Elektro backend integration guide

Table of contents

Introduction

We have REST web service to interact with backend and WebSockets to:

- Receive notifications about order's statuses
- Subscribe to orderbooks and receive orderbook updates
- Subscribe to trading reports and receive trading reports once batch is performed

REST description

Response format

REST response format:

```
{
  "result":"OK",
  "details":"",
  "payload":{…}
}
```

- If the result is OK, the payload field contains the appropriate payload, and the details field is empty.
- If the result is not OK, the payload is empty, details may contain additional information.

Result codes

Result code	**Description**
OK	Request was successfully processed
INVALID_NONCE	The nonce is invalid
INVALID_SIGNATURE	The signature is invalid
UNKNOWN_ETH_ADDRESS	Eth address is not registered. Deposit function was not invoked or event has not been caught by the backend yet
INVALID_REQUEST_DATA	Something is wrong with the request. May be request fields were not filled properly
INVALID_ORDER_CLASS	Something is wrong with the request. Maybe request fields were not filled correctly, or the request body doesn't fit the endpoint URL.
TRADING_IS_NOT_STARTED_YET	Need to wait till the system starts or becomes trading hour.
INVALID_ORDER_SIDE	Side field in the order was not filled properly
INVALID_ORDER_TYPE	Order type field in the order was not filled properly
INVALID_ORDER_TIF	Time in force field in the order was not filled properly
INVALID_ACCOUNT	Something is wrong with the account or eth address registered in the backend
INVALID_CONTRACT_ID	Contract id is invalid, contract does not exist or is not tradeable anymore
INVALID_ORDER_PRICE_PRECISION	The price field is filled with incorrect precision. Digits after the point should not exceed backend precision for the currency
INVALID_ORDER_PRICE	Price field was not set or negative for single order
INVALID_ORDER_QTY_PRECISION	The quantity field is filled with incorrect precision. Digits after the point should not exceed backend precision for the currency
INVALID_ORDER_QTY	Quantity field was not set or negative for single order

INVALID_ORDER_ID	Order id is invalid or does not exist
INVALID_CN_VALIDATION_FAILED	Something went wrong on deeper validation, check details field for more information
INVALID_ORDER_LEGS	Order legs were not found in conditional order or duplicate in contract ids found.

HTTP Codes

Code	Name	Description
200	OK	Response body contains REST response. Request successfully processed
412	PRECONDITION_FAILED	Response body contains REST response. Request processed, but error occurred
401	UNAUTHORIZED	Response body is empty. Authentication required. The session has expired or has not been authenticated
500	INTERNAL_SERVER_ERROR	Unexpected error. Response body is empty

Web socket description

Response format

Web socket response format:

```
{
    "error": true,
    "cause": "INVALID_AUTH_TOKEN",
    "details": "",
    "responseType": "VALIDATE_AUTH_TOKEN_RESPONSE",
    " payload" : {…}
}
```

- If the error flag is false, response type contains type of the payload and payload field contains appropriate payload, details and cause are empty.
- If the error flag is true, the payload is empty, details may contain additional information, cause contains error code.

Cause codes

Value	Description
GENERAL	Something went wrong on the backend
INVALID_REQUEST	Request format is invalid, backend was not able to parse it
INVALID_REQUEST_ACTION	Request action is invalid
INVALID_CONTRACT_ID	Contract is not found or not tradeable anymore
ALREADY_SUBSCRIBED	Client already subscribed to requested channel
NOT_SUBSCRIBED	Client was already unsubscribed from requested channel or was not subscribed there at all
SERVICE_NOT_AVAILABLE_YET	Need to wait till the system starts or becomes trading hour.
INVALID_AUTH_TOKEN	Authentication token is invalid
RECONNECT_REQUIRED	Need to disconnect web socket and connect again.

Response types

Value	Description
VALIDATE_AUTH_TOKEN_RESPONSE	See Web socket authentication chapter
ORDERBOOK	See subscribe orderbook chapter
ORDERBOOK_DELTA	See subscribe orderbook chapter
SUBSCRIBE_ORDER_BOOK_RESPONSE	See subscribe orderbook chapter
UNSUBSCRIBE_ORDER_BOOK_RESPONSE	See unsubscribe orderbook chapter
SUBSCRIBE_TRADE_REPORT_RESPONSE	See Subscribe trade reports chapter
UNSUBSCRIBE_TRADE_REPORT_RESPONSE	See Unsubscribe trade reports chapter
TRADE_REPORT	See Subscribe trade reports chapter
EXEC_REPORT	Contains execution report structure, see Trade history chapter
HEARTBEAT_RESPONSE	See Heartbeat request chapter
UNKNOWN	Check cause code, there is an error

Client registration

All that the client should do to register in Elektro system is deposit some amount to the smart contract.

