

Trading API

v1.4.0

Overview

This documentation provides both HTTP and WebSocket APIs for interacting with the exchange. Both allow read access to public market data and private read access to your account. Private write access to your account is available via the private API.

The public HTTP endpoint is accessed via GET requests while the private endpoint is accessed via HMAC-SHA512 signed POST requests using API keys. Both types of HTTP endpoints return results in JSON format.

The WebSocket API allows pushing notifications about the public order books and your private account. Similarly to the HTTP API, it requires HMAC-SHA512 signed requests using API keys for requests related to your private account.

The base URLs for requests are:

Code	URL
[base]	https://api.unistex.com:8443
[base2]	https://cmc-gate.unistex/marketdata/cmc/v1/

Required fields are marked with *****.

Private API

All REST requests must contain two HTTP header fields:

- "Key": public key
- "Sign": hash of payload

API keys are UUID string (UTF-8), for example: “ca3a03e1-fc5c-4954-99dc-876db3997d8f”. API keys are obtained through Trader’s Room: Profile > API Keys Management. The payload is a JSON body. For hashing the HMAC-SHA512 algorithm is used.

The POST and DELETE requests must include the `ts` field in body. The `ts` value contains a string representation of the current time in UTC date time, for example: “2019-12-20T08:20:51”.

API Rate Limits

- `/frontoffice/*` — 1 request per second
- `/marketdata/*` — 2 requests per second

Public REST Methods

Supported Instruments

GET `[base]/frontoffice/api/info`

Returns information on all currency pairs supported by the exchange. In case of success, the following JSON-structure will be returned in response body:

Field	Type	Description
<code>baseAsset</code>	string	Base asset
<code>quoteAsset</code>	string	Quote asset
<code>hidden</code>	number	Instrument availability: 0 — available, 1 — hidden (trading is not recommended, because this instrument may be preparing to disconnect)
<code>status</code>	string	Instrument status: <code>open</code> , <code>paused</code> (cannot be traded), <code>halted</code> (cannot be traded, orders are cancelled)
<code>makerFee</code>	number	Maker fee, paid if the order adds liquidity
<code>makerFeeLimit</code>	number	Minimum maker fee value
<code>takerFee</code>	number	Taker fee, paid if the order removes liquidity

takerFeeLimit	number	Minimum taker fee value
priceScale	number	Price step
amountScale	number	Volume step
createdAt	string	Date time of instrument creation: YYYY-MM-DDThh:mm:ss
updatedAt	string	Date time of instrument last update: YYYY-MM-DDThh:mm:ss

Response example

```
{
  "serverTime": 636880696809972288,
  "pairs": {
    "btc_usdt": {
      "baseAsset": "btc",
      "quoteAsset": "usdt",
      "status": "Open",
      "hidden": 0,
      "makerFee": 0,
      "makerFeeLimit": 0,
      "takerFee": 0.001,
      "takerFeeLimit": 0,
      "priceScale": 6,
      "amountScale": 6,
      "createdAt": "2019-11-14T16:18:49.253354",
      "updatedAt": "2019-11-14T16:18:49.253354"
    },
    "gnt_usdt": {
      "baseAsset": "gnt",
      "quoteAsset": "usdt",
      "status": "Open",
      "hidden": 0,
      "makerFee": 0,
      "makerFeeLimit": 0,
      "takerFee": 0.001,
      "takerFeeLimit": 0,
      "priceScale": 5,
      "amountScale": 5,
      "createdAt": "2019-11-14T16:18:49.253354",
      "updatedAt": "2019-11-14T16:18:49.253354"
    }
  }
}
```

Order Book Snapshot

GET [base]/marketdata/instruments/{instrument}/depth

Returns order book for a specified currency pair. The following request parameters are available:

Field	Type	Description
instrument*	string	Instrument identifier: {baseAsset}_{quoteAsset}

Request example

```
[base]/marketdata/instruments/eth_btc/depth
```

In case of success, the following JSON-structure will be returned in response body:

Field	Type	Description
instrument	string	Instrument identifier, same as in the request
bids	array	Each element represents one particular order and contains number fields amount and price
asks	array	Each element represents one particular order and contains number fields amount and price
version	number	Current snapshot version
askTotalAmount	number	Sum of all ask orders
bidTotalAmount	number	Sum of all bid orders
snapshot	boolean	This is a self-contained snapshot (always <code>true</code> for this request)

