TradeLocker REST API specification.

2.11.0

[ Base URL: demo.tradelocker.com/backend-api/ ]

Overview

TradeLocker users can interact with the demo.tradelocker.com and live.tradelocker.com environments using TradeLocker REST API and setting the corresponding Base URL to demo.tradelocker.com/backend-api/ or live.tradelocker.com/backend-api/. The API provides a request-response model, which allows external applications to create, modify and delete trade orders, as well as request info (historical quotes, list of orders and positions, balance info, etc.) from TradeLocker servers.

NEW: To start writing strategies, you can install the official TradeLocker API Python Client .

COMING SOON: Sign up to get access to TradeLocker Studio. Start using our AI to convert your ideas into running bots in seconds!

Authentication

TradeLocker uses JWT token authentication.

After completing the authorization process, you must send an accessToken with each request in the Authorization header: 'Authorization': 'Bearer {accessToken}'

JWT token request

To obtain an access token, a call to /auth/jwt/token should be sent using the following parameters:

email TradeLocker user's email.

password TradeLocker user's password.

server The name of the server that you connect to when logging into TradeLocker.

If all parameters are valid, TradeLocker server issues a new access token and a new refresh token, together with their expiration information.

Rate Limits

Each route has a rate limit that prevents too frequent querying. The exact values for each of the routes can be fetched from the /trade/config/ route, and require that you make a maximum of limit requests per each intervalNum SECONDS/MINUTES (e.g. max of 2 requests per second).

Important info

accNum

Every request to /trade/\* endpoints must also include accNum in the header. You can retrieve the list of all accounts, and their respective account numbers (accNum) from the /auth/jwt/all-accounts endpoint.

routeId

TradeLocker uses various connection identifiers (routeId) for fetching quote data (INFO) and making trade orders (TRADE). These can be user-specific and instrument-specific, so you should first fetch the list of all instruments, together with their corresponding INFO and TRADE routeId from the /trade/accounts/<{accountId}>/instruments endpoint.

Field names (/trade/config)

You can find the specification and names of fields returned when requesting positions, orders, ordersHistory, filledOrders and accountDetails, as well as rate limits for different routes and general limits by querying the /config endpoint.

Auth

JWT access and refresh tokens, list of all accounts.

POST

/auth/jwt/token

JWT tokens

Returns JWT accessToken and refreshToken.

Parameters

Name Description

body \*

object

(body)

User credentials - email, password, server

{

"email": "john@email.com",

"password": "secretPassword123",

"server": "SERVER"

}

Parameter content type

application/json

Responses

Response content type

application/json

Code Description

201

response

{

"accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ0cmFkZWxvY2tlci1hcGkiLCJhdWQiOiJ0cmFkZWxvY2tlci1hcGktdHJhZGVycyIsInR5cGUiOiJhY2Nlc3NfdG9rZW4iLCJzdWIiOiJPU1AjREVWQ0xMOWc0aXNUT0VpS2pmTmRKdzEzIiwidWlkIjoiMTQ0YmYyODctNTFlNC00YjQyLWJlNTQtYzBkZTRmMTE3ODMzIiwiYnJhbmQiOiJPU1AiLCJpYXQiOjE2ODUxMTY3OTMsImV4cCI6MTY4NTEyMDM5M30.cyDXRqUNVX6h5rtZb7m30vNIwEoYN7xUU2jfGM-Cf90",

"refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ0cmFkZWxvY2tlci1hcGkiLCJhdWQiOiJ0cmFkZWxvY2tlci1hcGktdHJhZGVycyIsInR5cGUiOiJyZWZyZXNoX3Rva2VuIiwic3ViIjoiT1NQI0RFVkNMTDlnNGlzVE9FaUtqZk5kSncxMyIsInVpZCI6IjE0NGJmMjg3LTUxZTQtNGI0Mi1iZTU0LWMwZGU0ZjExNzgzMyIsImJyYW5kIjoiT1NQIiwiaWF0IjoxNjg1MTE2NzkzLCJleHAiOjE2ODU3MjE1OTN9.GKNglolZzX76lKjTsrQ28MpmLTfU0A\_T7vCMrsojLcg"

}

400

Bad Request

POST

/auth/jwt/refresh

Refresh JWT tokens

Generate new JWT accessToken and refreshToken

Parameters

Name Description

body \*

object

(body)

Refresh token

{

"refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ0cmFkZWxvY2tlci1hcGkiLCJhdWQiOiJ0cmFkZWxvY2tlci1hcGktdHJhZGVycyIsInR5cGUiOiJyZWZyZXNoX3Rva2VuIiwic3ViIjoiT1NQI0RFVkNMTDlnNGlzVE9FaUtqZk5kSncxMyIsInVpZCI6IjE0NGJmMjg3LTUxZTQtNGI0Mi1iZTU0LWMwZGU0ZjExNzgzMyIsImJyYW5kIjoiT1NQIiwiaWF0IjoxNjg1MTE2NzkzLCJleHAiOjE2ODU3MjE1OTN9.GKNglolZzX76lKjTsrQ28MpmLTfU0A\_T7vCMrsojLcg"

}

Parameter content type

application/json

Responses

Response content type

application/json

Code Description

201

Returns accessToken and refreshToken

{

"accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ0cmFkZWxvY2tlci1hcGkiLCJhdWQiOiJ0cmFkZWxvY2tlci1hcGktdHJhZGVycyIsInR5cGUiOiJhY2Nlc3NfdG9rZW4iLCJzdWIiOiJPU1AjREVWQ0xMOWc0aXNUT0VpS2pmTmRKdzEzIiwidWlkIjoiMTQ0YmYyODctNTFlNC00YjQyLWJlNTQtYzBkZTRmMTE3ODMzIiwiYnJhbmQiOiJPU1AiLCJpYXQiOjE2ODUxMTY3OTMsImV4cCI6MTY4NTEyMDM5M30.cyDXRqUNVX6h5rtZb7m30vNIwEoYN7xUU2jfGM-Cf90",

"refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ0cmFkZWxvY2tlci1hcGkiLCJhdWQiOiJ0cmFkZWxvY2tlci1hcGktdHJhZGVycyIsInR5cGUiOiJyZWZyZXNoX3Rva2VuIiwic3ViIjoiT1NQI0RFVkNMTDlnNGlzVE9FaUtqZk5kSncxMyIsInVpZCI6IjE0NGJmMjg3LTUxZTQtNGI0Mi1iZTU0LWMwZGU0ZjExNzgzMyIsImJyYW5kIjoiT1NQIiwiaWF0IjoxNjg1MTE2NzkzLCJleHAiOjE2ODU3MjE1OTN9.GKNglolZzX76lKjTsrQ28MpmLTfU0A\_T7vCMrsojLcg"

}

400

Bad Request

GET

/auth/jwt/all-accounts

List all of the users accounts

List all of the users accounts, find the accNum for making requests to the /trade api

Parameters

Name Description

Authorization \*

string

(header)

Authorization token. Must be formatted as Bearer {accessToken}

Default value : Bearer {accessToken}

Bearer {accessToken}

Responses

Response content type

application/json

Code Description

201

response

[

{

"id": 7080,

"name": "BRAND1#DEVCLL9g4isTOEiKjfNdJw13#1#1",

"currency": "USD",

"status": 0,

"accNum": "1"

}

]

400

Bad Request

Config

Basic settings and schemas.

GET

/trade/config

Configuration

Parameters

Name Description

Authorization \*

string

(header)

Authorization token. Must be formatted as Bearer {accessToken}

Default value : Bearer {accessToken}

Bearer {accessToken}

accNum \*

integer($int32)

(header)

Account number

accNum

Responses

Response content type

application/json

Code Description

200

OK

{

"d": {

"accountDetailsConfig": {

"columns": [

{

"description": "string",

"id": "string"

}

],

"id": "string",

"title": "string"

},

"customerAccess": {

"filledOrders": true,

"marketDepth": true,

"orders": true,

"ordersHistory": true,

"positions": true,

"symbolInfo": true

},

"filledOrdersConfig": {

"columns": [

{

"description": "string",

"id": "string"

}

],

"id": "string",

"title": "string"

},

"ordersConfig": {

"columns": [

{

"description": "string",

"id": "string"

}

],

"id": "string",

"title": "string"

},

"ordersHistoryConfig": {

"columns": [

{

"description": "string",

"id": "string"

}

],

"id": "string",

"title": "string"

},

"positionsConfig": {

"columns": [

{

"description": "string",

"id": "string"

}

],

"id": "string",

"title": "string"

},

"rateLimits": {

"rateLimitType": "string",

"measure": "string",

"intervalNum": 0,

"limit": 0

},

"limits": {

"limitType": "string",

"limit": 0

}

},

"s": "ok"

}

401

Unauthorized

403

Forbidden

404

Not Found

Account

Account management and information.

