OceanEx Cryptocurrency Exchange Websocket API Specifications

Version 0.5

2020.07.01

1. Introduction

1.1. Overview

OceanEx provides Websocket APIs for API end users to access OceanEx Exchange. The functions cover subscription, trading, account info checking and so on.

This documentation sets out the New Websocket API specifications for OceanEx trading platform. It describes the content of the websocket messages request and response's headers and body, the contents of Public API messages and Private API messages, and provides detailed field descriptions.

1.2. Documentation History

Version	Date	Author	Description
0.1	March 07 2020	Technology	Initial version
0.2	June 10 2020	Technology	Update section 2.4
0.3	June 11 2020	Technology	Add section 1.4. Add a websocket client sample.
0.4	June 15 2020	Technology	Correct section 2.1 websocket endpoint. Update section 5 for support link.
0.5	July 01 2020	Technology	Correct section 2.1 and section 2.4 websocket endpoint to `wss://ws.oceanex.pro/ws/v1` after system upgrade.

1.3. Protocol Features

1.3.1. Subscription API - Legacy

The Subscription API is OceanEx's legacy websocket API, which is the first version of the public websocket API. The legacy's websocket API includes 3 public subscription messages. They are *Tickers*, *Order Book*, *Trades*. For learning how to use the protocol, please refer to https://api.oceanex.pro/doc/v1/#websockets.

1.3.2. Non-subscription API - New

The Non-subscription API is the NEW query-based websocket API. It provides the same level of OceanEx REST API. It allows users perform operations such as new orders, order cancel, order status query and order history query. The key advantage of the API is that it is much faster than REST API. On the other side, users have to maintain the websocket session. An API Rate-Limit policy is introduced to keep the OceanEx Websocket server under health conditions and prevented from attacks.
** Note** The following contents of this specification gives details about how to use the new websocket API. The Subscription API user guide as mentioned on 1.3.1 is available on OceanEx API guide website. We will **exclude** legacy API information from here.

1.4. OceanEx Websocket Client Sample

To demonstrate how the new websocket API works, an OceanEx websocket client sample can be downloaded from here. The sample program includes all the websocket's features. The details of each feature usage will be explained in rest sections of this doc.

2. Session

2.1. Build connection

The new websocket api server endpoint is

```
wss://ws.oceanex.pro/ws/v1
```

Once connected, the server will respond with the welcome messages to confirm a successful connection. The welcome message will carry an integer to indicate the heartbeat interval. For example:

```
{"type":"welcome", "message": 60}
```

Once a connection request is rejected, the server will also respond with an error message to indicate the reason for rejection. For example:

```
{"type":"disconnect", "reason":"unauthorized", "reconnect":false}
```

2.2. Heartbeat

Once connected successfully, users have to maintain connection between their program and oceanex websocket server. That means users' programs need to keep responding with the pong message when received a ping message from exchange. Users should not keep sending pong messages. As a fact, users should send pong messages when need to respond to ping messages only.

The heartbeat interval is default with 60 seconds. Users could get it from the welcome message body. If users' program did not respond to a continuous 3 ping messages, OceanEx websocket server will disconnect the user websocket application. Then, the user has to initialize the connection start over again for reconnection.

2.3. Logon

The Authentication message is the logon message for OceanEx websocket API. It is ONLY required for private API. The logon messages parameters ask users to provide *uid* and *apikid*.

The logon message also require users encoded their private api key with SHA256 into the *signature* field in the body,

Once the logon message is sent and authenticated. Users are permitted to send private API functions. Otherwise, the session disconnection will occur due to any un-permitted private API requests. The Authentication message is only needed to be sent once during the session.