Response example

```
{
  "name": "btc_usd",
  "timestamp": "1599662893",
  "bids": [
    ["10244.265471", "1.09680273"],
    ["10244.255472", "0.47521749"],
    ["10243.695528", "0.72676099"]
  ],
  "asks": [
    ["10246.794577", "0.28242939"],
    ["10246.814579", "10.35540379"],
    ["10247.604658", "0.72676099"]
  ]
}
```

Instrument Candles

New in version v1.2.0.

GET [base]/marketdata/instruments/{instrument}/history

Returns candlesticks graph for a specified currency pair. The following request parameters are available:

Field	Type	Description
instrument*	string	Instrument identifier: {baseAsset}_{quoteAsset}
startDate*	string	Date time of period start: YYYY-MM-DDThh:mm:ss
endDate*	string	Date time of period end: YYYY-MM-DDThh:mm:ss
type*	string	Timeframe, the following values are available: <ul style="list-style-type: none">● 1m — 1 Minute● 5m — 5 Minutes● 15m — 15 Minutes● 30m — 30 Minutes● 1h — 1 Hour● 12h — 12 Hours● 1d — 1 Day● 1w — 1 Week● 1M — 1 Month

count	number	Number of candles to return, defaults to 1000 (maximum value)
-------	--------	---

Request example

[base]/marketdata/instruments/btc_usdt/history?startDate=2019-03-13T09:00:00&endDate=2019-03-13T11:00:00&type=1h&count=2

In case of success, the following JSON-structure data will be returned in response body:

Field	Type	Description
instrument	string	Same as in the request
start	string	Same as in the request
end	string	Same as in the request
low	number	Lowest price
high	number	Highest price
volume	number	Total volume
quoteVolume	number	Total quote asset volume
open	number	Open price
close	number	Close price

Response example

```
{
  "success": true,
  "instrument": "btc_usdt",
  "data": [
    {
      "instrument": "btc_usdt",
      "start": "2019-03-13T09:00:00Z",
      "end": "2019-03-13T10:00:00Z",
      "low": 3842.855,
      "high": 3855.445,
      "volume": 4,
      "quoteVolume": 0,
    }
  ]
}
```

```
    "open": 3855.105,
    "close": 3842.855
  },
  {
    "instrument": "btc_usdt",
    "start": "2019-03-13T10:00:00Z",
    "end": "2019-03-13T11:00:00Z",
    "low": 3834.355,
    "high": 3848.855,
    "volume": 26,
    "quoteVolume": 0,
    "open": 3842.865,
    "close": 3835.655
  }
],
"startDateTime": "2019-03-13T09:00:00Z",
"endDateTime": "2019-03-13T11:00:00Z"
}
```

Asset Info

New in version v1.3.0.

GET [base2]/asset

Returns assets information.

In case of success, JSON-object will be returned in response body with the following fields:

Field	Type	Description
name	string	Asset name
can_withdraw	string	If true, the asset can be withdrawn
can_deposit	string	If true, the asset can be deposited
min_withdraw	string	Minimum withdrawal/deposit amount
max_withdraw	string	Maximum withdrawal/deposit amount

Response Example

```
{
  "BTC": {
    "name": "btc",
    "can_withdraw": true,
    "can_deposit": true,
    "min_withdraw": "0.00000001",
    "max_withdraw": "100000000"
  },
  "USDT": {
    "name": "usdt",
    "can_withdraw": true,
    "can_deposit": true,
    "min_withdraw": "0.00000001",
    "max_withdraw": "100000000"
  },
  "USD": {
    "name": "usd",
    "can_withdraw": true,
    "can_deposit": true,
    "min_withdraw": "0.00000001",
    "max_withdraw": "100000000"
  }
}
```

Summary

New in version v1.3.0.

GET [base2]/summary

Returns summary information.