GET

/trade/accounts

Accounts

Detailed information about the account selected by accNum.

Parameters

Name Description

Authorization \*

string

(header)

Authorization token. Must be formatted as Bearer {accessToken}

Default value : Bearer {accessToken}

Bearer {accessToken}

accNum \*

integer($int32)

(header)

Account number

accNum

Responses

Response content type

application/json

Code Description

200

response

{

"d": [

{

"currency": "JPY",

"id": "ACC-001",

"name": "Demo trading account",

"riskRules": {

"balanceRelativeDrawdown": 0,

"dailyLossLimit": {

"value": 0,

"warnLevel1": 0,

"warnLevel2": 0

},

"dailyProfitTarget": 0,

"maxDrawdownLevel": 0,

"maxOrderAmount": 0,

"maxOrderCapital": 0,

"maxOrdersCount": 0,

"maxPendingOrdersNumber": 0,

"maxPositionsNumber": 0,

"maxTrailingDrawdown": 0,

"positionLossLimit": 0,

"totalMaxPositionQty": 0,

"unrealizedLossLimit": {

"value": 0,

"warnLevel1": 0,

"warnLevel2": 0

},

"weeklyLossLimit": {

"value": 0,

"warnLevel1": 0,

"warnLevel2": 0

}

},

"status": "ACTIVE",

"tradingRules": {

"supportBrackets": true,

"supportPartialClosePosition": true,

"supportSelfTrading": true,

"supportTradingOutOfTradingHours": true,

"supportTrailingStop": true

},

"type": "demo"

}

],

"s": "ok"

}

401

Unauthorized

403

Forbidden

404

Not Found

GET

/trade/accounts/{accountId}/executions

Executions

Get the list of account's orders that were executed (filled/cancelled/rejected) during the current session. The column names are defined in the filledOrdersConfig part of the /trade/config route

Parameters

Name Description

accountId \*

integer($int32)

(path)

Account identifier.

accountId

Authorization \*

string

(header)

Authorization token. Must be formatted as Bearer {accessToken}

Default value : Bearer {accessToken}

Bearer {accessToken}

accNum \*

integer($int32)

(header)

Account number

accNum

Responses

Response content type

application/json

Code Description

200

response

{

"d": {

"executions": [

[

"string"

]

]

},

"s": "ok"

}

401

Unauthorized

403

Forbidden

404

Not Found

GET

/trade/accounts/{accountId}/instruments

Instruments

Get the list of the instruments that are available for trading with the specified account.

Parameters

Name Description

accountId \*

integer($int32)

(path)

Account identifier.

accountId

Authorization \*

string

(header)

Authorization token. Must be formatted as Bearer {accessToken}

Default value : Bearer {accessToken}

Bearer {accessToken}

accNum \*

integer($int32)

(header)

Account number

accNum

locale

string

(query)

Locale (language) id.

Available values : ar, en, es, fr, ja, ko, pl, pt, ru, tr, ua, zh\_sm, zh\_tr

--

Responses

Response content type

application/json

Code Description

200

response

{

"d": {

"instruments": [

{

"barSource": "ASK",

"continuous": true,

"contractMonth": "2025-01-01",

"country": 0,

"description": "string",

"hasDaily": true,

"hasIntraday": true,

"id": 0,

"localizedName": "string",

"logoUrl": "string",

"marketDataExchange": "string",

"name": "string",

"strikePrice": 0,

"strikeType": "CALL",

"tradableInstrumentId": 0,

"tradingExchange": "string",

"type": "CRYPTO",

"underlierId": 0

}

]

},

"s": "ok"

}

401

Unauthorized

403

Forbidden

404

Not Found

GET

/trade/accounts/{accountId}/orders

Orders

Get non-final orders for the account. The column names are defined in the ordersConfig part of the /trade/config route

Parameters

Name Description

accountId \*

integer($int32)

(path)

Account identifier.

accountId

Authorization \*

string

(header)

Authorization token. Must be formatted as Bearer {accessToken}

Default value : Bearer {accessToken}

Bearer {accessToken}

accNum \*

integer($int32)

(header)

Account number

accNum

from

integer($int64)

(query)

Unix timestamp in milliseconds (UTC) for the beginning of the interval. If not specified, all non-final orders are returned.

from

to

integer($int64)

(query)

Unix timestamp in milliseconds (UTC) for the end of the interval. If not specified, all non-final orders are returned.

to

tradableInstrumentId

integer($int64)

(query)

Instrument filter. If specified, only orders in this instrument will be returned.

tradableInstrumentId

Responses

Response content type

application/json

Code Description

200

response

{

"d": {

"orders": [

[

"string"

]

]

},

"s": "ok"

}

401

Unauthorized

403

Forbidden

404

Not Found

GET

/trade/accounts/{accountId}/ordersHistory

Orders History

Get order history for an account. Returns all user's orders with a final status (rejected, filled, canceled). The column names are defined in the ordersHistoryConfig part of the /trade/config route

Parameters

Name Description

accountId \*

integer($int32)

(path)

Account identifier.

accountId

Authorization \*

string

(header)

Authorization token. Must be formatted as Bearer {accessToken}

Default value : Bearer {accessToken}

Bearer {accessToken}

accNum \*

integer($int32)

(header)

Account number

accNum

from

integer($int64)

(query)

Unix timestamp in milliseconds (UTC) for the beginning of the interval. If not specified, all final orders are returned.

from

to

integer($int64)

(query)

Unix timestamp in milliseconds (UTC) for the end of the interval. If not specified, all final orders are returned.

to

tradableInstrumentId

integer($int64)

(query)

Instrument filter. If specified, only orders in this instrument will be returned.

tradableInstrumentId

Responses

Response content type

application/json

Code Description

200

response

{

"d": {

"ordersHistory": [

[

"string"

]

]

},

"s": "ok"

}

401

Unauthorized

403

Forbidden

404

Not Found

GET

/trade/accounts/{accountId}/positions

Positions

Get all currently open positions for an account. The column names are defined in the positionsConfig part of the /trade/config route

Parameters

Name Description

accountId \*

integer($int32)

(path)

Account identifier.

accountId

Authorization \*

string

(header)

Authorization token. Must be formatted as Bearer {accessToken}

Default value : Bearer {accessToken}

Bearer {accessToken}

accNum \*

integer($int32)

(header)

Account number

accNum

Responses

Response content type

application/json

Code Description

200

response

{

"d": {

"positions": [

[

"string"

]

]

},

"s": "ok"

}

401

Unauthorized

403

Forbidden

404

Not Found

GET

/trade/accounts/{accountId}/state

State

Get current account state, such as balance, available funds, PnL, etc. Field names can be fetched from /config route.

Parameters

Name Description

accountId \*

integer($int32)

(path)

Account identifier.

accountId

Authorization \*

string

(header)

Authorization token. Must be formatted as Bearer {accessToken}

Default value : Bearer {accessToken}

Bearer {accessToken}

accNum \*

integer($int32)

(header)

Account number

accNum

Responses

Response content type

application/json

Code Description

200

response

{

"d": {

"accountDetailsData": [

0

]

},

"s": "ok"

}

401

Unauthorized

403

Forbidden

404

Not Found

Instruments and Sessions

Detailed information about instruments and sessions.

GET

/trade/instruments/{tradableInstrumentId}

Detailed information about an instrument.

Get detailed instrument settings, such as lot steps and sizes, quoting currency, trade session id, etc.

Parameters

Name Description

tradableInstrumentId \*

integer($int64)

(path)

Identifier of the instrument that is used for trading purposes.

tradableInstrumentId

Authorization \*

string

(header)

Authorization token. Must be formatted as Bearer {accessToken}

Default value : Bearer {accessToken}

Bearer {accessToken}

accNum \*

integer($int32)

(header)

Account number

accNum

routeId \*

integer($int32)

(query)

Route identifier

routeId

locale

string

(query)

Locale (language) id.

Available values : ar, en, es, fr, ja, ko, pl, pt, ru, tr, ua, zh\_sm, zh\_tr

--

Responses

Response content type

application/json

Code Description

200

response

{

"d": {

"barSource": "ASK",

"baseCurrency": "string",

"betSize": 0,

"betStep": 0,

"bettingCurrency": "string",

"contractMonth": "2025-01-01",

"country": 0,

"deliveryStatus": "DELIVERED",

"description": "string",

"exerciseStyle": "AMERICAN",

"firstTradeDate": "2025-01-01",

"hasDaily": true,

"hasIntraday": true,

"industry": "string",

"isin": "string",

"lastTradeDate": "2025-01-01",

"localizedName": "string",

"logoUrl": "string",

"lotSize": 0,

"lotStep": 0,

"marketCap": 0,

"marketDataExchange": "string",

"maxLot": 0,

"minLot": 0,

"name": "string",

"noticeDate": "2025-01-01",

"quotingCurrency": "string",

"sector": "string",

"settlementDate": "2025-01-01",

"settlementSystem": "Immediate",

"strikePrice": 0,

"strikeType": "CALL",

"symbolStatus": "CLOSED",

"tickCost": [

{

"leftRangeLimit": 0,

"tickCost": 0

}

],

"tickSize": [

{

"leftRangeLimit": 0,

"tickSize": 0

}

],

"tradeSessionId": 0,

"tradeSessionStatusId": 0,

"tradingExchange": "string",

"type": "CRYPTO"

},

"s": "ok"

}

401

Unauthorized

403

Forbidden

404

Not Found

GET

/trade/sessions/{sessionId}

Detailed information about a trade session

Get detailed information about a specific trade session.