2.4. Example for connection and maintaining session

```
import json
import websocket
import uuid

def on_message(ws, message):
    message = json.loads(message)
    print("*** Receive Message as {}".format(message))
    if "type" in message and message['type'] == "ping":
        print("return pong")
w        ws.send(json.dumps({ "command": "pong" }))
    elif "handler" in message['identifier'] and message['identifier']['handler'] ==
"ToolHandler" and message['message']['action'] == "timestamp":
        on_timestamp(message)

def on_error(ws, error):
    print(error)

def on_close(ws):
    print("### closed ###")

def on_timestamp(data):
    print('received timestamp info');

def on_open(ws):
    print("######## WS Opened ########")

    args = {}
    data = {
        "identifier": {"handler": "ToolHandler"},
        "command": "message",
        "data": {
            "action": "timestamp",
            "uuid": str(uuid.uuid1()),
            "args": args
        }
    }
    ws.send(json.dumps(data))
```

```
if __name__ == "__main__":
    ws = websocket.WebSocketApp("wss://ws.oceanex.pro/ws/v1",
                                on_message = on_message,
                                on_error = on_error,
                                on_close = on_close)
    ws.on_open = on_open
    ws.run_forever()
```

3. Session Message

3.1. Outbound Messages (User to OceanEx)

3.1.1. Pong message

The pong message is used to respond to ping messages.

Field	Data Type	Description	Value
command	string	Type of request	“pong”

Example:

```
ws.send(json.dumps({ "command": "pong" }))
```

3.1.2. Authentication message

Once authenticated, users can send private API messages. The authentication is done through an authentication message. Inside the authentication message, the *signature* is a mandatory message field encoded with the user's private API key. The authentication message body is explained as below.

Field	**Data Type**	**Description**	**Value**
identifier	Json String	Type of handler	{"handler": "AuthHandler"}
command	String	Type of request	"message"
Data	Json String	Message body	See blow

Data

Field	**Data Type**	**Description**	**Value**
action	String	Type of action	"login"
uuid	String	unique request id	User specific
args	Json String	Data Arguments	{}
header	Json String	Header	See blow

Header

Field	**Data Type**	**Description**	**Value**
uid	String	OceanEx UID	User specific
api_key	String	OceanEx API Key ID	User specific
signature	String	SHA 256 encode string	See example
nonce	Int	Timestamp	Timestamp in sec
verb	String	Function Type	"GET"

path	String	Endpoint path	"/ws/v1"

```
# user info
userid = "ID4E5XXXXXX"
apikid = "KIDB7XXXXXX"

    def _generate_signature(self, user_private_key, cur):
        private_key = serialization.load_pem_private_key(
            user_private_key.encode('ascii'),
            password=None,
            backend=default_backend()
        )
        path = "GET|/ws/v1|{}|".format(cur)
        sign = private_key.sign(
            path.encode('utf-8'),
            padding.PKCS1v15(),
            hashes.SHA256()
        )
        signature = base64.b64encode(sign).decode()
        return signature

###################### sender function ##################
    def authenticate(self, args=None):
        logger.info("Send authentication")
        if self.uid is None or self.kid is None:
            print("Please set uid and kid first")
            return
        with open("key.pem", "r") as private_file:
            user_private_key = private_file.read()
        cur = int(time.time() * 1000)
        auth = {
            "identifier": {"handler": "AuthHandler"},
            "command": "message",
            "data": {
                "action": "login",
                "uuid": str(uuid.uuid1()),
                "args": {},
                "header": {
                    "uid": self.uid,
                    "api_key": self.kid,
                    "signature": self._generate_signature(user_private_key,
```

```
cur),
                    "nonce": cur,
                    "verb": "GET",
                    "path": "/ws/v1"
                }
            }
        }
        self.send_json(auth)

##### main #####
ws.authenticate()
```

Response Example:

```
{
  "identifier": {
    "handler": "AuthHandler"
  },
  "message": {
    "action": "login",
    "code": 0,
    "data": {
      "scopes": {
        "query": true,
        "sell/buy": true,
        "withdraw": true
      }
    },
    "uuid": "49f8254a-61bf-11ea-b613-7aa0b0f73a47"
  }
}
```

3.2. Inbound Messages (OceanEx to User)

3.2.1. Welcome message

The welcome message is always sent as the first message to the user upon a successful websocket connection.