In case of success, JSON-object will be returned in response body with the following fields:

Field	Type	Description
id	string	Market identifier
last	string	Price of the last trade in the last 24 hours. If the deal was more than 24 hours ago: 0
lowestAsk	string	Current lowest ask price
highestBid	string	Current highest bid price

percentChange	string	Price change in the last 24 hours
baseVolume	string	All traded volume in the last 24 hours in base currency
quoteVolume	string	All traded volume in the last 24 hours in quote currency
isFrozen	string	Market status: 0 — working, 1 — temporarily suspended
high24hr	string	Maximum trade price in the last 24 hours
low24hr	string	Minimum trade price in the last 24 hours

Response example

```
{
  "BTC_USDT": {
    "id": "btc_usdt",
    "last": "10978.93578",
    "lowestAsk": "10979.0",
    "highestBid": "10978.71",
    "percentChange": "0.0813730364297798727996051454",
    "baseVolume": "6.47119743",
    "quoteVolume": "70829.9781692126756",
    "isFrozen": "0",
    "high24hr": "10985.0049",
    "low24hr": "10857.95376"
  },
  "BTC_USD": {
    "id": "btc_usd",
    "last": "0",
    "lowestAsk": "0",
    "highestBid": "0",
    "percentChange": "0",
    "baseVolume": "0",
    "quoteVolume": "0",
    "isFrozen": "0",
    "high24hr": "0",
    "low24hr": "0"
  }
}
```

Ticker Info

New in version v1.3.0.

GET [base2]/ticker

Returns ticker information.

In case of success, JSON-object will be returned in response body with the following fields:

Field	Type	Description
base_name	string	Base asset name
quote_name	string	Quote asset name
last_price	string	Price of the last trade in the last 24 hours. If the deal was more than 24 hours ago: 0
base_volume	string	All traded volume in the last 24 hours in base currency
quote_volume	string	All traded volume in the last 24 hours in quote currency
isFrozen	string	Market status: 0 — working, 1 — temporarily suspended

Response example

```
{
  "dash_btc": {
    "base_name": "dash",
    "quote_name": "btc",
    "last_price": "0",
    "base_volume": "0",
    "quote_volume": "0",
    "isFrozen": "1"
  },
  "eth_usdt": {
    "base_name": "eth",
    "quote_name": "usdt",
    "last_price": "423.9936",
    "base_volume": "2942.97774",
    "quote_volume": "1273092.080666887",
    "isFrozen": "0"
  }
}
```

Trades Info

New in version v1.3.0.

GET [base2]/trades/{instrument}

Returns all trades for a specified instrument in the last 24 hours. The following request parameters are available:

Field	Type	Description
instrument*	string	Instrument identifier: {baseAsset}_{quoteAsset}

Request Example

```
[base2]/trades/btc_usd
```

In case of success, JSON-object will be returned in response body with the following fields:

Field	Type	Description
tradeID	string	Trade identifier
price	string	Trade price
base_volume	string	Amount in base currency
quote_volume	string	Amount in quote currency
trade_timestamp	string	Datetime of trade in UNIX timestamp format
type	string	Trade side: "buy" or "sell"

Response example

```
[  
  {  
    "tradeID": "1247307",  
    "price": "10093.92246491",  
    "base_volume": "0.0259",  
    "quote_volume": "261.432591841169",  
    "trade_timestamp": "1599577070",
```

```
    "type": "buy"
  },
  {
    "tradeID": "1247309",
    "price": "10091.69185435",
    "base_volume": "0.0754",
    "quote_volume": "760.913565817990",
    "trade_timestamp": "1599577128",
    "type": "sell"
  }
]
```

Private REST Methods

Place an Order

POST [base]/frontoffice/api/order

Places a new order. Request body must contain ts field and JSON-object order with the following fields:

Field	Type	Description
instrument*	string	Instrument identifier: {baseAsset}_{quoteAsset}
type*	string	Order side: "buy" or "sell"
amount*	number	Order amount, must be greater than 0
price	number	Order price. Required for limit orders, optional for market orders. If isLimit is true, must be greater than 0, in other cases can be equal to 0
isLimit	boolean	If true, the order is limit. If false, the order is market. Ref. to Flags Combinations for details
isStop	boolean	<i>Currently not supported</i> If true, the order is stop market or stop limit (depending on isLimit value). Ref. to Flags Combinations for details
isFok	boolean	If true, the order must be executed immediately or be cancelled if not filled. Ref. to Flags Combinations for details

activationPrice	number	<i>Currently not supported</i> Required and applicable only if <code>isStop</code> is <code>true</code>
clientOrderId	number	Client provided order identifier: any UUID, except for 00000000-0000-0000-0000-000000000000