Parameters

Name Description

sessionId \*

integer($int32)

(path)

Identifier of the trade session.

sessionId

Authorization \*

string

(header)

Authorization token. Must be formatted as Bearer {accessToken}

Default value : Bearer {accessToken}

Bearer {accessToken}

accNum \*

integer($int32)

(header)

Account number

accNum

Responses

Response content type

application/json

Code Description

200

response

{

"d": {

"blockTrading": true,

"holidays": [

{

"date": "2025-01-01",

"name": "string",

"session\_aftermarket": {

"sub\_type": [

{

"name": "string",

"sessionTime": [

{

"endTime": "string",

"startTime": "string"

}

]

}

]

},

"session\_beforemarket": {

"sub\_type": [

{

"name": "string",

"sessionTime": [

{

"endTime": "string",

"startTime": "string"

}

]

}

]

},

"session\_postclose": {

"sub\_type": [

{

"name": "string",

"sessionTime": [

{

"endTime": "string",

"startTime": "string"

}

]

}

]

},

"session\_preopen": {

"sub\_type": [

{

"name": "string",

"sessionTime": [

{

"endTime": "string",

"startTime": "string"

}

]

}

]

},

"session\_regular": {

"sub\_type": [

{

"name": "string",

"sessionTime": [

{

"endTime": "string",

"startTime": "string"

}

]

}

]

},

"workingState": "CLOSE"

}

],

"session\_aftermarket": {

"sub\_type": [

{

"name": "string",

"sessionTime": [

{

"endTime": "string",

"startTime": "string"

}

]

}

]

},

"session\_beforemarket": {

"sub\_type": [

{

"name": "string",

"sessionTime": [

{

"endTime": "string",

"startTime": "string"

}

]

}

]

},

"session\_postclose": {

"sub\_type": [

{

"name": "string",

"sessionTime": [

{

"endTime": "string",

"startTime": "string"

}

]

}

]

},

"session\_preopen": {

"sub\_type": [

{

"name": "string",

"sessionTime": [

{

"endTime": "string",

"startTime": "string"

}

]

}

]

},

"session\_regular": {

"sub\_type": [

{

"name": "string",

"sessionTime": [

{

"endTime": "string",

"startTime": "string"

}

]

}

]

},

"timeZone": "string"

},

"s": "ok"

}

401

Unauthorized

403

Forbidden

404

Not Found

GET

/trade/sessionStatuses/{sessionStatusId}

Info about allowed orders in this session.

Get information about the orders operations (new, modify, cancel) and types (manual, market, stop, limit, stop limit, trailing stop) allowed in some session, given its session status.

Parameters

Name Description

sessionStatusId \*

integer($int64)

(path)

Identifier of the trade session status.

sessionStatusId

Authorization \*

string

(header)

Authorization token. Must be formatted as Bearer {accessToken}

Default value : Bearer {accessToken}

Bearer {accessToken}

accNum \*

integer($int32)

(header)

Account number

accNum

Responses

Response content type

application/json

Code Description

200

response

{

"d": {

"allowedOperations": [

0

],

"allowedOrderTypes": [

0

]

},

"s": "ok"

}

401

Unauthorized

403

Forbidden

404

Not Found

Market data

Price and other related data for a financial instrument.

GET

/trade/dailyBar

DailyBar

Get current daily bar.

Parameters

Name Description

Authorization \*

string

(header)

Authorization token. Must be formatted as Bearer {accessToken}

Default value : Bearer {accessToken}

Bearer {accessToken}

accNum \*

integer($int32)

(header)

Account number

accNum

routeId \*

integer($int32)

(query)

Route identifier.

routeId

barType \*

string

(query)

Defines the OHCL data source.

Available values : ASK, BID, TRADE

ASK

tradableInstrumentId \*

integer($int64)

(query)

Identifier of the instrument that is used for trading purposes.

tradableInstrumentId

Responses

Response content type

application/json

Code Description

200

response

{

"d": {

"c": 0,

"h": 0,

"l": 0,

"o": 0,

"v": 0

},

"s": "ok"

}

401

Unauthorized

403

Forbidden

404

Not Found

GET

/trade/depth

Depth

Get the current depth of the market for the instrument.

Parameters

Name Description

Authorization \*

string

(header)

Authorization token. Must be formatted as Bearer {accessToken}

Default value : Bearer {accessToken}

Bearer {accessToken}

accNum \*

integer($int32)

(header)

Account number

accNum

routeId \*

integer($int32)

(query)

Route identifier.

routeId

tradableInstrumentId \*

integer($int64)

(query)

Identifier of the instrument that is used for trading purposes.

tradableInstrumentId

Responses

Response content type

application/json

Code Description

200

response

{

"d": {

"asks": [

[

0

]

],

"bids": [

[

0

]

]

},

"s": "ok"

}

401

Unauthorized

403

Forbidden

404

Not Found

GET

/trade/history

History

Get the historical bars for an instrument. Any request that would result in a response with more than 20,000 bars will be rejected.

Each property of the response object is a table column. Bar time for daily bars is 00:00 UTC. Bar time for monthly bars is 00:00 UTC and is the first trading day of the month. If there is no data in the requested time period but there is data in the previous time period the status code will be set to no\_data and the nb property to UNIX timestamp of the next available bar behind the range. If there is no data in the requested and previous time periods the status code will be set to no\_data.

Parameters

Name Description

Authorization \*

string

(header)

Authorization token. Must be formatted as Bearer {accessToken}

Default value : Bearer {accessToken}

Bearer {accessToken}

accNum \*

integer($int32)

(header)

Account number

accNum

routeId \*

integer($int32)

(query)

Route identifier.

routeId

from \*

integer($int64)

(query)

Unix timestamp in milliseconds (UTC) of the leftmost required bar, including from.

from

resolution \*

string

(query)

Symbol resolution. Possible resolutions are monthly (1M), weekly (1W), daily (1D), hourly (4H, 1H) in minutes (30m, 15m, 5m, 1m)

Available values : 1M, 1W, 1D, 4H, 1H, 30m, 15m, 5m, 1m

1M

to \*

integer($int64)

(query)

Unix timestamp in milliseconds (UTC) of the rightmost required bar, inclusive. If in the future, the rightmost returned bar will be the latest available in the system.

to

tradableInstrumentId \*

integer($int64)

(query)

Identifier of the instrument that is used for trading purposes.

tradableInstrumentId

Responses

Response content type

application/json

Code Description

200

response

{

"d": {

"barDetails": [

{

"c": 0,

"h": 0,

"l": 0,

"o": 0,

"t": 0,

"v": 0

}

]

},

"s": "ok"

}

401

Unauthorized

403

Forbidden

404

Not Found

GET

/trade/quotes

Quotes

Get current prices of the instrument.

Parameters

Name Description

Authorization \*

string

(header)

Authorization token. Must be formatted as Bearer {accessToken}

Default value : Bearer {accessToken}

Bearer {accessToken}

accNum \*

integer($int32)

(header)

Account number

accNum

routeId \*

integer($int32)

(query)

Route identifier.

routeId

tradableInstrumentId \*

integer($int64)

(query)

Identifier of the instrument that is used for trading purposes.

tradableInstrumentId

Responses

Response content type

application/json

Code Description

200

response

{

"d": {

"ap": 0,

"as": 0,

"bp": 0,

"bs": 0

},

"s": "ok"

}

401

Unauthorized

403

Forbidden

404

Not Found

Trading

Creating, modifying and deleting orders.

POST

/trade/accounts/{accountId}/orders

Place a new order.

Place a new order.

Fields qty, routeId, side, validity, type, tradableInstrumentId are mandatory inside of the request body. Price can be set to 0 for market orders.

If using a stop type of order, you must specify the stopPrice.

Validity (also known as TimeInForce - TIF in some error messages) must be IOC for market orders and GTC for limit and stop orders.

For more details on each of the fields in the body, click on Model (next to 'Example Value') and expand it by clicking on three dots [...]