Field	Data Type	Description	Value
Message	Int	Time interval receiving ping message	Exchange specific
Type	String	Message type	“weclome”

Example:

```
{
  "message": 60,
  "type": "welcome"
}
```

3.2.2. Ping message

The ping message is a heartbeat message to maintain session connectivity. Once users receive it, users should respond with a pong message right away and no delay for over 3 ping messages.

Field	Data Type	Description	Value
Message	Int	TimeStamp in sec	Exchange specific
Type	String	Message type	“ping”

Example:

```
{
  "message": 1583729664,
```

```
  "type": "ping"
}
```

4. Application Message

4.1. Public API

The OceanEx Websocket Public API doesn't require user logon and the request can work without logon. Since the protocol is query-based, there is no subscription-like Public API. Users get one response per one request.

4.1.1. Timestamp

The timestamp API is used to get Exchange System Timestamp.

Field	**Data Type**	**Description**	**Value**
identifier	JSON string	Handler type	{"handler": "ToolHandler"}
command	String	Message type	“message”
data	JSON string	Body	See below

Data

Field	Data Type	Description	Value
action	String	request type	"timestamp"
uuid	String	unique request id	User specific
args	JSON string	arguments	{}

Example:

```
        data = {
            "identifier": {"handler": "ToolHandler"},
            "command": "message",
            "data": {
                "action": "timestamp",
                "uuid": str(uuid.uuid1()),
                "args": args
            }
        }
```

Response Example:

```
{
  "identifier": {
    "handler": "ToolHandler"
  },
  "message": {
    "action": "timestamp",
    "code": 0,
    "data": 1583764947,
    "uuid": "32c15ce8-6214-11ea-b613-7aa0b0f73a47"
  }
}
```

4.1.2. Markets

The market API is used to get all markets from OceanEx.

Field	Data Type	Description	Value
identifier	JSON string	Handler type	{"handler": "MarketsHandler"}
command	String	Message type	“message”
data	JSON string	Body	See below

Data

Field	Data Type	Description	Value
action	String	request type	"index"
uuid	String	unique request id	User specific
args	JSON string	arguments	{}

Example:

```
        data = {
            "identifier": {"handler": "MarketsHandler"},
            "command": "message",
            "data": {
                "action": "index",
                "uuid": str(uuid.uuid1()),
                "args": args
            }
        }
```

Response Example:

```
{
  "identifier": {
    "handler": "MarketsHandler"
  },
  "message": {
    "action": "index",
    "code": 0,
    "data": [
```

```
      {
        "amount_precision": 8,
        "ask_precision": 8,
        "bid_precision": 8,
        "display_name": "VET/USDT",
        "enabled": true,
        "group": "USDT",
        "id": "vetusdt",
        "minimum_trading_amount": "0.0",
        "name": "VET/USDT",
        "price_precision": 8,
        "usd_precision": 8
      }
    ],
    "uuid": "ce5c2134-6218-11ea-b613-7aa0b0f73a47"
  }
}
```


4.1.3. Trading Fee

The trading fee API gets all the fees' information for all markets.

Field	Data Type	Description	Value
identifier	JSON string	Handler type	{"handler": "FeeHandler"}
command	String	Message type	“message”
data	JSON string	Body	See below

Data

Field	Data Type	Description	Value
action	String	request type	"trading_fee"
uuid	String	unique request id	User specific
args	JSON string	arguments	{}

Example:

```
        data = {
            "identifier": {"handler": "FeeHandler"},
            "command": "message",
            "data": {
                "action": "trading_fee",
                "uuid": str(uuid.uuid1()),
                "args": args
            }
        }
```

Response Example:


```
{
  "identifier": {
    "handler": "FeeHandler"
  },
  "message": {
    "action": "trading_fee",
    "code": 0,
    "data": [
      {
        "ask_fee": {
          "type": "relative",
          "value": "0.001"
        },
        "bid_fee": {
          "type": "relative",
          "value": "0.001"
        },
        "market_id": "vetusdt"
      }
    ],
    "uuid": "513eae54-621a-11ea-b613-7aa0b0f73a47"
  }
}
```