Flags Combinations

Order	Description	Flags
Limit	Limit order	<code>isLimit</code> is <code>true</code> , <code>isStop</code> is <code>false</code> , <code>isFok</code> is <code>false</code>
Market (Market IOK)	Market order (Immediate-or-Cancel market order) — order must be executed immediately and cancel any unfilled portion	<code>isLimit</code> is <code>false</code> , <code>isStop</code> is <code>false</code> , <code>isFok</code> is <code>false</code>
Market FOK	Fill-or-Kill market order — order must be executed immediately or cancelled if not filled	<code>isLimit</code> is <code>false</code> , <code>isStop</code> is <code>false</code> , <code>isFok</code> is <code>true</code>

Request example

```
{
  "ts": "2019-12-12T01:01:01",
  "order": {
    "instrument": "btc_usdt",
    "type": "sell",
    "amount": 1,
    "price": 1,
    "isLimit": true,
    "isStop": false,
    "isFok": false,
    "activationPrice": 0,
    "clientOrderId": "6fdf688e-00b0-4c68-82dd-3aee5c727ed1"
  }
}
```

In case of success, JSON-structure object will be returned in response body:

Field	Type	Description
orderId	number	System order identifier
total	number	Total amount of quoted currency sold (quote amount)
orderType	number	Order side: 0 — sell, 1 — buy
commission	number	Commission in quoted currency
createdAt	string	Date time of order creation
unitsFilled	string	Base amount
isPending	boolean	If <code>true</code> , the order is active (not rejected, cancelled, or completed)
status	string	Order status: Working, Rejected, Cancelled, Completed
type	string	Same as in the request
amount	number	Same as in the request
remaining	number	Remaining amount
price	number	Same as in the request
stopPrice	number	Order stop price
isLimit	boolean	Same as in the request
isStop	boolean	Same as in the request
isFok	boolean	Same as in the request
instrument	string	Same as in the request
side	string	Order side: 0 — buy, 1 — sell

Response example

```
{
  "order": {
    "orderId": "-72057594037927933",
    "total": 0.0,
    "orderType": 0,
    "commission": 0.0,
    "createdAt": "2020-08-18T12:18:12.2296801Z",
    "unitsFilled": 0.0,
    "isPending": true,
    "status": "working",
    "type": "buy",
    "amount": 1,
    "remaining": 1,
    "price": 1,
    "stopPrice": 0.0,
    "isLimit": true,
    "instrument": "eth_usd",
    "side": 0
  }
}
```

JavaScript

```
const crypto = require('crypto');
const https = require('https');
const Key = "7fa6ceec-d8fc..."; // Change to your PublicKey
const Secret = "7ae49b5b-99db..."; // Change to your Secret
const Host = "api.unistex.com"; //
const Port = 8443;
const Method = '/trading/frontoffice/api/order'
const Payload = new Date().toISOString();
const body = {
  ts: Payload,
  order: {
    instrument: 'eth_usd',
    type: 'buy',
    amount: 1,
    price: 1,
    isLimit: true
  }
};
const data = JSON.stringify(body);
const Sign = crypto
  .createHmac('sha512', Secret)
  .update(data)
  .digest('hex')
  .toUpperCase();
```

```

const options = {
  hostname: Host,
  path: Method,
  port: Port,
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    'Content-Length': data.length,
    Key,
    'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:47.0) Gecko/20100101
Firefox/47.0',
    Sign
  }
};
const req = https.request(options, res => {
  console.log(`statusCode: ${res.statusCode}`);
  res.on('data', d => {
    process.stdout.write(d);
  });
});
req.on('error', e => {
  console.error(e);
});
req.write(data);
req.end();

```

Cancel an Order

DELETE [base]/frontoffice/api/orders?orderId={id}

Cancels the order. The following request parameters are available:

Field	Type	Description
id*	string	Order identifier: either <code>orderId</code> — same as in the response message generated with Place an Order request, or <code>clientOrderId</code> — same as in the Place an Order request

Changed in version v1.1.0: Possibility to cancel an order by clientOrderId added.