Elektro backend will catch deposit event through the blockchain logs and register client automatically.

Authentication

Most actions require authentication. The only things work without authentication are:

- Subscribe to orderbooks and receive orderbook updates
- Subscribe to trading reports and receive trading reports once batch is performed
- Retrieve contracts list

REST authentication

- Client sends request to the server
- Server sends nonce
- Clients signs nonce
- Client sends signed nonce to the server
- Server validates signature and nonce and authenticates client's session

Authentication is based on the session. Client's session is based on HTTP cookie.

Endpoint	`POST /api/v1/auth/requestAuth`
Request	`{ "ethAddress":"0x7fda7543e01caafd1af39585a156eadb3375d234" }`
Response payload	`{ "nonce":"auth-request-for-0xbd44572e53343a0f003b719cf438c6338bd29d9c-415731094491489277 2" }`

Endpoint	POST /api/v1/auth/validateAuth

Request	{
	"nonce":"auth-request-for-0xbd44572e53343a0f003b719cf438c6338bd29d9c-4157310944914892772",
	"signature":"1B.3B7636212157F57A934164D616EB497ED7C6B799B2934BC74F993A2A0B7220CA.5C8E2CD93DD3B4FFF3CE4E82176967EF0679B66A8C99995E3C5E8B2258A06C97"
	}

Response payload	{
	"authToken":"hFKNWhm/KaI9sC6INSPOS7kVRU1MkaEA35TE5eANLyE=",
	"clientId" : 1
	}

Signature generation details	
Private key	0x0000012001
Eth address	0xbd44572e53343A0f003b719cf438C6338bD29d9C

Message	{
	"types": {
	"EIP712Domain": [
	{"name": "name", "type": "string"},
	{"name": "version", "type": "string"},
	{"name": "chainId", "type": "uint256"}
	],
	"AuthMessage": [
	{"name": "content", "type": "string"}
	]
	},
	"primaryType": "AuthMessage",
	"domain": {
	"name": "Elektro",
	"version": "0.1",
	"chainId": 1337
	},
	"message": {
	"content": "auth-request-for-0xbd44572e53343a0f003b719cf438c6338bd29d9c-4157310944914892772"
	}
	}

Signature format	V.R.S
Signature	1B.84C55FA150A7AA145B21D2169AE31764B7CBAE768AC13A06613A979F0D7D95E5.3C98405BAF7196F801C11582BBD4D0690D3837F763C6828E22545AD2D05DE0A5

Get session info

Check session state. If the session is not authenticated, function returns UNAUTHORIZED(401).

Endpoint	GET /api/v1/auth/getSessionInfo
Request	Empty request body

Response payload	{
	“clientId” : “1”
	}

Signature example

Private key	0x75bcd78
Eth address	0x7fda7543e01caafd1af39585a156eadb3375d234

Nonce	auth-request-for-0x7fda7543e01caafd1af39585a156eadb3375d234--625753431108354586 1
Bytes to be signed in HEX	617574682d726571756573742d666f722d307837666461373534336530316361616643161663339353835613135366561646233333735640323334 2d2d363235373533343331313038333534353836 31
Signature V in HEX	1C
Signature R in HEX	DBA4A64802D6EE2D096831CA7AB7FEF785692428B2E116DDFA5C3BBF3CE5DC17
Signature S in HEX	60976FDAC389ED9D9C05022F98789B7935B36E13B5F43346E0484EBCF59440C9
Signature format	V.R.S
Signature	1C.DBA4A64802D6EE2D096831CA7AB7FEF785692428B2E116DDFA5C3BBF3CE5DC17.60976FDAC389ED9D9C05022F98789B7935B36E13B5F43346E0484EBCF59440C9

Logout

Endpoint	POST /api/v1/auth/logout
Request	Empty request body
Response payload	Empty payload

Web socket authentication

Use auth token received from REST authentication step

Request	{
	"action" : "VALIDATE_AUTH_TOKEN",
	"payload" : {
	"authToken" : "hFKNWhm/KaI9sC6INSPOS7kVRU1MkaEA35TE5eANLyE="
	}
	}

Response	{
	"error" : false,
	"responseType" : "VALIDATE_AUTH_TOKEN_RESPONSE",
	"payload" : {
	"clientId" : 101041
	}
	}

Orders submission

There are 2 order classes:

- Client order – single order (conditional order with only one leg)
- Conditional order – conditional order with multiple legs

After the order submission, backend will send execution reports through the authenticated client's websocket connections.