4.1.4. Tickers

The ticker API returns current ticker information for multiple market trading pairs. It is not a subscription API

Field	Data Type	Description	Value
identifier	JSON string	Handler type	{"handler": "TickerHandler"}
command	String	Message type	“message”
data	JSON string	Body	See below

Data

Field	Data Type	Description	Value
action	String	request type	"index"
uuid	String	unique request id	User specific
args	JSON string	arguments	{} for all markets. Or use an array to limit markets. {“market_ids”: [“ocevet”,”vetusdt”]}

Example:

```
data = {
            "identifier": {"handler": "TickerHandler"},
            "command": "message",
            "data": {
                "action": "index", "uuid": str(uuid.uuid1()),
                "args": args
            }
        }
```

Response Example:

```
{
  "identifier": {
    "handler": "TickerHandler"
  },
  "message": {
    "action": "index",
    "code": 0,
    "data": {
      "ethusdt": {
        "at": 1583767923,
        "base_unit": "eth",
        "buy": "0.0",
        "display_name": "ETH/USDT",
        "first": "0.0",
        "funds": "0.0",
        "h24_trend": "0.0",
        "high": "0.0",
        "last": "0.0",
        "low": "0.0",
        "name": "ETH/USDT",
        "open": "0.0",
        "position": 8,
        "quote_unit": "usdt",
        "sell": "0.0",
        "volume": "0.0"
      }
    },
    "uuid": "2016d918-621b-11ea-b613-7aa0b0f73a47"
  }
}
```


4.1.5. Order Book

The order book API is used to retrieve information for a specific market.

Field	Data Type	Description	Value
identifier	JSON string	Handler type	{"handler": "OrderBookHandler"}
command	String	Message type	“message”

data	JSON string	Body	See below

Data

Field	Data Type	Description	Value
action	String	request type	"index"
uuid	String	unique request id	User specific
args	JSON string	arguments	See below

Arguments

Field	Data Type	Description	Value	Mandatory
market_id	String	Market ID	Example: "vetusdt"	Y
level	Int	Depth to query order book	Example: 2	N
precision	Int	Pricing Precision	Example: 6	N

Example:

```
args = {'market_id': "vetusdt", 'level': 2, 'precision': 6 }
args = json.loads(json.dumps(args))

data = {
            "identifier": {"handler": "OrderBookHandler"},
            "command": "message",
            "data": {
                "action": "index",
                "uuid": str(uuid.uuid1()),
                "args": args
            }
        }
```

Response Example:

```
{
  "identifier": {
    "handler": "OrderBookHandler"
  },
  "message": {
    "action": "index",
    "code": 0,
    "data": {
      "asks": [],
      "bids": [
        [
          "0.75",
          "1000.0"
        ]
      ]
    },
    "uuid": "8eab3610-6227-11ea-b613-7aa0b0f73a47"
  }
}
```


4.1.6. K line

The k-line API is used to retrieve K-Line history data.

Field	Data Type	Description	Value
identifier	JSON string	Handler type	{"handler": "KlineHandler"}
command	String	Message type	“message”
data	JSON string	Body	See below

Data

Field	Data Type	Description	Value

action	String	request type	"index"
uuid	String	unique request id	User specific
args	JSON string	arguments	See below

Arguments

Field	**Data Type**	**Description**	**Value**	**Mandatory**
market_id	String	Market ID	Example: "vetusdt”	Y
limit	Int	Limit num of data	Example: 30	N
period	Int	period of K line data	Example: 5	N
start_time	Int	Start time in sec	Example: 1581104400	N
end_time	Int	End time in sec	Example: 1581104700	N

Example:

```
args = {'market_id': "vetusdt"], 'limit': 30, 'period': 5 }
args = json.loads(json.dumps(args))

data = {
            "identifier": {"handler": "KlineHandler"},
            "command": "message",
            "data": {
                "action": "index",
                "uuid": str(uuid.uuid1()),
                "args": args
            }
        }
```