Parameters

Name Description

accountId \*

integer($int32)

(path)

Account identifier.

accountId

Authorization \*

string

(header)

Authorization token. Must be formatted as Bearer {accessToken}

Default value : Bearer {accessToken}

Bearer {accessToken}

accNum \*

integer($int32)

(header)

Account number

accNum

body \*

object

(body)

Request body with parameters

{

"price": 0,

"qty": 0,

"routeId": 0,

"side": "buy",

"strategyId": "string",

"stopLoss": 0,

"stopLossType": "absolute",

"stopPrice": 0,

"takeProfit": 0,

"takeProfitType": "absolute",

"trStopOffset": 0,

"tradableInstrumentId": 0,

"type": "limit",

"validity": "GTC"

}

Parameter content type

application/json

Responses

Response content type

application/json

Code Description

200

response

{

"d": {

"orderId": "1"

},

"s": "ok"

}

201

Created

401

Unauthorized

403

Forbidden

404

Not Found

DELETE

/trade/accounts/{accountId}/orders

Cancel all orders.

Cancel all existing orders.

Parameters

Name Description

accountId \*

integer($int32)

(path)

Account identifier.

accountId

Authorization \*

string

(header)

Authorization token. Must be formatted as Bearer {accessToken}

Default value : Bearer {accessToken}

Bearer {accessToken}

accNum \*

integer($int32)

(header)

Account number

accNum

tradableInstrumentId \*

integer($int64)

(query)

Instrument filter.

If specified, only orders in this instrument will be deleted.

tradableInstrumentId

Responses

Response content type

application/json

Code Description

200

response

{

"s": "ok"

}

204

No Content

401

Unauthorized

403

Forbidden

DELETE

/trade/accounts/{accountId}/positions

Place orders to close all positions.

Place closing orders for all open positions. Isn't guaranteed to close all positions, or close them immediately. Will attempt to place an IOC, then GTC closing order, so the execution might be delayed.

Parameters

Name Description

accountId \*

integer($int32)

(path)

Account identifier.

accountId

Authorization \*

string

(header)

Authorization token. Must be formatted as Bearer {accessToken}

Default value : Bearer {accessToken}

Bearer {accessToken}

accNum \*

integer($int32)

(header)

Account number

accNum

tradableInstrumentId \*

integer($int64)

(query)

Instrument filter.

If specified, only positions in this instrument will be closed.

tradableInstrumentId

Responses

Response content type

application/json

Code Description

200

response

{

"d": {

"positions": [

[

"string"

]

]

},

"s": "ok"

}

401

Unauthorized

403

Forbidden

404

Not Found

DELETE

/trade/orders/{orderId}

Cancel an existing order.

Cancel an existing order. Only available before an order is executed.

Parameters

Name Description

orderId \*

integer($int64)

(path)

Order identifier.

orderId

Authorization \*

string

(header)

Authorization token. Must be formatted as Bearer {accessToken}

Default value : Bearer {accessToken}

Bearer {accessToken}

accNum \*

integer($int32)

(header)

Account number

accNum

Responses

Response content type

application/json

Code Description

200

response

{

"s": "ok"

}

204

No Content

401

Unauthorized

403

Forbidden

PATCH

/trade/orders/{orderId}

Modify an existing order.

Modify an existing order.

For more details on each of the fields in the body, look at their definitions in the POST /trade/accounts/{accountId}/orders route.

Parameters

Name Description

orderId \*

integer($int64)

(path)

Order identifier.

orderId

Authorization \*

string

(header)

Authorization token. Must be formatted as Bearer {accessToken}

Default value : Bearer {accessToken}

Bearer {accessToken}

accNum \*

integer($int32)

(header)

Account number

accNum

body \*

object

(body)

Request body with parameters

{

"price": 0,

"qty": 0,

"stopLoss": 0,

"stopLossType": "absolute",

"stopPrice": 0,

"takeProfit": 0,

"takeProfitType": "absolute",

"trStopOffset": 0,

"validity": "GTC"

}

Parameter content type

application/json

Responses

Response content type

application/json

Code Description

200

response

{

"s": "ok"

}

204

No Content

401

Unauthorized

403

Forbidden

DELETE

/trade/positions/{positionId}

Place an order to close an existing position.

Place an order to close an existing position. Isn't guaranteed to close the positions, or close it immediately. Will attempt to place an IOC, then GTC closing order, so the execution might be delayed.

Parameters

Name Description

positionId \*

integer($int64)

(path)

Position identifier.

positionId

Authorization \*

string

(header)

Authorization token. Must be formatted as Bearer {accessToken}

Default value : Bearer {accessToken}

Bearer {accessToken}

accNum \*

integer($int32)

(header)

Account number

accNum

body \*

object

(body)

Request body with parameters

{

"qty": 0

}

Parameter content type

application/json

Responses

Response content type

application/json

Code Description

200

response

{

"s": "ok"

}

204

No Content

401

Unauthorized

403

Forbidden

PATCH

/trade/positions/{positionId}

Modify an existing position.

Modify an existing position's stop loss, take profit, or both.

Parameters

Name Description

positionId \*

integer($int64)

(path)

Position identifier.

positionId

Authorization \*

string

(header)

Authorization token. Must be formatted as Bearer {accessToken}

Default value : Bearer {accessToken}

Bearer {accessToken}

accNum \*

integer($int32)

(header)

Account number

accNum

body \*

object

(body)

Request body with parameters

{

"stopLoss": 0,

"takeProfit": 0,

"trailingOffset": 0

}

Parameter content type

application/json

Responses

Response content type

application/json

Code Description

200

response

{

"s": "ok"

}

204

No Content

401

Unauthorized

403

Forbidden

Models

Account

{

currency string

example: JPY

Abbreviation of account currency.

id\* string

example: ACC-001

Unique account identifier.

name\* string

example: Demo trading account

Account title that is displayed to a user.

riskRules RiskRules

{

balanceRelativeDrawdown number

The minimum allowable Projected balance value that the account can have. Max drawdown value calculates as % from the current Balance value

dailyLossLimit LossLimitRule

{

value number

value

warnLevel1 number

WarnLevel1

warnLevel2 number

WarnLevel2

}

dailyProfitTarget number

A maximum value of daily net profit allowed for an account

maxDrawdownLevel number

The minimal allowed Projected balance value in the account currency for the account not to trigger a Stop out

maxOrderAmount number

The maximum amount for a single order measured in the instrument units

maxOrderCapital number

The maximum order capital which can be sent by user. Order capital = Qty. \* Lot size \* Price \* Cross price

maxOrdersCount number

The maximum number of orders that user can place or modify during a day

maxPendingOrdersNumber number

The maximum quantity of pending orders which can be placed by account. Closing SL/TP orders aren’t taken into consideration in the counter, and are accepted without restrictions

maxPositionsNumber number

The maximum number of positions which can be opened by one account. Closing orders (Market and SL/TP) are accepted without checking and restrictions

maxTrailingDrawdown number

Maximum drawdown level for an account. A reached 'Max trailing drawdown' blocks trading

positionLossLimit number

Loss limit value for the position in percent

totalMaxPositionQty number

Maximum possible qty in lots of all positions and orders for all trading instruments. A reached 'Total max position qty' blocks an order placement and an order modification

unrealizedLossLimit LossLimitRule

{

value number

value

warnLevel1 number

WarnLevel1

warnLevel2 number

WarnLevel2

}

weeklyLossLimit LossLimitRule

{

value number

value

warnLevel1 number

WarnLevel1

warnLevel2 number

WarnLevel2

}

}

status string

Status of the account

Enum:

Array [ 6 ]

tradingRules TradingRules

{

supportBrackets boolean

Whether the brackets (take profit and stop loss) for orders and positions are supported

supportPartialClosePosition boolean

Supporting partial closing of a position.

supportSelfTrading boolean

Whether the user is allowed to trade on this account

supportTradingOutOfTradingHours boolean

Whether placing orders out of trading hours is allowed for the account

supportTrailingStop boolean

Whether the trailing stop orders are supported

}

type string

Enum:

Array [ 2 ]

}

Account details data

{

accountDetailsData

[number]

}

AccountResponse

{

d\*

[Account

{

currency string

example: JPY

Abbreviation of account currency.

id\* string

example: ACC-001

Unique account identifier.

name\* string

example: Demo trading account

Account title that is displayed to a user.