Fields

- clientOrderId should be based on current time in milliseconds left shifted to 10 bits. Last 10 bits might be filled with any value. clientOrderId should be unique, so if multiple orders are generated at the same time some sort of counter. Example:

System.currentTimeMillis() << 10 | cnt++

- Side – BUY or SELL
- Order type – only LIMIT is supported
- Time in force – supported values: DAY, GOOD_TILL_CANCEL, IMMEDIATE_OR_CANCEL, GOOD_TILL_DATE, AT_AUCTION_ONLY
- Contract id – id of the contract
- Signature is based on EIP-712 structure

Single order funds estimation

Endpoint	POST /api/v1/clientapi/estimateOrderLock

Request	{
	"clientOrderId" : 1604911104000001,
	"orderType" : "LIMIT",
	"timeInForce" : "GOOD_TILL_CANCEL",
	"clientEthAddress" : "0x7fda7543e01caafd1af39585a156eadb3375d234",
	"totalNetPrice" : 330.0000,
	"legs" : [{
	"contractId" : 1,
	"side" : "BUY",
	"quantity" : 30.00000000
	}]
	}

Response	{
	"result" : "OK",
	"details" : "",
	"payload" : {
	"currencyPair" : "WBTC/USDC",
	"underlierAmount" : 0E-8,
	"numeraireAmount" : 330.0000
	}
	}

Conditional order funds estimation

Endpoint	POST /api/v1/clientapi/estimateOrderLock

Request	{
	"clientOrderId" : 1604911104000001,
	"orderType" : "LIMIT",
	"timeInForce" : "GOOD_TILL_CANCEL",
	"clientEthAddress" : "0x7fda7543e01caafd1af39585a156eadb3375d234",
	"totalNetPrice" : 110.0000,
	"legs" : [{
	"contractId" : 4926,
	"side" : "BUY",
	"quantity" : 100.00000000,
	}, {
	"contractId" : 4925,
	"side" : "SELL",
	"quantity" : 100.00000000,
	}]
	}

Response	{ "result" : "OK",

	"details" : "",
	"payload" : {
	"currencyPair" : "WBTC/USDC",
	"underlierAmount" : 0E-8,
	"numeraireAmount" : 600110.0000
	}
	}

Order payoff estimation

Endpoint	POST /api/v1/clientapi/estimateOrderPayoff

Request	{
	"clientOrderId" : 1604911104000001,
	"orderType" : "LIMIT",
	"timeInForce" : "GOOD_TILL_CANCEL",
	"clientEthAddress" : "0x7fda7543e01caafd1af39585a156eadb3375d234",
	"totalNetPrice" : 110.0000,
	"legs" : [{
	"contractId" : 4926,
	"side" : "BUY",
	"quantity" : 100.00000000,
	}, {
	"contractId" : 4925,
	"side" : "SELL",
	"quantity" : 100.00000000,
	}]
	}

Response	{
	"result" : "OK",
	"details" : "",
	"payload" : {
	"4000.0000" : "2000.0000",
	"4500.0000" : "2100.0000",
	}
	}

Single order submission

Endpoint	POST /api/v1/clientapi/newOrder

Request	{
	"clientOrderId" : 1676258037557249,
	"orderType" : "LIMIT",
	"timeInForce" : "GOOD_TILL_CANCEL",
	"clientEthAddress" : "0xbd44572e53343a0f003b719cf438c6338bd29d9c",
	"totalNetPrice" : 330.0000,
	"legs" : [{
	"contractId" : 1,
	"side" : "BUY",
	"quantity" : 30.00000000
	}],
	"signature" :
	"1C.273FFD7E9ADA7C3DE33BA4B30958720F565CCBF2EE96353AED8D49687B5D05B8.
	04E416E3E081EEC7DE4CDC1357740BAA6C48BC5ABC3B761A25BEFA094F890352"
	}