Response Example:

```
{
  "identifier": {
    "handler": "KlineHandler"
  },
  "message": {
    "action": "index",
    "code": 0,
    "data": [
      [
        1583772540,
        1000.0,
        1000.0,
        1000.0,
        1000.0,
        0
      ]
    ],
    "uuid": "1182bed0-622a-11ea-b613-7aa0b0f73a47"
  }
}
```


4.2. Private API

OceanEx provides private APIs for sensitive functions such as trading, order status check, account information query. As mentioned in the logon section, it is mandatory to send an authentication message before sending private API messages. Once authenticated, users don't need to send an authentication message again. Disconnection will occur when users send private api messages without authentication.

4.2.1. Account Information

The account API can retrieve user's account information from exchange.

Field	Data Type	Description	Value
identifier	JSON string	Handler type	{"handler": "AccountHandler"}
command	String	Message type	“message”
data	JSON string	Body	See below

Data

Field	Data Type	Description	Value
action	String	request type	"index"
uuid	String	unique request id	User specific
args	JSON string	arguments	{}

Example:

```
  data = {
            "identifier": {"handler": "AccountHandler"},
            "command": "message",
            "data": {
                "action": "index",
                "uuid": str(uuid.uuid1()),
                "args": args
            }
        }
```

Response Example:

```
{
  "identifier": {
    "handler": "AccountHandler"
  },
  "message": {
    "action": "index",
    "code": 0,
    "data": {
      "accounts": [
```

```
        {
          "balance": "9999999247.75",
          "btc_equivalent_balance": "0.0",
          "currency": "usdt",
          "deposit": "enabled",
          "exchange": "enabled",
          "is_combination_address": false,
          "locked": "750.0",
          "markets": [],
          "type": "coin",
          "withdraw": "enabled"
        },
        {
          "balance": "9999999999.99775",
          "btc_equivalent_balance": "0.0",
          "currency": "vet",
          "deposit": "enabled",
          "exchange": "enabled",
          "is_combination_address": false,
          "locked": "0.0",
          "markets": [
            {
              "base_unit": "vet",
              "display_name": "VET/ETH",
              "quote_unit": "eth"
            },
            {
              "base_unit": "vet",
              "display_name": "VET/USDT",
              "quote_unit": "usdt"
            }
          ],
          "type": "coin",
          "withdraw": "enabled"
        }
      ]
    },
    "uuid": "809a0cf8-622d-11ea-b613-7aa0b0f73a47"
  }
}
```

4.2.2. New Order

The order api supports sending market order and limit order.

Field	Data Type	Description	Value
identifier	JSON string	Handler type	{"handler": "OrderHandler"}
command	String	Message type	“message”
data	JSON string	Body	See below

Data

Field	Data Type	Description	Value
action	String	request type	"create"
uuid	String	unique request id	User specific
args	JSON string	arguments	See below

Arguments

Field	Data Type	Description	Value	Mandatory
market	String	Market ID	Example: "vetusdt”	Y
side	String	Order Side	“Buy” or “Sell”	Y
volume	doule	Order Volume	Example 1000	N
price	doule	Order Price	Example: 0.75	N

Example:

```
args = {'market': "vetusdt", 'side': "buy", 'price': 0.75, 'volume': 1000,
'ord_type': "limit"}
args = json.loads(json.dumps(args))

data = {
            "identifier": {"handler": "OrderHandler"},
            "command": "message",
            "data": {
                "action": "create",
                "uuid": str(uuid.uuid1()),
                "args": args
            }
        }
```

Response Example:


```
{
  "identifier": {
    "handler": "OrderHandler"
  },
  "message": {
    "action": "create",
    "code": 0,
    "data": {
      "avg_price": "0.0",
      "created_at": "2020-03-09T18:04:13Z",
      "created_on": 1583777053,
      "executed_volume": "0.0",
      "id": 39,
      "market": "vetusdt",
      "market_display_name": "VET/USDT",
      "market_name": "VET/USDT",
      "ord_type": "limit",
      "price": "0.75",
      "remaining_volume": "1000.0",
      "side": "bid",
      "state": "wait",
      "trades_count": 0,
      "volume": "1000.0"
    },
    "uuid": "627dfa24-6230-11ea-b613-7aa0b0f73a47"
```

```
  }
}
```

4.2.3. Order Cancel

The order cancel API support cancelling single or multiple orders.