Request example — Cancel by `orderId`

```
[base]/frontoffice/api/orders?orderId=-72057594037927933&ts=2019-12-12T01:01:01
```

Request example — Cancel by `clientOrderId`

```
[base]/frontoffice/api/orders?orderId=6fdf688e-00b0-4c68-82dd-3aee5c727ed1&ts=2019-12-12T01:01:01
```

In case of success, JSON-structure object will be returned in the response body (for details ref. to Place an Order response description).

Response example

```
{
  "order": {
    "orderId": "-72057594037927933",
    "total": 0.0,
    "orderType": 0,
    "commission": 0.0,
    "createdAt": "2020-08-18T12:28:26.6415513Z",
    "unitsFilled": 0.0,
    "isPending": false,
    "status": "cancelled",
    "type": "buy",
    "amount": 0.1,
    "remaining": 0.1,
    "price": 0.001,
    "stopPrice": 0.0,
    "isLimit": true,
    "instrument": "eth_usd",
    "side": 0
  }
}
```

JavaScript — Cancel by `orderId` example

```
const crypto = require('crypto');
const https = require('https');
const Key = "7fa6ceec-d8fc..."; // Change to your PublicKey
const Secret = "7ae49b5b-99db..."; // Change to your Secret
5
const Port = 8443;
const Payload = new Date().toISOString();
const OrderId = '-72057594037927933'; // Change to your OrderId
```

```

const Method = `/trading/frontoffice/api/orders?OrderId=${OrderId}&ts=${Payload}`
async function deleteOrder() {
  console.log(`?OrderId=${OrderId}&ts=${Payload}`);
  const Sign = crypto
    .createHmac('sha512', Secret)
    .update(`?OrderId=${OrderId}&ts=${Payload}`)
    .digest('hex')
    .toUpperCase();
  const options = {
    hostname: Host,
    path: Method,
    port: Port,
    method: 'DELETE',
    headers: {
      'Content-Type': 'application/json',
      'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:47.0)
Gecko/20100101 Firefox/47.0',
      Key,
      Sign
    }
  };
  const req = https.request(options, res => {
    console.log(`statusCode: ${res.statusCode}`);
    console.log(`statusMessage: ${res.statusMessage}`);
    res.on('data', d => {
      process.stdout.write(d);
    });
  });
  req.on('error', e => {
    console.error(e);
  });
  req.end();
}
deleteOrder().catch(err => {
  console.log(err);
});

```

JavaScript — Cancel by `clientId` example

```

const crypto = require('crypto');
const https = require('https');
const Key = "7fa6ceec-d8fc..."; // Change to your PublicKey
const Secret = "7ae49b5b-99db..."; // Change to your Secret
const Host = "api.unistex.com"; //
const Port = 8443;
const Payload = new Date().toISOString();
const ClientOrderId = '6fdf688e-00b0-4c68-82dd-3aee5c727ed1'; // Change to your
ClientOrderId

```

```

const Method =
`/trading/frontoffice/api/orders?OrderId=${ClientOrderId}&ts=${Payload}`
async function deleteOrder() {
  console.log(`?OrderId=${ClientOrderId}&ts=${Payload}`);
  const Sign = crypto
    .createHmac('sha512', Secret)
    .update(`?OrderId=${ClientOrderId}&ts=${Payload}`)
    .digest('hex')
    .toUpperCase();
  const options = {
    hostname: Host,
    path: Method,
    port: Port,
    method: 'DELETE',
    headers: {
      'Content-Type': 'application/json',
      'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:47.0)
Gecko/20100101 Firefox/47.0',
      Key,
      Sign
    }
  };
  const req = https.request(options, res => {
    console.log(`statusCode: ${res.statusCode}`);
    console.log(`statusMessage: ${res.statusMessage}`);
    res.on('data', d => {
      process.stdout.write(d);
    });
  });
  req.on('error', e => {
    console.error(e);
  });
  req.end();
}
deleteOrder().catch(err => {
  console.log(err);
});

```

Orders History

GET [base]/frontoffice/api/order_history

Returns the orders history. The following request parameters are available:

Field	Type	Description
market	string	Market filter: currency pair (for example: btc_usdt)

side	string	Order side: either "buy" or "sell"
status	string	Order status: Working, Rejected, Cancelled, Completed
startDate	string	Date time of period start: YYYY-MM-DDThh:mm:ss.s
endDate	string	Date time of period end: YYYY-MM-DDThh:mm:ss.s

Request example

```
[base]/frontoffice/api/order_history?market=btc_usdt&ts=2019-12-12T01:01:01
```

In case of success, JSON-object data will be returned in the response body (for details ref. to Place an Order response description).

