/webapi/bot' -H 'BotID: 88'  -H 'RefreshToken: REFRESHTOKEN'  -d '' | BOTID = fsdfsd  REFRESHTOKEN = fsdffds | {  "Data": null,  "Code": 400,  "Message": "Bad Request, Malformed ID"  } |
| BOTID = 88  REFRESHTOKEN = sdfsdf | {  "Data": null,  "Code": 400,  "Message": "Bad Request, Refresh Token is not valid"  } |
| BOTID = 88  REFRESHTOKEN = (A Valid **RefreshToken** for this **Bot**) | {  "Data": {  "Currency": {  "OwnerLogin": {  OWNERDETAILS  "ID": 39  },  "LoginConfig": null,  "CommandConfig": { COMMAND CONFIGURATION },  "ID": 237  },  "AccessToken": "ACCESSTOKEN",  "RefreshToken": "REFRESHTOKEN",  "BotName": "No Name Given",  "TokenRefreshDateTime": "2019-04-03T18:02:30.8611651+01:00",  "OwnerLogin": {  OWNERDETAILS  "ID": 39  },  "IsSuperBot": false,  "ID": 88  },  "Code": 200,  "Message": "The requested task was performed successfully"  } |
| Now we have **Bot** authorisation details, we can create a **Viewer** for the **Currency**.  We do this by providing the authorisation details and the **Twitch** and/or **Discord** **ID**.  The provided **ID** will then become the initial method of identifying an account. | curl -X POST 'https://owlcoin.co.uk/webapi/viewer'  -H 'BotID: 88'  -H 'AuthToken: AUTHTOKEN'  -H 'TwitchID: TWITCHID'  -H 'CurrencyID: 237' -d '' | TWITCHID = fsdfsd  AUTHTOKEN = dfasd | {  "Data": null,  "Code": 400,  "Message": "Bad Request, TwitchID contains invalid characters"  } |
| TWITCHID = 1234  AUTHTOKEN = fsdfsd | {  "Data": null,  "Code": 400,  "Message": "Bad Request, AuthToken is invalid for that Bot"  } |
| TWITCHID = 1234  AUTHTOKEN = (A Valid **AuthToken** for this **Bot**) | {  "Data": null,  "Code": 200,  "Message": "The requested task was performed successfully"  } |
| Now that the **Currency** has a **Viewer**, we can view a list of all of its **Viewers**.  We do this by providing the **ID** of the **Currency** that we wish to see the **Viewers** of. | curl -X GET 'https://owlcoin.co.uk/webapi/viewer'  -H 'CurrencyID: CURRENCYID' | CURRENCYID = -1 | {  "Data": null,  "Code": 400,  "Message": "Bad Request, CurrencyID does not match an existing object"  } |
| CURRENCYID = fdsfds | {  "Data": null,  "Code": 400,  "Message": "Bad Request, Malformed CurrencyID"  } |
| CURRENCYID = 237 | {  "Data": [  {  "Balance": 250,  "WatchTime": 0,  "TwitchID": "1234",  "DiscordID": "",  "Currency": null,  "LiveNotifcations": false,  "DontReward": false,  "ID": 1747  }  ],  "Code": 200,  "Message": "The requested task was performed successfully"  } |
| Now we have a **Viewer** we can increase or decrease their balance.  To do this, we indicate wether we are to increase (+) or decrease (-), the value of the change and the **ID** of the **Viewer**. We also are required to provide authentication details, to be allowed to make this change. | curl -X POST 'https://owlcoin.co.uk/webapi/viewer'  -H 'BotID: 88'  -H 'AuthToken: AUTHTOKEN'  -H 'ID: ID'  -H 'Operator: OPERATOR'  -H 'Value: VALUE' -d '' | ID = dsf  OPERATOR = \*  VALUE = gdfg  AUTHTOKEN = fsdfds | {  "Data": null,  "Code": 400,  "Message": "Bad Request, Malformed ID and/or Value"  } |
| ID = 1747  OPERATOR = \*  VALUE = gdfg  AUTHTOKEN = fsdf | SAME AS ABOVE |
| ID = 1747  OPERATOR = \*  VALUE = 1000  AUTHTOKEN = sdsa | {  "Data": null,  "Code": 400,  "Message": "Bad Request, Operator must be + or -"  } |
| ID = 1747  OPERATOR = +  VALUE = 1000  AUTHTOKEN = fsdf | {  "Data": null,  "Code": 400,  "Message": "Bad Request, AuthToken is invalid for that Bot"  } |
| ID = 1747  OPERATOR = +  VALUE = 1000  AUTHTOKEN = (A Valid **AuthToken** for this **Bot**) | {  "Data": null,  "Code": 200,  "Message": "The requested task was performed successfully"  } |
|  |  |  |  |