Field	Data Type	Description	Value
identifier	JSON string	Handler type	{"handler": "OrderHandler"}
command	String	Message type	“message”
data	JSON string	Body	See below

Data

Field	Data Type	Description	Value
action	String	request type	"cancel"
uuid	String	unique request id	User specific
args	JSON string	arguments	See blow

Arguments

Field	Data Type	Description	Value	Mandatory
ids	array	An array of numeric order ids	Example: [11, 23, 56]	Y

Example:

```
ids = []
ids.append(22)
ids.append(56)
args = {'ids': ids}
args = json.loads(json.dumps(args))

data = {
            "identifier": {"handler": "OrderHandler"},
            "command": "message",
            "data": {
                "action": "cancel",
                "uuid": str(uuid.uuid1()),
                "args": args
            }
}
```

Response Example:

```
{
  "identifier": {
    "handler": "OrderHandler"
  },
  "message": {
    "action": "cancel",
    "code": 0,
    "data": [
      39
    ],
    "uuid": "a9fdce32-6231-11ea-b613-7aa0b0f73a47"
  }
}
```

4.2.4. Mass Cancel

The mass cancel API provides the ability to cancel ALL the orders.

Field	Data Type	Description	Value
identifier	JSON string	Handler type	{"handler": "OrderHandler"}
command	String	Message type	“message”
data	JSON string	Body	See below

Data

Field	Data Type	Description	Value
action	String	request type	"clear"
uuid	String	unique request id	User specific
args	JSON string	arguments	See blow

Example:

```
args = {}
data = {
            "identifier": {"handler": "OrderHandler"},
            "command": "message",
            "data": {
                "action": "clear",
                "uuid": str(uuid.uuid1()),
                "args": args
            }
   }
```

Response Example:

```
{
  "identifier": {
    "handler": "OrderHandler"
  },
  "message": {
    "action": "clear",
    "code": 0,
```

```
    "data": [
      40,
      41,
      42,
      43
    ],
    "uuid": "4d213bd8-6234-11ea-b613-7aa0b0f73a47"
  }
}
```

4.2.5. List Open Orders

The open order list API provides the feature listing all the open orders for a market.

Field	**Data Type**	**Description**	**Value**
identifier	JSON string	Handler type	{"handler": "OrderHandler"}
command	String	Message type	“message”
data	JSON string	Body	See below

Data

Field	**Data Type**	**Description**	**Value**
action	String	request type	"index"
uuid	String	unique request id	User specific
args	JSON string	arguments	See blow

Arguments

Field	**Data Type**	**Description**	**Value**	**Mandatory**
market	String	Market ID	Example: "vetusdt”	Y

order_type	String	Order Type	“limit”, “market”	N
order_by	String	Display Order	“desc”, “asc”	N
page	Int	Page Num	Example: 1	N
per	Int	Num of Record in a Page	Example: 10	N

Example:

```
args = {'market': "vetusdt", 'ord_type': "limit"}
args = json.loads(json.dumps(args))

data = {
            "identifier": {"handler": "OrderHandler"},
            "command": "message",
            "data": {
                "action": "index",
                "uuid": str(uuid.uuid1()),
                "args": args
            }
 }
```