Private key	0x0000012001
Eth address	0xbd44572e53343A0f003b719cf438C6338bD29d9C

Message	<pre>{
	"types": {
	"EIP712Domain": [
	{"name": "name", "type": "string"},
	{"name": "version", "type": "string"},
	{"name": "chainId", "type": "uint256"}
	],
	"ConditionalOrder": [
	{"name": "orderType", "type": "string"},
	{"name": "orderId", "type": "uint64"},
	{"name": "price", "type": "string"},
	{"name": "timeInForce", "type": "string"},
	{"name": "address", "type": "string"},
	{"name": "leg0", "type": "Leg"}
	],
	"Leg": [
	{"name": "contractId", "type": "uint32"},
	{"name": "side", "type": "string"},
	{"name": "quantity", "type": "string"}
	]
	},
	"primaryType": "ConditionalOrder",
	"domain": {
	"name": "Elektro",
	"version": "0.1",
	"chainId": 1
	},
	"message": {
	"orderType": "LIMIT",
	"orderId": 1676259483811841,
	"price": "330.00000000",
	"timeInForce": "GOOD_TILL_CANCEL",
	"address": "0xbd44572e53343a0f003b719cf438c6338bd29d9c",
	"leg0":{"contractId":1,"side":"BUY","quantity":"30.00000000"}
	}
	}</pre>

Signature format	V.R.S

Signature	1C.273FFD7E9ADA7C3DE33BA4B30958720F565CCBF2EE96353AED8D49687B5D05B8.0 4E416E3E081EEC7DE4CDC1357740BAA6C48BC5ABC3B761A25BEFA094F890352

Conditional order submission

Endpoint	POST /api/v1/clientapi/newOrder

Request	{
	"clientOrderId" : 1604911104000001,
	"orderType" : "LIMIT",
	"timeInForce" : "GOOD_TILL_CANCEL",
	"clientEthAddress" : "0xbd44572e53343a0f003b719cf438c6338bd29d9c",
	"totalNetPrice" : 110.0000,
	"legs" : [{
	"contractId" : 4926,
	"side" : "BUY",
	"quantity" : 100.00000000
	}, {
	"contractId" : 4925,
	"side" : "SELL",
	"quantity" : 100.00000000
	}]
	}

Private key	0x0000012001
Eth address	0xbd44572e53343A0f003b719cf438C6338bD29d9C
Note	Order legs should be sorted by contract id with natural order

Message	{
	"types": {
	"EIP712Domain": [
	{"name": "name", "type": "string"},
	{"name": "version", "type": "string"},
	{"name": "chainId", "type": "uint256"}
	],
	"ConditionalOrder": [
	{"name": "orderType", "type": "string"},
	{"name": "orderId", "type": "uint64"},
	{"name": "price", "type": "string"},
	{"name": "timeInForce", "type": "string"},
	{"name": "address", "type": "string"},
	{"name": "leg0", "type": "Leg"},{"name": "leg1", "type": "Leg"}
	],
	"Leg": [
	{"name": "contractId", "type": "uint32"},
	{"name": "side", "type": "string"},
	{"name": "quantity", "type": "string"}
	]
	},
	"primaryType": "ConditionalOrder",
	"domain": {
	"name": "Elektro",
	"version": "0.1",
	"chainId": 1337
	},
	"message": {
	"orderType": "LIMIT",
	"orderId": 1604911104000001,
	"price": "110.0000",
	"timeInForce": "GOOD_TILL_CANCEL",

	"address": "0xbd44572e53343a0f003b719cf438c6338bd29d9c", "leg0":{"contractId":4925,"side":"SELL","quantity":"100.00000000" },"leg1":{"contractId":4926,"side":"BUY","quantity":"100.00000000 "} } }
Signature format	V.R.S
Signature	1B.C04734C58C7467AF7D04579959D81DDD0B9F7F9E86417AA5E263E1C456D7D123.3 BA584CEC49A3328DC11A1FF88AAD912FC3671CC1F8E0CEA9328F332D3315E22

Order cancellation

Endpoint	POST /api/v1/clientapi/orderCancel
Request	{ "clientOrderId" : 1604911104000001, }

Order's status updates

Order's status update notifications come through the Web socket. Web socket should be authenticated. Statuses:

- REJECTED – Something went wrong, order was rejected. See ordRejReason for details
- PENDING_NEW – Waiting for validation
- NEW – Validation passed
- CANCELED – Order cancelled
- FILLED – Order matched and was completely filled
- PARTIALLY_FILLED – Order matched and was partially filled
- PARTIALLY_FILLED_REM_CANCELLED – Order matched and was partially filled, but remaining quantity was cancelled.
- CANCEL_REJECTED - Order cancellation rejected, order id is invalid or order has been fully filled
- SETTLE_PENDING – Transaction was submitted to the blockchain. See txHash field to find transaction hash. **Note:** single execution may have multiple SETTLE_NOTIFICATIONS with different tx hash. It is because we optimize cash movements on the backend and then send them to blockchain. When we have a lot of cash movements in the batch we may split them into multiple blockchain transactions.
- SETTLE_CONFIRMED – Transaction was successfully performed.
- SETTLE_FAILED – Transaction was reverted (usually because of block reorg). Transaction submission will be repeated
- SETTLE_REPLACED – Gas price was too low and transaction pended too long. Backend increased gas price and re-submitted transaction.

Client information

Client fund lock state

Endpoint	POST /api/v1/clientapi/clientFundLockState
Request	Empty
Response	{ "result" : "OK", "details" : "", "payload" : [{ "currency" : "USDC",

```
    "orderValue" : 0,
    "fundLockValue" : 0,
    "settleValue" : 0
  }, {
    "currency" : "WETH",
    "orderValue" : 0,
    "fundLockValue" : 0,
    "settleValue" : 0
  }, {
    "currency" : "WBTC",
    "orderValue" : 0,
    "fundLockValue" : 0,
    "settleValue" : 0
  } ]
}
```


Client open orders

Endpoint	POST /api/v1/clientapi/clientOpenOrders
Request	Empty

Response	{
	"result":"OK",
	"details":"",
	"payload":[
	{
	"clientOrderId":1601317888000003,
	"batchId":1563787000000,
	"orderType":"CONDITIONAL",
	"timeInForce":"GOOD_TILL_CANCEL",
	"totalNetPrice":260000.0000,
	"legs":[
	{
	"contractId":6732,
	"side":"SELL",
	"price":0.0000,
	"quantity":15.00000000,
	"filledQty":0E-8,
	"cancelledQty":0E-8
	},
	{
	"contractId":2000,
	"side":"BUY",
	"price":0.0000,
	"quantity":20.00000000,
	"filledQty":0E-8,
	"cancelledQty":0E-8
	}
	]
	}
	]
	}

Client positions

Endpoint	POST /api/v1/clientapi/clientCurrentPositions
Request	Empty
Response	{ "result":"OK", "details":"", "payload":[{ "contractId":6732,

	"positionsAvgPrice":100.0000, "positionsQty":-15.00000000 }] }

Order status

Endpoint	POST /api/v1/clientapi/orderStatus

Request	{
	"clientOrderId":1601317888000003
	}

Response	{
	"result":"OK",
	"details":"",
	"payload":{
	"orderId":1601317888000003,
	"clientId":1003,
	"orderStatus":"NEW",
	"netPrice":260000.0000,
	"details":[
	{
	"contractId":6732,
	"side":"SELL",
	"originalQty":15.00000000,
	"remainingQty":15.00000000,
	"cancelledQty":0E-8,
	"price":0,
	"totalCost":0.0000
	},
	{
	"contractId":2000,
	"side":"BUY",
	"originalQty":20.00000000,
	"remainingQty":20.00000000,
	"cancelledQty":0E-8,
	"price":0,
	"totalCost":0.0000
	}
	],
	"conditional":true
	}
	}

Trade history

Endpoint	POST /api/v1/clientapi/tradeHistory

Request	{
	"expiry":"20211126",
	"currencyPair":"WBTC/USDC"
	}

Note	Filter values are optional
Response	{ "result":"OK", "details":"", "payload":[{ "revDate":1637822220000, "orderId":1677130009950209, "clientId":419280940573952, "orderStatus":"PARTIALLY_FILLED",

<table>
<tr><td></td><td><pre>
          "details":[
             {
                "contractId":1000012,
                "side":"BUY",
                "currencyPair":"WBTC/USDC",
                "expiry":20211126,
                "originalQty":1.00000000,
                "remainingQty":0.50000000,
                "cancelledQty":0E-8,
                "price":1200.0000,
                "totalCost":600.0000,
                "originalPrice":1200.0000
             }
          ]
       },
       …
   ]
}
</pre></td></tr>
</table>