Response example

```
{
  "filters": {
    "market": "btc_usd"
  },
  "paging": {
    "page": 1,
    "per_page": 15,
    "total": 0
  },
  "data": [{
    "orderId": "-72057593704231199",
    "total": 0.0,
    "orderType": 0,
    "commission": 0.0,
    "createdAt": "2020-08-18T12:53:19.755247Z",
    "unitsFilled": 0.0,
    "isPending": false,
    "status": "cancelled",
    "type": "buy",
    "amount": 0.57945583,
    "remaining": 0.57945583,
    "price": 12241.61,
    "stopPrice": 0.0,
    "isLimit": true,
    "instrument": "btc_usd",
    "side": 0
  },
  {
```

```

    "orderId": "-72057593704231200",
    "total": 0.0,
    "orderType": 0,
    "commission": 0.0,
    "createdAt": "2020-08-18T12:53:19.755163Z",
    "unitsFilled": 0.0,
    "isPending": false,
    "status": "cancelled",
    "type": "sell",
    "amount": 0.4081,
    "remaining": 0.4081,
    "price": 12253.93,
    "stopPrice": 0.0,
    "isLimit": true,
    "instrument": "btc_usd",
    "side": 1
  }
]
}

```

JavaScript

```

const crypto = require('crypto');
const https = require('https');
const Key = "7fa6ceec-d8fc..."; // Change to your PublicKey
const Secret = "7ae49b5b-99db..."; // Change to your Secret
const Host = "api.unistex.com";
const Port = 8443;
const Payload = new Date().toISOString();
const Method = `/trading/frontoffice/api/order_history?ts=${Payload}&market=btc_usd`;
async function GetOrders() {
  console.log(`ts=${Payload}`);
  const Sign = crypto
    .createHmac('sha512', Secret)
    .update(`?ts=${Payload}&market=btc_usd`)
    .digest('hex')
    .toUpperCase();
  const options = {
    hostname: Host,
    port: Port,
    path: Method,
    method: 'GET',
    headers: {
      'Content-Type': 'application/json',
      'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:47.0)
Gecko/20100101 Firefox/47.0',
      Key,
      Sign
    }
  }
}

```

```

};
const req = https.request(options, res => {
  console.log(`statusCode: ${res.statusCode}`);
  console.log(`statusMessage: ${res.statusMessage}`);
  res.on('data', d => {
    process.stdout.write(d);
  });
});
req.on('error', e => {
  console.error(e);
});
req.end();
}
GetOrders().catch(err => {
  console.log(err);
});

```

Trades History

GET [base]/frontoffice/api/trade_history

Returns trades history. The following request parameters are available:

Field	Type	Description
orderId	number	Order identifier, by which it is possible to get information about all trades, related to it <i>New in version v1.4.0.</i>
market	string	Market filter: currency pair (for example: btc_usdt)
side	string	Order side: either "buy" or "sell"
startDate	string	Date time of period start: YYYY-MM-DDThh:mm:ss.s
endDate	string	Date time of period end: YYYY-MM-DDThh:mm:ss.s

Request example

[base]/frontoffice/api/trade_history?market=btc_usdt

In case of success, JSON-object data will be returned in response body with the following fields:

Field	Type	Description
tradeSeq	number	Sequence number of the trade
tradeTime	string	Date time of the trade
amount	number	Trade amount
executionPrice	number	Trade execution price
instrument	string	Trade instrument
side	number	Trade side: 0 — buy, 1 — sell
commission	number	Trade commission
orderId	number	Order identifier of the trade