Response Example:

```
{
  "identifier": {
    "handler": "OrderHandler"
  },
  "message": {
    "action": "index",
    "code": 0,
    "data": {
      "resources": [
        {
          "avg_price": "0.0",
          "created_at": "2020-03-09T18:49:15Z",
```

```
    "created_on": 1583779755,
    "executed_volume": "0.0",
    "id": 46,
    "market": "vetusdt",
    "market_display_name": "VET/USDT",
    "market_name": "VET/USDT",
    "ord_type": "limit",
    "price": "0.75",
    "remaining_volume": "1000.0",
    "side": "bid",
    "state": "wait",
    "trades_count": 0,
    "volume": "1000.0"
  },
  {
    "avg_price": "0.0",
    "created_at": "2020-03-09T18:49:11Z",
    "created_on": 1583779751,
    "executed_volume": "0.0",
    "id": 45,
    "market": "vetusdt",
    "market_display_name": "VET/USDT",
    "market_name": "VET/USDT",
    "ord_type": "limit",
    "price": "0.75",
    "remaining_volume": "1000.0",
    "side": "bid",
    "state": "wait",
    "trades_count": 0,
    "volume": "1000.0"
  },
  {
    "avg_price": "0.0",
    "created_at": "2020-03-09T18:49:08Z",
    "created_on": 1583779748,
    "executed_volume": "0.0",
    "id": 44,
    "market": "vetusdt",
    "market_display_name": "VET/USDT",
    "market_name": "VET/USDT",
    "ord_type": "limit",
    "price": "0.75",
```

```
          "remaining_volume": "1000.0",
          "side": "bid",
          "state": "wait",
          "trades_count": 0,
          "volume": "1000.0"
        }
      ],
      "resources_count": 3
    },
    "uuid": "b0fa4670-6236-11ea-b613-7aa0b0f73a47"
  }
}
```


4.2.6. Open Order Info

The open order info is used to retrieve the particular open order's status information based on the OID given by the user. If the order is cancelled or filled, the API won't be able to find the record. To find a particular cancelled or filled order, please refer to history order info.

Field	Data Type	Description	Value
identifier	JSON string	Handler type	{"handler": "OrderHandler"}
command	String	Message type	"message"
data	JSON string	Body	See below

Data

Field	Data Type	Description	Value
action	String	request type	"show"
uuid	String	unique request id	User specific
args	JSON string	arguments	See blow

Arguments

Field	Data Type	Description	Value	Mandatory
id	Int	Order ID	Example: 11	Y

Example:

```
      args = {'id': args[0]}
      args = json.loads(json.dumps(args))

      data = {
            "identifier": {"handler": "OrderHandler"},
            "command": "message",
            "data": {
                "action": "show",
                "uuid": str(uuid.uuid1()),
                "args": args
            }
        }
```

Response Example:


```
{
  "identifier": {
    "handler": "OrderHandler"
  },
  "message": {
    "action": "show",
    "code": 0,
    "data": {
      "avg_price": "0.0",
      "created_at": "2020-03-09T19:00:48Z",
      "created_on": 1583780448,
      "executed_volume": "0.0",
      "id": 47,
      "market": "vetusdt",
      "market_display_name": "VET/USDT",
      "market_name": "VET/USDT",
      "ord_type": "limit",
      "price": "0.75",
```

```
      "remaining_volume": "1000.0",
      "side": "bid",
      "state": "wait",
      "trades": [],
      "trades_count": 0,
      "volume": "1000.0"
    },
    "uuid": "4e3cd03c-6238-11ea-b613-7aa0b0f73a47"
  }
}
```


4.2.7. History Order Info

The history order info API is used to retrieve a particular order that is in ***closed*** status. Those orders are either cancelled or filled.

Field	**Data Type**	**Description**	**Value**
identifier	JSON string	Handler type	{"handler": "OrderHistoryHandler"}
command	String	Message type	“message”
data	JSON string	Body	See below

Data

Field	**Data Type**	**Description**	**Value**
action	String	request type	"show"
uuid	String	unique request id	User specific
args	JSON string	arguments	See blow

Arguments

Field	Data Type	Description	Value	Mandatory
id	Int	Order ID	Example: 11	Y

Example:

```
args = {'id': args[0]}
args = json.loads(json.dumps(args))

data = {
            "identifier": {"handler": "OrderHistoryHandler"},
            "command": "message",
            "data": {
                "action": "show",
                "uuid": str(uuid.uuid1()),
                "args": args
            }
        }
```