Common information

System info

Does not require authentication

Endpoint	POST /api/v1/clientapi/systemInfo
Request	Empty

Response	<pre>{
	  "result": "OK",
	  "details": "",
	  "payload": {
	  "chainId": 1,
	  "fundlockAddress": "${env:ETHEREUM_FUNDLOCK_ADDRESS}",
	  "tokenAddress": {
	  "WBTC": "0x2260fac5e5542a773aa44fbcfedf7c193bc2c599",
	  "WETH": "0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2",
	  "USDC": "0xa0b86991c6218b36c1d19d4a2e9eb0ce3606eb48"
	  },
	  "tokenDecimals": {
	  "WBTC": 8,
	  "WETH": 18,
	  "USDC": 6
	  },
	  "currencyPrecision": {
	  "WBTC": 8,
	  "WETH": 7,
	  "USDC": 4
	  }
	  }
	}</pre>

Contract list

Does not require authentication

Endpoint	POST /api/v1/clientapi/contractList
Request	Empty

Response	<pre>{
	  "result":"OK",
	  "details":"",</pre>

	"payload":[
	{
	"contractId":1,
	"payoff":"Call",
	"economics":{
	"currencyPair":"WBTC/USDC",
	"expiry":20220304,
	"strike":10.0000,
	"priceCurrency":"USDC",
	"qtyCurrency":"WBTC"
	}
	},
	…
	]
	}

Contract list by ids

Does not require authentication

Endpoint	POST /api/v1/clientapi/contractListByIds
Request	{"ids":[1,2,34]}

Response	{
	"result":"OK",
	"details":"",
	"payload":[
	{
	"contractId":1,
	"payoff":"Call",
	"economics":{
	"currencyPair":"WBTC/USDC",
	"expiry":20220304,
	"strike":10.0000,
	"priceCurrency":"USDC",
	"qtyCurrency":"WBTC"
	}
	},
	…
	]
	}

Reference price list

Endpoint	POST /api/v1/clientapi/referencePrices
Request	{ "expiry" : 20211105,

	"currencyPair":"WBTC/USDC" }

Response	{
	"result":"OK",
	"details":"",
	"payload":[
	{
	"contractId":1,
	"referencePrice":100.0,
	"lowRange":90.0,
	"highRange":120.0,
	"updateAt":"2021-11-03T13:01:30-07:00"
	},
	...
	]
	}

Spot price list

Endpoint	POST /api/v1/clientapi/spotPrices
Request	Empty

Response	{
	"result":"OK",
	"details":"",
	"payload":
	{
	"WBTC/USDC":60000.0000,
	"WETH/USDC":4000.0000,
	…
	}
	}

Next auction

Endpoint	POST /api/v1/clientapi/nextAuction
Request	Empty

Response	{
	"result":"OK",
	"details":"",
	"payload":1638388860000
	}

Subscribe orderbook

Does not require authentication

Endpoint	Web Sockets

Request	<pre>{
	"action":"SUBSCRIBE_ORDER_BOOK",
	"payload":{
	"contractId":1
	}
	}</pre>

Response	<pre>{
	"error":false,
	"responseType":"SUBSCRIBE_ORDER_BOOK_RESPONSE",
	"payload":{
	"contractId":1
	}
	}</pre>

Note	• Backend sends whole orderbook and then sends orderbook updates (delta) approximately once per second. • To update orderbook replace size field with the value from delta. • Since the price for conditional order is total, conditional order ASKs and BIDs are summarized in separate field. • Delta num increments monotonically. If the gap is detected, re-subscription is required.

Orderbook	<pre>{
	"error" : false,
	"responseType" : "ORDERBOOK",
	"payload" : {
	"contractId" : 2000,
	"deltaNum" : 11,
	"batchId" : 1563787000000,
	"asks":[
	{
	"price":100,
	"size":15
	}
	],
	"bids" : [
	{
	"price":95,
	"size":30
	}
	],
	"conditionalAskQty" : 0.0,
	"conditionalBidQty" : 20.0
	}
	}</pre>

Orderbook delta	<pre>{
	"error" : false,
	"responseType" : "ORDERBOOK_DELTA",
	"payload" : {
	"contractId" : 2000,
	"deltaNum" : 12,
	"batchId" : 1563787000000,
	"asks":[
	{
	"price":100,
	"size":15
	}
	],
	"bids" : [
	],
	"conditionalAskQty" : 0.0,
	"conditionalBidQty" : 10.0</pre>