riskRules RiskRules

{

balanceRelativeDrawdown number

The minimum allowable Projected balance value that the account can have. Max drawdown value calculates as % from the current Balance value

dailyLossLimit LossLimitRule

{

value number

value

warnLevel1 number

WarnLevel1

warnLevel2 number

WarnLevel2

}

dailyProfitTarget number

A maximum value of daily net profit allowed for an account

maxDrawdownLevel number

The minimal allowed Projected balance value in the account currency for the account not to trigger a Stop out

maxOrderAmount number

The maximum amount for a single order measured in the instrument units

maxOrderCapital number

The maximum order capital which can be sent by user. Order capital = Qty. \* Lot size \* Price \* Cross price

maxOrdersCount number

The maximum number of orders that user can place or modify during a day

maxPendingOrdersNumber number

The maximum quantity of pending orders which can be placed by account. Closing SL/TP orders aren’t taken into consideration in the counter, and are accepted without restrictions

maxPositionsNumber number

The maximum number of positions which can be opened by one account. Closing orders (Market and SL/TP) are accepted without checking and restrictions

maxTrailingDrawdown number

Maximum drawdown level for an account. A reached 'Max trailing drawdown' blocks trading

positionLossLimit number

Loss limit value for the position in percent

totalMaxPositionQty number

Maximum possible qty in lots of all positions and orders for all trading instruments. A reached 'Total max position qty' blocks an order placement and an order modification

unrealizedLossLimit LossLimitRule

{

value number

value

warnLevel1 number

WarnLevel1

warnLevel2 number

WarnLevel2

}

weeklyLossLimit LossLimitRule

{

value number

value

warnLevel1 number

WarnLevel1

warnLevel2 number

WarnLevel2

}

}

status string

Status of the account

Enum:

Array [ 6 ]

tradingRules TradingRules

{

supportBrackets boolean

Whether the brackets (take profit and stop loss) for orders and positions are supported

supportPartialClosePosition boolean

Supporting partial closing of a position.

supportSelfTrading boolean

Whether the user is allowed to trade on this account

supportTradingOutOfTradingHours boolean

Whether placing orders out of trading hours is allowed for the account

supportTrailingStop boolean

Whether the trailing stop orders are supported

}

type string

Enum:

Array [ 2 ]

}]

s\* string

example: ok

Status will always be ok.

}

AccountStateResponse

{

description:

Account state response

d\* Account details data

{

accountDetailsData

[number]

}

s\* string

example: ok

Status will always be ok.

}

Bar Details

{

c number

Close price.

h number

High price.

l number

Low price.

o number

Open price.

t integer($int64)

Unix timestamp in milliseconds (UTC) of the rightmost required bar, including to. It can be in the future. In this case, the rightmost required bar is the latest available bar.

v number

Volume.

}

Column

{

description string

Column description.

id string

Column identifier.

}

Config

{

accountDetailsConfig PanelConfig{...}

customerAccess Permissions to use specific functions and information in the client terminal.

{

filledOrders boolean

Whether the Executions tab is supported. If set to false, the /executions endpoint should not be used.

marketDepth boolean

Whether the Market depth is supported. If set to false, the /depth endpoint should not be used.

orders boolean

Whether the Orders tab is supported. If set to false, the /orders endpoint should not be used.

ordersHistory boolean

Whether the Orders history tab is supported. If set to false, the /ordersHistory endpoint should not be used

positions boolean

Whether the Positions tab is supported. If set to false, the /positions endpoint should not be used.

symbolInfo boolean

Whether the Symbol info is supported. If set to false, the symbol\_info endpoint should not be used.

}

filledOrdersConfig PanelConfig

{

columns

[Column

{

description string

Column description.

id string

Column identifier.

}]

id string

Panel identifier.

title string

Localized title of a table

}

ordersConfig PanelConfig

{

columns

[Column

{

description string

Column description.

id string

Column identifier.

}]

id string

Panel identifier.

title string

Localized title of a table

}

ordersHistoryConfig PanelConfig

{

columns

[Column

{

description string

Column description.

id string

Column identifier.

}]

id string

Panel identifier.

title string

Localized title of a table

}

positionsConfig PanelConfig

{

columns

[Column

{

description string

Column description.

id string

Column identifier.

}]

id string

Panel identifier.

title string

Localized title of a table

}

rateLimits RateLimits

{

rateLimitType string

Limit type

measure string

Frequency measurement unit (SECONDS or MINUTES)

intervalNum integer

How many SECONDS/MINUTES are in the measured interval

limit integer

Limit value (maximum number of allowed times in intervalNum SECONDS/MINUTES)

}

limits Limits

{

limitType string

Limit type

limit integer

Limit value

}

}

ConfigResponse

{

d Config

{

accountDetailsConfig PanelConfig

{

columns

[Column

{

description string

Column description.

id string

Column identifier.

}]

id string

Panel identifier.

title string

Localized title of a table

}

customerAccess Permissions to use specific functions and information in the client terminal.

{

filledOrders boolean

Whether the Executions tab is supported. If set to false, the /executions endpoint should not be used.

marketDepth boolean

Whether the Market depth is supported. If set to false, the /depth endpoint should not be used.

orders boolean

Whether the Orders tab is supported. If set to false, the /orders endpoint should not be used.

ordersHistory boolean

Whether the Orders history tab is supported. If set to false, the /ordersHistory endpoint should not be used

positions boolean

Whether the Positions tab is supported. If set to false, the /positions endpoint should not be used.

symbolInfo boolean

Whether the Symbol info is supported. If set to false, the symbol\_info endpoint should not be used.

}

filledOrdersConfig PanelConfig

{

columns

[Column

{

description string

Column description.

id string

Column identifier.

}]

id string

Panel identifier.

title string

Localized title of a table

}

ordersConfig PanelConfig

{

columns

[Column

{

description string

Column description.

id string

Column identifier.

}]

id string

Panel identifier.

title string

Localized title of a table

}

ordersHistoryConfig PanelConfig

{

columns

[Column

{

description string

Column description.

id string

Column identifier.

}]

id string

Panel identifier.

title string

Localized title of a table

}

positionsConfig PanelConfig

{

columns

[Column

{

description string

Column description.

id string

Column identifier.

}]

id string

Panel identifier.

title string

Localized title of a table

}

rateLimits RateLimits

{

rateLimitType string

Limit type

measure string

Frequency measurement unit (SECONDS or MINUTES)

intervalNum integer

How many SECONDS/MINUTES are in the measured interval

limit integer

Limit value (maximum number of allowed times in intervalNum SECONDS/MINUTES)

}

limits Limits

{

limitType string

Limit type

limit integer

Limit value

}

}

s\* string

example: ok

Status will always be ok.

}

Daily bar

{

c number

Close.

h number

High.

l number

Low.

o number

Open.

v number

Volume.

}

DailyBarResponse

{

description:

Daily bar response

d\* Daily bar

{

c number

Close.

h number

High.

l number

Low.

o number

Open.

v number

Volume.

}

s\* string

example: ok

Status will always be ok.

}

Depth

{

description:

Depth of market for an instrument.

asks\*

[

Array of arrays with two numeric elements - price and volume. Must be sorted by price in asc order.

[number]]

bids\*

[

Array of arrays with two numeric elements - price and volume. Must be sorted by price in asc order.

[number]]

}

DepthResponse

{

d\* Depth

{

description:

Depth of market for an instrument.

asks\*

[

Array of arrays with two numeric elements - price and volume. Must be sorted by price in asc order.

[number]]

bids\*

[

Array of arrays with two numeric elements - price and volume. Must be sorted by price in asc order.

[number]]

}

s\* string

example: ok

Status will always be ok.

}

ErrorResponse

{

errmsg\* string

example: An error occurred.

Error message.

s\* string

example: error

Status will always be error.

}

Executions

{

executions

[

[string]]

}

ExecutionsResponse

{

description:

Executions response

d\* Executions

{

executions

[

[string]]

}

s\* string

example: ok

Status will always be ok.

}

HistoryResponse

{

barDetails

[

Bar details

Bar Details

{

c number

Close price.

h number

High price.

l number

Low price.

o number

Open price.

t integer($int64)

Unix timestamp in milliseconds (UTC) of the rightmost required bar, including to. It can be in the future. In this case, the rightmost required bar is the latest available bar.

v number

Volume.

}]

}

HistorySuccessResponse

{

description:

History success response

d\* HistoryResponse

{

barDetails

[

Bar details

Bar Details

{

c number

Close price.

h number

High price.

l number

Low price.

o number

Open price.

t integer($int64)

Unix timestamp in milliseconds (UTC) of the rightmost required bar, including to. It can be in the future. In this case, the rightmost required bar is the latest available bar.

v number

Volume.

}]

}

s\* string

example: ok

Status will always be ok.

}

Holiday

{

date string($date)

Holiday date.

name string

Holiday name.

session\_aftermarket SessionPeriod

{

sub\_type

[SubType

{

name string

sessionTime

[

Trading hours. Next sequence is used: sun, mon, tue, wed, thu, fri, sat, shortened day

SessionTime

{

endTime string

Trading period end time

startTime string

Trading period start time.

}]

}]

}

session\_beforemarket SessionPeriod

{

sub\_type

[SubType

{

name string

sessionTime

[

Trading hours. Next sequence is used: sun, mon, tue, wed, thu, fri, sat, shortened day

SessionTime

{

endTime string

Trading period end time

startTime string

Trading period start time.