Response example

```
{
  "filters": {
    "market": "btc_usdt"
  },
  "paging": {
    "page": 1,
    "per_page": 15,
    "total": 0
  },
  "data": [
    {
      "tradeSeq": 0,
      "tradeTime": "2019-12-20T06:17:03.093597",
      "amount": 0.00000001,
      "executionPrice": 0.00000001,
      "instrument": "btc_usdt",
      "side": 0,
      "commission": 0.00000000,
      "orderId": -72057593704402280
    },
    {
      "tradeSeq": 3927,
      "tradeTime": "2019-12-20T06:17:03.093597",
      "amount": 0.00000001,
      "executionPrice": 0.00000001,
```

```

    "instrument": "btc_usdt",
    "side": 1,
    "commission": 0.00000000,
    "orderId": -72057593704402281
  }
]
}

```

JavaScript

```

const crypto = require('crypto');
const https = require('https');
const Key = "7fa6ceec-d8fc..."; // Change to your PublicKey
const Secret = "7ae49b5b-99db..."; // Change to your Secret
const Host = "api.unistex.com";
const Port = 8443;
const Payload = new Date().toISOString();
const Method =
`/trading/frontoffice/api/trade_history?ts=${Payload}&market=btc_usd&startDate=2020-08-12T01:01:01`;
async function GetOrders() {
  console.log(`ts=${Payload}`);
  const Sign = crypto
    .createHmac('sha512', Secret)
    .update(`?ts=${Payload}&market=btc_usd&startDate=2020-08-12T01:01:01`)
    .digest('hex')
    .toUpperCase();
  const options = {
    hostname: Host,
    port: Port,
    path: Method,
    method: 'GET',
    headers: {
      'Content-Type': 'application/json',
      'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:47.0) Gecko/20100101 Firefox/47.0',
      Key,
      Sign
    }
  };
};
const req = https.request(options, res => {
  console.log(`statusCode: ${res.statusCode}`);
  console.log(`statusMessage: ${res.statusMessage}`);
  res.on('data', d => {
    process.stdout.write(d);
  });
});
req.on('error', e => {
  console.error(e);
}

```

```
});  
req.end();  
}  
GetOrders().catch(err => {  
  console.log(err);  
});
```

HTTP Codes and Errors

200 OK

400 Bad Request: incorrect parameters

```
{  
  "errors": [  
    {  
      "message": "Unknown instrument 'gold_silver'",  
      "value": "gold_silver",  
      "key": "instrument"  
    }  
  ]  
}
```

401 Unauthorized: incorrect keys or ts value differs from the current time by more than 5 seconds

404 Not Found

```
{  
  "errors": [  
    {  
      "message": "Not Found",  
      "key": "instrument"  
    }  
  ]  
}
```

429 Too Many Requests: API Rate Limits violated

500 Internal Server Error

```
{
  "errors": [
    {
      "message": "Stop orders are not supported."
    }
  ]
}
```

503 Service Unavailable

Either JSON:

```
{
  "errors": [
    {
      "message": "System is currently overloaded."
    }
  ]
}
```

or string:

```
"This service is currently undergoing maintenance."
```

Public WebSocket Methods

Real-Time Order Book Data

Open connection with public hub: `[base]/marketdata/info`

Order book channel name is "Book".

Attention

Upon subscription, a complete snapshot of a book will be sent (`snapshot: true`), after that subsequent responses will contain only updates, not the whole book (`snapshot: false`).

For response details ref. to Order Book Snapshot response description.

Response example

```
{
  "instrument": "btc_usd",
  "bids": [
    {
      "amount": 0,
      "price": 11883.731508
    },
    {
      "amount": 0,
      "price": 11884.241457
    },
    {
      "amount": 0,
      "price": 11884.251456
    },
    {
      "amount": 6.57132172,
      "price": 11884.421439
    },
    {
      "amount": 4.18222543,
      "price": 11884.431438
    },
    {
      "amount": 0,
      "price": 11885.181363
    },
    {
      "amount": 1.30694544,
      "price": 11887.141167
    }
  ],
  "asks": [
    {
      "amount": 12.40938478,
      "price": 11890.1889
    },
    {
      "amount": 28.87423081,
      "price": 11890.208902
    },
    {
      "amount": 1.30694544,
      "price": 11890.58894
    },
    {
      "amount": 0,
      "price": 11890.728954
    }
  ]
}
```

```
    }  
  ],  
  "version": 443366001,  
  "askTotalAmount": 170.26792556,  
  "bidTotalAmount": 164.11677626,  
  "snapshot": false  
}
```