	} }

Unsubscribe orderbook

Does not require authentication

Endpoint	Web Sockets

Request	{
	"action":"UNSUBSCRIBE_ORDER_BOOK",
	"payload":{
	"contractId":1
	}
	}

Response	{
	"error":false,
	"responseType":"UNSUBSCRIBE_TRADE_REPORT_RESPONSE ",
	"payload":{
	"contractId":1
	}
	}

Subscribe trade reports

Does not require authentication

Endpoint	Web Sockets

Request	{
	"action":"SUBSCRIBE_TRADE_REPORTS",
	"payload":{
	"contractId":1
	}
	}

Response	{
	"error":false,
	"responseType":"SUBSCRIBE_TRADE_REPORT_RESPONSE ",
	"payload":{
	"contractId":1
	}
	}

Trade report	{
	"error":false,
	"responseType":"TRADE_REPORT",
	"payload":{
	"contractId":1,
	"batchId":1563787005000,
	"tradeDetails":{
	"100":5
	}
	}
	}

Unsubscribe trade reports

Does not require authentication

Endpoint	Web Sockets

Request	{
	"action":"UNSUBSCRIBE_TRADE_REPORTS",
	"payload":{
	"contractId":1

	} }

Response	{
	"error":false,
	"responseType":"UNSUBSCRIBE_TRADE_REPORT_RESPONSE ",
	"payload":{
	"contractId":1
	}
	}

Heartbeat request

Does not require authentication

Endpoint	Web Sockets

Request	{
	"action":"HEARTBEAT_REQUEST",
	}

Response	{
	"error":false,
	"responseType":"HEARTBEAT_RESPONSE",
	}

Execute backend locally

- Install Ganache.
- Add Ganache folder to PATH variable
- Install Gurobi client
- Add Gurobi bin folder to PATH variable
- Create GUROBI_HOME environment variable
- Create LD_LIBRARY_PATH environment variable, it should contain path to the Gurobi bin folder
- It is recommended to install gurobi to the standard place (in Linux /opt/gurobi)
- Install Gurobi license: https://www.gurobi.com/documentation/9.1/remoteservices/licensing.html
- Install Java 11 or higher
- Execute: java -jar ui-test-0.1.011-alpha.jar ganache=true port=8545
- PC should be connected to internet
- Chain id for signing is 1
- Chain id to connect Ganache RPC: 1337

Endpoints

- REST http://localhost:8078
- WebSocket ws://localhost:8079

Clients

To simplify development process - we pre-created some accounts:

Private key	Funds	Deposited funds
0x0000012001	50000USDC, 1WBTC, 10WETH	50000USDC, 1WBTC, 10WETH
0x0000012002	50000USDC, 1WBTC, 10WETH	50000USDC, 1WBTC, 10WETH
0x0000012003	50000USDC, 1WBTC, 10WETH	50000USDC, 1WBTC, 10WETH
0x0000012004	50000USDC, 1WBTC, 10WETH	50000USDC, 1WBTC, 10WETH
0x0000012005	50000USDC, 1WBTC, 10WETH	0USDC, 0WBTC, 0WETH
0x0000012006	50000USDC, 1WBTC, 10WETH	0USDC, 0WBTC, 0WETH
0x0000012007	50000USDC, 1WBTC, 10WETH	0USDC, 0WBTC, 0WETH
0x0000012008	50000USDC, 1WBTC, 10WETH	0USDC, 0WBTC, 0WETH

Metamask

Metamask import key feature may say that the key length is invalid, just add leading zeros, so 0x0000012001 becomes 0x0000000000000000000000000000000000000000000000000000000012001

Configure Metamask network to be Localhost:8545, if it requests chain id set it to 1337.

Batch time

Batch time is 1 minute.

Troubleshooting

If Java application doesn't start, but previous execution worked.

May be ganache was not closed from previous run, please check running processes.

In Windows process name is **node.exe,** in Linux it should be **ganache** or **ganache-cli**.

Kill running ganache processes and try again.

Known issues

Metamask doesn't show ERC20 tokens balance even if you import tokens manually.

Execute test orders

If you need some orders to check trade history or notifications, execute:

```
java -jar test-client-0.1.025-alpha.jar cli cli_mode.properties
simpleconditionalordertest cancelconditionalordertest
```