}]

}]

}

session\_postclose SessionPeriod

{

sub\_type

[SubType

{

name string

sessionTime

[

Trading hours. Next sequence is used: sun, mon, tue, wed, thu, fri, sat, shortened day

SessionTime

{

endTime string

Trading period end time

startTime string

Trading period start time.

}]

}]

}

session\_preopen SessionPeriod

{

sub\_type

[SubType

{

name string

sessionTime

[

Trading hours. Next sequence is used: sun, mon, tue, wed, thu, fri, sat, shortened day

SessionTime

{

endTime string

Trading period end time

startTime string

Trading period start time.

}]

}]

}

session\_regular SessionPeriod

{

sub\_type

[SubType

{

name string

sessionTime

[

Trading hours. Next sequence is used: sun, mon, tue, wed, thu, fri, sat, shortened day

SessionTime

{

endTime string

Trading period end time

startTime string

Trading period start time.

}]

}]

}

workingState string

Working state of the holiday.

Enum:

Array [ 3 ]

}

Instrument

{

barSource string

The principle of building bars. Available values

Enum:

Array [ 3 ]

continuous boolean

Defines whether the contract is continuous.

contractMonth string($date)

Final day for delivery.

country integer($int32)

The country for the instrument.

description string

Any useful information about the instrument.

hasDaily boolean

Boolean value showing whether the symbol includes daily bars historical data.

hasIntraday boolean

Boolean value showing whether the symbol includes intraday (minutes) historical data.

id\* integer($int32)

Identifier of the instrument.

localizedName string

The name of the instrument on the specified language in request.

logoUrl string

URL to get instrument logo.

marketDataExchange\* string

The name of the exchange, which provides market data for the instrument.

name\* string

Name of the instrument.

strikePrice number

Price at which a derivative contract can be bought or sold when it is exercised.

strikeType string

Put or Call

Enum:

Array [ 2 ]

tradableInstrumentId\* integer($int64)

Identifier of the instrument that is used for trading purposes.

tradingExchange\* string

The name of the exchange, on which the instrument trade will be performed.

type\* string

Symbol type.

Enum:

Array [ 10 ]

underlierId integer($int32)

Identifier of the base asset

}

Instrument details

{

barSource string

The principle of building bars. Available values

Enum:

Array [ 3 ]

baseCurrency string

For currency pairs only. This field contains the first currency of the pair.

betSize number

The standard bet size for the instrument with spreadbet type in units.

betStep number

The minimal bet change of quantity in the betting currency. Required for spreadbet type symbols only.

bettingCurrency string

The currency in which the instrument will be traded. Required for spreadbet type symbols only.

contractMonth string($date)

Final day for delivery.

country integer($int32)

The country for the instrument.

deliveryStatus string

A contract delivery status.

Enum:

Array [ 3 ]

description string

Any useful information about the instrument.

exerciseStyle string

Defines the date on which the option may be exercised.

Enum:

Array [ 2 ]

firstTradeDate string($date)

The first day of trading on exchange

hasDaily boolean

Boolean value showing whether the symbol includes daily bars historical data.

hasIntraday boolean

Boolean value showing whether the symbol includes intraday (minutes) historical data.

industry string

The Industry to which the instrument belongs.

isin string

The International Securities Identification Number (ISIN) for the instrument.

lastTradeDate string($date)

The last day of trading on exchange. At the end of the session all orders will be

canceled and positions will be blocked (a trader won't be able to close them).

localizedName string

The name of the instrument on the specified language in request.

logoUrl string

URL to get instrument logo.

lotSize number

The standard lot size for the instrument in units.

lotStep number

The minimal lot quantity change.

marketCap number($double)

The market capitalization of the Instrument.

marketDataExchange\* string

The name of the exchange, which provides market data for the instrument.

maxLot number

The largest allowed trade quantity in lots.

minLot number

The smallest allowed trade quantity in lots.

name\* string

Name of the instrument.

noticeDate string($date)

The date, when the server will send Futures notice email about the fact that contract

is going to expire soon.

quotingCurrency\* string

Symbol currency, also named as counter currency. If a symbol is a currency pair, then the currency field has to contain the second currency of this pair.

sector string

The Sector to which the instrument belongs.

settlementDate string($date)

Settlement date of a contract.

settlementSystem string

The time between the trade date, when an order is executed in the market, and the settlement date.

Enum:

Array [ 9 ]

strikePrice number

Price at which a derivative contract can be bought or sold when it is exercised.

strikeType string

Put or Call

Enum:

Array [ 2 ]

symbolStatus\* string

Modes for trading.

Enum:

Array [ 3 ]

tickCost

[

Amount of base asset for one tick.

TickCost

{

leftRangeLimit number

Left end of range.

tickCost number

Amount of base asset for one tick.

}]

tickSize

[

Minimum price change for an instrument.

TickSize

{

leftRangeLimit number

Left end of range.

tickSize number

Minimum price change for an instrument.

}]

tradeSessionId integer($int32)

Identifier of the instrument trade session

tradeSessionStatusId integer($int64)

Identifier of the current status of the trade session for the instrument

tradingExchange\* string

The name of the exchange, on which the instrument trade will be performed.

type\* string

Symbol type.

Enum:

Array [ 10 ]

}

InstrumentDetailsResponse

{

d\* Instrument details

{

barSource string

The principle of building bars. Available values

Enum:

Array [ 3 ]

baseCurrency string

For currency pairs only. This field contains the first currency of the pair.

betSize number

The standard bet size for the instrument with spreadbet type in units.

betStep number

The minimal bet change of quantity in the betting currency. Required for spreadbet type symbols only.

bettingCurrency string

The currency in which the instrument will be traded. Required for spreadbet type symbols only.

contractMonth string($date)

Final day for delivery.

country integer($int32)

The country for the instrument.

deliveryStatus string

A contract delivery status.

Enum:

Array [ 3 ]

description string

Any useful information about the instrument.

exerciseStyle string

Defines the date on which the option may be exercised.

Enum:

Array [ 2 ]

firstTradeDate string($date)

The first day of trading on exchange

hasDaily boolean

Boolean value showing whether the symbol includes daily bars historical data.

hasIntraday boolean

Boolean value showing whether the symbol includes intraday (minutes) historical data.

industry string

The Industry to which the instrument belongs.

isin string

The International Securities Identification Number (ISIN) for the instrument.

lastTradeDate string($date)

The last day of trading on exchange. At the end of the session all orders will be

canceled and positions will be blocked (a trader won't be able to close them).

localizedName string

The name of the instrument on the specified language in request.

logoUrl string

URL to get instrument logo.

lotSize number

The standard lot size for the instrument in units.

lotStep number

The minimal lot quantity change.

marketCap number($double)

The market capitalization of the Instrument.

marketDataExchange\* string

The name of the exchange, which provides market data for the instrument.

maxLot number

The largest allowed trade quantity in lots.

minLot number

The smallest allowed trade quantity in lots.

name\* string

Name of the instrument.

noticeDate string($date)

The date, when the server will send Futures notice email about the fact that contract

is going to expire soon.

quotingCurrency\* string

Symbol currency, also named as counter currency. If a symbol is a currency pair, then the currency field has to contain the second currency of this pair.

sector string

The Sector to which the instrument belongs.

settlementDate string($date)

Settlement date of a contract.

settlementSystem string

The time between the trade date, when an order is executed in the market, and the settlement date.

Enum:

Array [ 9 ]

strikePrice number

Price at which a derivative contract can be bought or sold when it is exercised.

strikeType string

Put or Call

Enum:

Array [ 2 ]

symbolStatus\* string

Modes for trading.

Enum:

Array [ 3 ]

tickCost

[

Amount of base asset for one tick.

TickCost{...}

]

tickSize

[

Minimum price change for an instrument.

TickSize{...}

]

tradeSessionId integer($int32)

Identifier of the instrument trade session

tradeSessionStatusId integer($int64)

Identifier of the current status of the trade session for the instrument

tradingExchange\* string

The name of the exchange, on which the instrument trade will be performed.

type\* string

Symbol type.

Enum:

Array [ 10 ]

}

s\* string

example: ok

Status will always be ok.

}

Instruments

{

instruments

[Instrument

{

barSource string

The principle of building bars. Available values

Enum:

Array [ 3 ]

continuous boolean

Defines whether the contract is continuous.

contractMonth string($date)

Final day for delivery.

country integer($int32)

The country for the instrument.

description string

Any useful information about the instrument.

hasDaily boolean

Boolean value showing whether the symbol includes daily bars historical data.

hasIntraday boolean

Boolean value showing whether the symbol includes intraday (minutes) historical data.

id\* integer($int32)

Identifier of the instrument.

localizedName string

The name of the instrument on the specified language in request.

logoUrl string

URL to get instrument logo.

marketDataExchange\* string

The name of the exchange, which provides market data for the instrument.

name\* string

Name of the instrument.

strikePrice number

Price at which a derivative contract can be bought or sold when it is exercised.

strikeType string

Put or Call

Enum:

Array [ 2 ]

tradableInstrumentId\* integer($int64)

Identifier of the instrument that is used for trading purposes.

tradingExchange\* string

The name of the exchange, on which the instrument trade will be performed.

type\* string

Symbol type.