Response Example:

```
  "identifier": {
    "handler": "OrderHistoryHandler"
  },
  "message": {
    "action": "show",
    "code": 0,
    "data": {
      "avg_price": "1000.0",
      "created_at": "2020-03-04T17:02:56Z",
      "created_on": 1583341376,
      "executed_volume": "0.75",
      "id": 5,
      "market": "vetusdt",
      "market_display_name": "VET/USDT",
      "market_name": "VET/USDT",
      "ord_type": "limit",
```

```
      "price": "1000.0",
      "remaining_volume": "0.0",
      "side": "ask",
      "state": "done",
      "trades": [
        {
          "ask_fee": "0.75",
          "ask_fee_currency_id": "usdt",
          "bid_fee": "0.00075",
          "bid_fee_currency_id": "vet",
          "created_at": "2020-03-04T17:02:57Z",
          "created_on": 1583341377,
          "fee": "0.001",
          "funds": "750.0",
          "id": 1,
          "market": "vetusdt",
          "market_display_name": "VET/USDT",
          "market_name": "VET/USDT",
          "order_id": 5,
          "price": "1000.0",
          "side": "ask",
          "volume": "0.75"
        }
      ],
      "trades_count": 1,
      "volume": "0.75"
    },
    "uuid": "dffb5632-6239-11ea-b613-7aa0b0f73a47"
  }
}
```


4.2.8. History Orders

The history orders API is similar to the open order list API. However, it only displays orders which are under *'cancel'* or *'done'* status. Open orders are excluded from this API.

Field	Data Type	Description	Value
identifier	JSON string	Handler type	{"handler": "OrderHistoryHandler"}

command	String	Message type	“message”
data	JSON string	Body	See below

Data

Field	**Data Type**	**Description**	**Value**
action	String	request type	"index"
uuid	String	unique request id	User specific
args	JSON string	arguments	See blow

Arguments

Field	**Data Type**	**Description**	**Value**	**Mandatory**
market	String	Market ID	Example: "vetusdt”	Y
state	String	Order Status	“cancel” or “done”	N
order_by	String	Display Order	“desc”, “asc”	N
page	Int	Page Num	Example: 1	N
per	Int	Num of Record in a Page	Example: 10	N
start_time	Int	Start time in sec	Example: 1581104400	N
end_time	Int	End time in sec	Example: 1581104700	N

Example:

```
args = {'market': "vetusdt"}
args = json.loads(json.dumps(args))
```

```
        data = {
            "identifier": {"handler": "OrderHistoryHandler"},
            "command": "message",
            "data": {
                "action": "index",
                "uuid": str(uuid.uuid1()),
                "args": args
            }
        }
```

Response Example:


```
{
  "identifier": {
    "handler": "OrderHistoryHandler"
  },
  "message": {
    "action": "index",
    "code": 0,
    "data": {
      "resources": [
        {
          "avg_price": "0.0",
          "created_at": "2020-03-09T18:31:40Z",
          "created_on": 1583778700,
          "executed_volume": "0.0",
          "id": 43,
          "market": "vetusdt",
          "market_display_name": "VET/USDT",
          "market_name": "VET/USDT",
          "ord_type": "limit",
          "price": "0.75",
          "remaining_volume": "1000.0",
          "side": "bid",
          "state": "cancel",
          "trades_count": 0,
          "volume": "1000.0"
        },
        {
          "avg_price": "0.0",
```

```
          "created_at": "2020-03-09T18:31:37Z",
          "created_on": 1583778697,
          "executed_volume": "0.0",
          "id": 42,
          "market": "vetusdt",
          "market_display_name": "VET/USDT",
          "market_name": "VET/USDT",
          "ord_type": "limit",
          "price": "0.75",
          "remaining_volume": "1000.0",
          "side": "bid",
          "state": "cancel",
          "trades_count": 0,
          "volume": "1000.0"
        }
      ],
      "resources_count": 43
    },
    "uuid": "00f07308-623b-11ea-b613-7aa0b0f73a47"
  }
}
```


5. Support

Please email questions or comments regarding this specification to [OceanEx Support.]