JavaScript

```
const { HubConnectionBuilder } = require("@microsoft/signalr");  
  
const url = "https://api.unistex.com:8443/trading/marketdata/info";  
const instrument = "btc_usd";  
const channel = 'Book';  
  
let connection = new HubConnectionBuilder()  
  .withUrl(url)  
  .build();  
  
connection.start().then(function () {  
  connection.stream(channel, instrument)  
    .subscribe({  
      next: (item) => {  
        console.log(JSON.stringify(item));  
      },  
      error: (err) => {  
        console.log(err);  
      },  
    });  
});
```

Private WebSocket Methods

Orders Data

Open connection with private hub: [base]/ws/account

Open orders channel name is "OpenOrders".

Upon subscription, the following JSON-object will be sent (for details ref. to Place an Order response description).

Response example

```
[
  {
    "orderId": "-72057593698565937",
    "total": 0,
    "orderType": 0,
    "commission": 0,
    "createdAt": "2020-09-01T09:01:15.3651338Z",
    "unitsFilled": 0,
    "isPending": true,
    "status": 0,
    "type": "sell",
    "amount": 1,
    "remaining": 1,
    "price": 1,
    "stopPrice": 0,
    "isLimit": true,
    "instrument": "btc_usdt",
    "side": 1
  }
]
```

JavaScript

```
const { HubConnectionBuilder } = require("@microsoft/signalr");
const crypto = require("crypto");
const Key = "7fa6ceec-d8fc..."; // Change to your PublicKey
const Secret = "7ae49b5b-99db..."; // Change to your Secret
const Payload = new Date().toISOString();
const Sign = crypto
  .createHmac('sha512', Secret)
  .update(Payload)
  .digest('hex')
  .toUpperCase();
const headers = {
  Key,
  Payload,
  Sign
};

const url = "https://api.unistex.com:8443/trading/ws/account";
const channel = "OpenOrders";
let connection = new HubConnectionBuilder()
  .withUrl(url, {headers: headers})
  .build();

connection.start().then(function () {
  connection.stream(channel)
```

```
.subscribe({
  next: (item) => {
    console.log(JSON.stringify(item));
  },
  error: (err) => {
    console.log(err);
  },
});
});
```

Real-Time Balances

Open connection with private hub: [base]/ws/account

Balance updates channel name is "FullBalance".

Upon subscription, a JSON-structure with the following fields will be sent:

Field	Type	Description
asset	string	Asset name
amount	number	Asset amount
balance	number	Full number
locked	number	Locked amount

Response example

```
[
  {
    "asset": "ada",
    "amount": 1000000000,
    "balance": 1000000000,
    "locked": 0
  },
  {
    "asset": "uni",
    "amount": 10000000,
    "balance": 10000000,
    "locked": 0
  },
  {
    "asset": "bch",
```

```
    "amount": 9999999.999999999,  
    "balance": 9999999.999999999,  
    "locked": 0  
  }  
]
```

JavaScript

```
const { HubConnectionBuilder } = require("@microsoft/signalr");  
const crypto = require("crypto");  
const Key = "7fa6ceec-d8fc..."; //Change to your PublicKey  
const Secret = "7ae49b5b-99db..."; //Change to your Secret  
const Payload = new Date().toISOString();  
const Sign = crypto  
  .createHmac('sha512', Secret)  
  .update(Payload)  
  .digest('hex')  
  .toUpperCase();  
const headers = {  
  Key,  
  Payload,  
  Sign  
};  
  
const url = "https://api.bitbull.trade:8443/trading/ws/account";  
const channel = "FullBalance";  
let connection = new HubConnectionBuilder()  
  .withUrl(url, {headers: headers})  
  .build();  
  
connection.start().then(function () {  
  connection.stream(channel, {})  
    .subscribe({  
      next: (item) => {  
        console.log(JSON.stringify(item));  
      },  
      error: (err) => {  
        console.log(err);  
      },  
    });  
});
```