Enum:

Array [ 10 ]

underlierId integer($int32)

Identifier of the base asset

}]

}

InstrumentsResponse

{

d\* Instruments

{

instruments

[Instrument

{

barSource string

The principle of building bars. Available values

Enum:

Array [ 3 ]

continuous boolean

Defines whether the contract is continuous.

contractMonth string($date)

Final day for delivery.

country integer($int32)

The country for the instrument.

description string

Any useful information about the instrument.

hasDaily boolean

Boolean value showing whether the symbol includes daily bars historical data.

hasIntraday boolean

Boolean value showing whether the symbol includes intraday (minutes) historical data.

id\* integer($int32)

Identifier of the instrument.

localizedName string

The name of the instrument on the specified language in request.

logoUrl string

URL to get instrument logo.

marketDataExchange\* string

The name of the exchange, which provides market data for the instrument.

name\* string

Name of the instrument.

strikePrice number

Price at which a derivative contract can be bought or sold when it is exercised.

strikeType string

Put or Call

Enum:

Array [ 2 ]

tradableInstrumentId\* integer($int64)

Identifier of the instrument that is used for trading purposes.

tradingExchange\* string

The name of the exchange, on which the instrument trade will be performed.

type\* string

Symbol type.

Enum:

Array [ 10 ]

underlierId integer($int32)

Identifier of the base asset

}]

}

s\* string

example: ok

Status will always be ok.

}

User credentials

{

email\* string

example: john@email.com

User email

password\* string

example: secretPassword123

User password

server\* string

example: SERVER

Name of the broker server, also used during web login.

}

JWTRefreshToken

{

refreshToken\* string

example: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ0cmFkZWxvY2tlci1hcGkiLCJhdWQiOiJ0cmFkZWxvY2tlci1hcGktdHJhZGVycyIsInR5cGUiOiJyZWZyZXNoX3Rva2VuIiwic3ViIjoiT1NQI0RFVkNMTDlnNGlzVE9FaUtqZk5kSncxMyIsInVpZCI6IjE0NGJmMjg3LTUxZTQtNGI0Mi1iZTU0LWMwZGU0ZjExNzgzMyIsImJyYW5kIjoiT1NQIiwiaWF0IjoxNjg1MTE2NzkzLCJleHAiOjE2ODU3MjE1OTN9.GKNglolZzX76lKjTsrQ28MpmLTfU0A\_T7vCMrsojLcg

Refresh token for refreshing JWT tokens.

}

GetAllAccountsResponse

[

title: GetAllAccountsResponse

GetAllAccountsResponse

{

id integer

example: 7080

Account ID

name string

example: BRAND1#DEVCLL9g4isTOEiKjfNdJw13#1#1

Account Name

currency string

example: USD

Account currency

status integer

example: 0

Account status

accNum string

example: 1

Account number - used to select the account on /trade API

}]

JWTTokensResponse

{

accessToken\* string

example: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ0cmFkZWxvY2tlci1hcGkiLCJhdWQiOiJ0cmFkZWxvY2tlci1hcGktdHJhZGVycyIsInR5cGUiOiJhY2Nlc3NfdG9rZW4iLCJzdWIiOiJPU1AjREVWQ0xMOWc0aXNUT0VpS2pmTmRKdzEzIiwidWlkIjoiMTQ0YmYyODctNTFlNC00YjQyLWJlNTQtYzBkZTRmMTE3ODMzIiwiYnJhbmQiOiJPU1AiLCJpYXQiOjE2ODUxMTY3OTMsImV4cCI6MTY4NTEyMDM5M30.cyDXRqUNVX6h5rtZb7m30vNIwEoYN7xUU2jfGM-Cf90

Access token for accessing /trade APIs.

refreshToken\* string

example: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJ0cmFkZWxvY2tlci1hcGkiLCJhdWQiOiJ0cmFkZWxvY2tlci1hcGktdHJhZGVycyIsInR5cGUiOiJyZWZyZXNoX3Rva2VuIiwic3ViIjoiT1NQI0RFVkNMTDlnNGlzVE9FaUtqZk5kSncxMyIsInVpZCI6IjE0NGJmMjg3LTUxZTQtNGI0Mi1iZTU0LWMwZGU0ZjExNzgzMyIsImJyYW5kIjoiT1NQIiwiaWF0IjoxNjg1MTE2NzkzLCJleHAiOjE2ODU3MjE1OTN9.GKNglolZzX76lKjTsrQ28MpmLTfU0A\_T7vCMrsojLcg

Refresh token for refreshing JWT tokens.

}

LossLimitRule

{

value number

value

warnLevel1 number

WarnLevel1

warnLevel2 number

WarnLevel2

}

OrderHistory

{

ordersHistory

[

ordersHistory

[string]]

}

Orders

{

orders

[

orders

[string]]

}

OrdersHistory

{

description:

Orders history response

d\* OrderHistory

{

ordersHistory

[

ordersHistory

[string]]

}

s\* string

example: ok

Status will always be ok.

}

OrdersResponse

{

description:

Orders response

d\* Orders

{

orders

[

orders

[string]]

}

s\* string

example: ok

Status will always be ok.

}

PanelConfig

{

columns

[Column

{

description string

Column description.

id string

Column identifier.

}]

id string

Panel identifier.

title string

Localized title of a table

}

Limits

{

limitType string

Limit type

limit integer

Limit value

}

RateLimits

{

rateLimitType string

Limit type

measure string

Frequency measurement unit (SECONDS or MINUTES)

intervalNum integer

How many SECONDS/MINUTES are in the measured interval

limit integer

Limit value (maximum number of allowed times in intervalNum SECONDS/MINUTES)

}

Patch Order

{

price number

qty number

stopLoss number

stopLossType string

Enum:

Array [ 3 ]

stopPrice number

takeProfit number

takeProfitType string

Enum:

Array [ 2 ]

trStopOffset number

validity string

Enum:

Array [ 2 ]

}

Close Position

{

qty number

}

Patch Position

{

stopLoss number

takeProfit number

trailingOffset number

}

Permissions to use specific functions and information in the client terminal.

{

filledOrders boolean

Whether the Executions tab is supported. If set to false, the /executions endpoint should not be used.

marketDepth boolean

Whether the Market depth is supported. If set to false, the /depth endpoint should not be used.

orders boolean

Whether the Orders tab is supported. If set to false, the /orders endpoint should not be used.

ordersHistory boolean

Whether the Orders history tab is supported. If set to false, the /ordersHistory endpoint should not be used

positions boolean

Whether the Positions tab is supported. If set to false, the /positions endpoint should not be used.

symbolInfo boolean

Whether the Symbol info is supported. If set to false, the symbol\_info endpoint should not be used.

}

Positions

{

positions

[

[string]]

}

PositionsResponse

{

description:

Positions response

d\* Positions

{

positions

[

[string]]

}

s\* string

example: ok

Status will always be ok.

}

Post Order

{

price number

Limit Price for Limit order.

qty\* number

The number of units to open the buy or sell order.

routeId\* number

Identifier of the trade connection. Find the corresponding INFO or TRADE routeId by querying the /trade/accounts/{accountId}/instruments endpoint.

side\* string

Order side. If the creating order is a closing one for the position, then the side must

be opposite to the side of the position

Enum:

Array [ 2 ]

strategyId string

Arbitrary string (up to 31 chars) that can be attached to identify orders and positions placed through algorithmic trading.

This value will also be visible in GET /orders, GET /ordersHistory, and GET /positions.

stopLoss number

Stop loss amount for the order. Must be specified in case of absolute or offset stop loss type.

stopLossType string

Type of stop loss price for the order. Available types: absolute, offset, trailingOffset.

Enum:

Array [ 3 ]

stopPrice number

Stop Price for Stop orders.

takeProfit number

TakeProfit amount for the order. Must be specified together with the takeProfitType field. Specifies either the absolute price, or an offset in pips.

takeProfitType string

Type of take profit for the order. Available types: [absolute, offset]. Must be specified together with the takeProfit field.

Enum:

Array [ 2 ]

trStopOffset number

For the 'trailingOffset' stopLossType. The trailing offset in pips.

tradableInstrumentId\* integer($int64)

Identifier of the instrument that is used for trading purposes.

type\* string

Order type. Available types: limit, market, stop. If using a stop order, the stopPrice field must be specified. If using a limit order, the price field must be specified. If using a market order, the price field is ignored.

