WebSocket Indicative Quotes API

K33 Markets WebSocket Indicative Quotes OTC API

Overview

The document describes the functionality of the OTC API. This API allows access to K33 Markets' market functions. The examples below are subject to change based on individual deployment.

All calls are made via websocket connection.

Contents

Creating a Web Socket

In order to send and receive JSON messages, you will need to create a Web Socket in your preferred programming language. Below is the sample code for doing so in JavaScript (surrounding code is included as it will be helpful when describing how to log in in the next section).

Web Socket creation in Javascript

```
var websocket = new WebSocket(address);
var serverNonceValue = '';
var cnOnce Value = '';
websocket.onopen = onOpen;
websocket.onclose = onClose;
websocket.onmessage = onMessage;
websocket.onerror = onError;
```

Market

There are currently 5 message types you may want to handle in the onMessage event handler in order to process JSON responses from the API. They are 'generalException', 'serverNonce', 'loginResult', 'subscribeResponse' and 'subscription'. We will provide an example in the Login section below.

1. Login

In order to log into the API, you will make use of the serverNonceValue variable that is shown above as well as the onMessage event handler. Before being able to log in, you will need to use the nonce value that is sent by the API server. You will get a message containing the server nonce value at the start of the connection. (case 'serverNonce').

onMessage event handler in Javascript

```
function onMessage(evt) {
   var jsonMessage = JSON.parse(evt.data);
   switch(jsonMessage.messageType){
      case 'generalException':
           // code
           serverNonceValue = jsonMessage.serverNonce;
           break;
      case 'serverNonce':  //unique value with each connection!
           serverNonceValue = jsonMessage.serverNonce;
           break;
      case 'loginResult':
         //code
```

Now that you have the server nonce value, you can create a client nonce value and tokenHash. (as shown in example code below). For the client nonce value, the example code makes use of Date.getTime(), but there are many ways to create this value. As mentioned above, the serverNonceValue was set at the start of the web socket connection.

JavaScript code for creating a login request

```
const todaysDate = new Date();
const cnOnceValue = todaysDate.getTime().toString();
const serverNonceValue = 0; //set elsewhere in the code
const userName = ""; // set your username
const password = ""; // set your password
const token = serverNonceValue + cnOnceValue + password;
const tokenHash = Crypto.createHash("sha256").update(token).digest("hex");

const loginRequest = {
   route: "login",
   userName: userName,
   cnOnce: cnOnceValue,
   token: tokenHash,
};

websocket.send(JSON.stringify(loginRequest));
```

JSON Request Parameters and Descriptions

Parameter	Type	Description	Example
route	string	Type of message sent	'login'
userName	string	Your username	'assignedUserName'
cnOnce	string	Unique identifier for client	'1605040844949'
token	string	Computed token hash	'4b463ab499a732d0cf7d18f38f8bf037de0052c8e79fcac241b9321629e6924e'

Example JSON request

```
{
  "route": "login",
  "userName": "assignedUserName",
  "cnOnce": "1605060344539",
  "token":"d2d798bdc995ac07cf795c4da8e1d83a2449e2910bb606d8c75f654b47840942"
}
```

JSON Response Parameters and Descriptions

Parameter	Type	Description	Example
messageType	string	Type of message	'loginResult'
success	boolean	Whether login request was successful	true
errorDescription	string	Description of error if one occurred	' ' or 'Invalid login'

newNonce	string	New nonce sent from server	‘ ’ or 1605061700976 *used for the next login attempt if unsuccessful
clientIdent	String	Universally unique identifier returned in login response.	1605033314105

Example JSON response for successful login request

```
{
  "messageType": "loginResult",
  "success": true,
  "errorDescription": "",
  "newNonce": ""
  "clientIdent": "1605033314105",
}
```

Example JSON response for unsuccessful login request

```
{
  "messageType": "loginResult",
  "success": false,
  "errorDescription": "Invalid login",
  "newNonce": 1605061700976
}
```

*Note On any login failure the server will transmit a new nonce value ‘newNonce’ for the next login attempt

errorDescription return values
‘Invalid login’
‘User ID does not exist’

2. Subscribe

Get the prices of a symbol, it is a one-time request, then price updates will arrive through the "subscription" message.

Subscription by symbol

To subscribe to a specific symbol, you must send the symbol in the key field.

JSON Request Parameters and Descriptions

Parameter	Type	Description	Example
route	string	Type of message sent	‘subscribe‘
channel	string	Subscription type (quote)	‘quote‘
key	string	Subscription symbol	‘BTC/USD‘
clientIdent	string	Universally unique identifier returned in login response.	‘1605033314105’

Example JSON request

{
"route": "subscribe",
"clientIdent": "1605033314105",
"channel": "quote",
"key": "BTC/USD",
};

Subscription to all symbols

To subscribe to all the available symbols, you only need to send the channel.

JSON Request Parameters and Descriptions

Parameter	Type	Description	Example
route	string	Type of message sent	‘subscribe‘
channel	string	Subscription type (quote)	‘quote‘
clientIdent	string	Universally unique identifier returned in login response.	‘1605033314105’

Example JSON request

{
"route": "subscribe",
"clientIdent": "1605033314105",
"channel": "quote"
};

JSON Response Parameters and Descriptions

Parameter	Type	Description	Example
messageType	string	Type of message	‘subscribeResponse‘

success	boolean	Whether request was successful	true
channel	string	Channel processed	‘quote‘
key	string	key processed	‘BTC/USD‘

Example JSON Response

{
"messageType": 'subscribeResponse',
"success": true,
"channel": "Quote",
"key": "BTC/USD",
};

Subscription update message.

This message arrives for each price update per symbol, while the user is subscribed. You can unsubscribe

Parameter	Type	Description	Example
messageType	string	Type of message	‘subscribeResponse‘
success	boolean	Whether request was successful	true
channel	string	Channel processed	‘quote‘
key	string	key processed	‘BTC/USD‘
symbol	string	symbol info	‘BTC/USD‘
bid	number	buy side price	42600
ask	number	sell side price	42400

Example JSON Response

{
"messageType": "subscription",
"channel": "Quote",
"key": "BTC/USD",
"data": {
"symbol": "BTC/USD",
"bid": 42600,
"ask": 42400,
}
}

2. Unsubscribe

To unsubscribe to a specific symbol it is necessary to send this request. Once confirmed with the unsubscribeResponse message, no more updates will be received for the processed symbol

JSON Request Parameters and Descriptions

Parameter	Type	Description	Example
route	string	Type of message sent	‘unsubscribe‘
channel	string	Subscription type (quote)	‘quote‘
key	string	Symbol to unsubscribe	‘BTC/USD‘
clientIdent	string	Universally unique identifier returned in login response.	‘1605033314105’

Example JSON request

```
{
"route": "unsubscribe",
"clientIdent": "1605033314105",
"channel": "quote",
"key": "BTC/USD",
};
```

JSON Response Parameters and Descriptions

Parameter	Type	Description	Example
messageType	string	Type of message	‘unsubscribeResponse‘
success	boolean	Whether request was successful	true
channel	string	Channel processed	‘quote‘
key	string	key processed	‘BTC/USD‘

Example JSON Response

```
{
"messageType": 'unsubscribeResponse',
"success": true,
"channel": "Quote",
"key": "BTC/USD",
};
```