| If we desire to view a specific **Viewer**. We can use their unique **ID** to identify them.  This comes in useful to check if the balance was adjusted (above). | curl -X GET 'https://owlcoin.co.uk/webapi/viewer'  -H 'ID: ID' | ID = -1 | {  "Data": null,  "Code": 400,  "Message": "Bad Request, ID does not match an existing object"  } |
| ID = fsdfsdf | {  "Data": null,  "Code": 400,  "Message": "Bad Request, Malformed ID"  } |
| ID = 1747 | {  "Data": {  "Balance": 1250,  "WatchTime": 10,  "TwitchID": "1234",  "DiscordID": "",  "Currency": {  "OwnerLogin": {  OWNERDETAILS  },  "LoginConfig": null,  "CommandConfig": {COMMAND CONFIGURATION}  "ID": 237  },  "LiveNotifcations": false,  "DontReward": false,  "ID": 1747  },  "Code": 200,  "Message": "The requested task was performed successfully" } |
| Instead of incrementing a specific **Viewers** balance, we can increment the balance and/or watchtime for all **Viewers** with a **Twitch** or **Discord** **ID** that appears in our **JSON** and in the given **Currency**.  This allows us to update many users at once, while only having to make a single request to the server.  If no **Twitch** or **Discord ID**s are provided in the **JSON**, we still report a success. As we did successfully increment all given **ID**s. | curl -X POST 'https://owlcoin.co.uk/webapi/viewer'  -H 'BotID: 88'  -H 'AuthToken: AUTHTOKEN'  -H 'WatchTimeIncrement: INCREMENT'  -H 'CurrencyID: 237'  -d 'JSON' | INCREMENT = fgsd  JSON = {fsd}]  AUTHTOKEN = sdfsd | {  "Data": null,  "Code": 500,  "Message": "Internal Server Error"  } |
| INCREMENT = fgsd  JSON = {}  AUTHTOKEN = sdfsd | {  "Data": null,  "Code": 400,  "Message": "Bad Request, AuthToken is invalid for that Bot"  } |
| INCREMENT = fgsd  JSON = {}  AUTHTOKEN = (A Valid **AuthToken** for this **Bot**) | {  "Data": null,  "Code": 400,  "Message": "Bad Request, Malformed WatchTimeIncrement"  } |
| INCREMENT = 10  JSON = {}  AUTHTOKEN = (A Valid **AuthToken** for this **Bot**) | {  "Data": null,  "Code": 200,  "Message": "The requested task was performed successfully"  } |
| INCREMENT = 10  JSON = {"TwitchIDs":["1234"]}  AUTHTOKEN = (A Valid **AuthToken** for this **Bot**) | {  "Data": null,  "Code": 200,  "Message": "The requested task was performed successfully"  } |
| We can also change other details about the **Viewer**, such as the **Discord** and **Twitch ID**, the Booleans determining if they should receive **LiveNotifcations** or if they shouldn’t receive rewards (**Don’t Reward**).  To do this, we simply provide the new state for the value, along with authentication details.  This is particularly useful if a **Viewer** changes the **Discord** or **Twitch** account they are to be bound too. | curl -X POST 'https://owlcoin.co.uk/webapi/viewer'  -H 'BotID: 88'  -H 'AuthToken: AUTHTOKEN'  -H 'ID: 1747'  -H 'DontReward: DONTREWARD'  -H 'DiscordID: DISCORDID'  -d '' | DONTREWARD = fsdf  DISCORDID = fsdf  AUTHTOKEN = fsd | {  "Data": null,  "Code": 400,  "Message": "Bad Request, DiscordID contains invalid characters"  } |
| DONTREWARD = fsdf  DISCORDID = 7896  AUTHTOKEN = fsd | {  "Data": null,  "Code": 400,  "Message": "Bad Request, AuthToken is invalid for that Bot"  } |
| DONTREWARD = fdsf  DISCORDID = 7896  AUTHTOKEN = (A Valid **AuthToken** for this **Bot**)  This code set the **DontReward** of the **Viewer** to false, when it should’ve thrown an error. If the **DontReward** is anything bar True, it will do this. | {  "Data": null,  "Code": 200,  "Message": "The requested task was performed successfully"  } |
| DONTREWARD = True  DISCORDID = 7896  AUTHTOKEN = (A Valid **AuthToken** for this **Bot**) | {  "Data": null,  "Code": 200,  "Message": "The requested task was performed successfully" } |