Enum:

Array [ 3 ]

validity\* string

Whether the order is Good Till Cancelled (GTC) or Immediate or Cancel (IOC). For market orders, use IOC, otherwise, use GTC.

In error messages, Validity is sometimes referred to as TimeInForce (TIF).

Enum:

Array [ 2 ]

}

PostOrderResponse

{

d\* PostOrderResponseD{...}

s\* [...]

}

PostOrderResponseD

{

orderId string

example: 1

New order identifier.

}

QuoteResponse

{

d\* SingleQuote

{

description:

Price response for an instrument.

ap number

Ask price.

as number

Best ask size.

bp number

Bid price.

bs number

Best bid size.

}

s\* string

example: ok

Status will always be ok.

}

RiskRules

{

balanceRelativeDrawdown number

The minimum allowable Projected balance value that the account can have. Max drawdown value calculates as % from the current Balance value

dailyLossLimit LossLimitRule

{

value number

value

warnLevel1 number

WarnLevel1

warnLevel2 number

WarnLevel2

}

dailyProfitTarget number

A maximum value of daily net profit allowed for an account

maxDrawdownLevel number

The minimal allowed Projected balance value in the account currency for the account not to trigger a Stop out

maxOrderAmount number

The maximum amount for a single order measured in the instrument units

maxOrderCapital number

The maximum order capital which can be sent by user. Order capital = Qty. \* Lot size \* Price \* Cross price

maxOrdersCount number

The maximum number of orders that user can place or modify during a day

maxPendingOrdersNumber number

The maximum quantity of pending orders which can be placed by account. Closing SL/TP orders aren’t taken into consideration in the counter, and are accepted without restrictions

maxPositionsNumber number

The maximum number of positions which can be opened by one account. Closing orders (Market and SL/TP) are accepted without checking and restrictions

maxTrailingDrawdown number

Maximum drawdown level for an account. A reached 'Max trailing drawdown' blocks trading

positionLossLimit number

Loss limit value for the position in percent

totalMaxPositionQty number

Maximum possible qty in lots of all positions and orders for all trading instruments. A reached 'Total max position qty' blocks an order placement and an order modification

unrealizedLossLimit LossLimitRule

{

value number

value

warnLevel1 number

WarnLevel1

warnLevel2 number

WarnLevel2

}

weeklyLossLimit LossLimitRule

{

value number

value

warnLevel1 number

WarnLevel1

warnLevel2 number

WarnLevel2

}

}

SessionPeriod

{

sub\_type

[SubType{...}

]

}

SessionTime

{

endTime string

Trading period end time

startTime string

Trading period start time.

}

SingleQuote

{

description:

Price response for an instrument.

ap number

Ask price.

as number

Best ask size.

bp number

Bid price.

bs number

Best bid size.

}

SubType

{

name string

sessionTime

[

Trading hours. Next sequence is used: sun, mon, tue, wed, thu, fri, sat, shortened day

SessionTime

{

endTime string

Trading period end time

startTime string

Trading period start time.

}]

}

SuccessResponse

{

s\* string

example: ok

Status will always be ok.

}

TickCost

{

leftRangeLimit number

Left end of range.

tickCost number

Amount of base asset for one tick.

}

TickSize

{

leftRangeLimit number

Left end of range.

tickSize number

Minimum price change for an instrument.

}

Trade session status

{

allowedOperations

[

List of allowed operations. The sequence used is: new, modification, cancel.

integer($int32)]

allowedOrderTypes

[

List of allowed order types. The used sequence is: market, stop, limit

integer($int32)]

}

TradeSession

{

blockTrading\* boolean

If true, trading will be fully blocked for all instruments for which a selected trade session is active.

holidays

[

The list of national holidays specified for the trading session.

Holiday

{

date string($date)

Holiday date.

name string

Holiday name.

session\_aftermarket SessionPeriod

{

sub\_type

[SubType

{

name string

sessionTime

[

Trading hours. Next sequence is used: sun, mon, tue, wed, thu, fri, sat, shortened day

SessionTime

{

endTime string

Trading period end time

startTime string

Trading period start time.

}]

}]

}

session\_beforemarket SessionPeriod

{

sub\_type

[SubType

{

name string

sessionTime

[

Trading hours. Next sequence is used: sun, mon, tue, wed, thu, fri, sat, shortened day

SessionTime

{

endTime string

Trading period end time

startTime string

Trading period start time.

}]

}]

}

session\_postclose SessionPeriod

{

sub\_type

[SubType

{

name string

sessionTime

[

Trading hours. Next sequence is used: sun, mon, tue, wed, thu, fri, sat, shortened day

SessionTime

{

endTime string

Trading period end time

startTime string

Trading period start time.

}]

}]

}

session\_preopen SessionPeriod

{

sub\_type

[SubType

{

name string

sessionTime

[

Trading hours. Next sequence is used: sun, mon, tue, wed, thu, fri, sat, shortened day

SessionTime

{

endTime string

Trading period end time

startTime string

Trading period start time.

}]

}]

}

session\_regular SessionPeriod

{

sub\_type

[SubType

{

name string

sessionTime

[

Trading hours. Next sequence is used: sun, mon, tue, wed, thu, fri, sat, shortened day

SessionTime

{

endTime string

Trading period end time

startTime string

Trading period start time.

}]

}]

}

workingState string

Working state of the holiday.

Enum:

Array [ 3 ]

}]

session\_aftermarket SessionPeriod

{

sub\_type

[SubType

{

name string

sessionTime

[

Trading hours. Next sequence is used: sun, mon, tue, wed, thu, fri, sat, shortened day

SessionTime

{

endTime string

Trading period end time

startTime string

Trading period start time.

}]

}]

}

session\_beforemarket SessionPeriod

{

sub\_type

[SubType

{

name string

sessionTime

[

Trading hours. Next sequence is used: sun, mon, tue, wed, thu, fri, sat, shortened day

SessionTime

{

endTime string

Trading period end time

startTime string

Trading period start time.

}]

}]

}

session\_postclose SessionPeriod

{

sub\_type

[SubType

{

name string

sessionTime

[

Trading hours. Next sequence is used: sun, mon, tue, wed, thu, fri, sat, shortened day

SessionTime

{

endTime string

Trading period end time

startTime string

Trading period start time.

}]

}]

}

session\_preopen SessionPeriod

{

sub\_type

[SubType

{

name string

sessionTime

[

Trading hours. Next sequence is used: sun, mon, tue, wed, thu, fri, sat, shortened day

SessionTime

{

endTime string

Trading period end time

startTime string

Trading period start time.

}]

}]

}

session\_regular SessionPeriod

{

sub\_type

[SubType

{

name string

sessionTime

[

Trading hours. Next sequence is used: sun, mon, tue, wed, thu, fri, sat, shortened day

SessionTime

{

endTime string

Trading period end time

startTime string

Trading period start time.

}]

}]

}

timeZone\* string

Timezone of the exchange for this symbol. List of the available time zone ids is used from Java TimeZone class.

}

TradeSessionResponse

{

d\* TradeSession

{

blockTrading\* boolean

If true, trading will be fully blocked for all instruments for which a selected trade session is active.

holidays

[

The list of national holidays specified for the trading session.

Holiday

{

date string($date)

Holiday date.

name string

Holiday name.

session\_aftermarket SessionPeriod

{

sub\_type

[SubType

{

name string

sessionTime

[

Trading hours. Next sequence is used: sun, mon, tue, wed, thu, fri, sat, shortened day

SessionTime

{

endTime string

Trading period end time

startTime string

Trading period start time.

}]

}]

}

session\_beforemarket SessionPeriod

{

sub\_type

[SubType

{

name string

sessionTime

[

Trading hours. Next sequence is used: sun, mon, tue, wed, thu, fri, sat, shortened day

SessionTime

{

endTime string

Trading period end time

startTime string

Trading period start time.

}]

}]

}

session\_postclose SessionPeriod

{

sub\_type

[SubType

{

name string

sessionTime

[

Trading hours. Next sequence is used: sun, mon, tue, wed, thu, fri, sat, shortened day

SessionTime

{

endTime string

Trading period end time

startTime string

Trading period start time.

}]

}]

}

session\_preopen SessionPeriod

{

sub\_type

[SubType

{

name string

sessionTime

[

Trading hours. Next sequence is used: sun, mon, tue, wed, thu, fri, sat, shortened day

SessionTime

{

endTime string

Trading period end time

startTime string

Trading period start time.

}]

}]

}

session\_regular SessionPeriod

{

sub\_type

[SubType

{

name string

sessionTime

[

Trading hours. Next sequence is used: sun, mon, tue, wed, thu, fri, sat, shortened day

SessionTime

{

endTime string

Trading period end time

startTime string

Trading period start time.

}]

}]

}

workingState string

Working state of the holiday.

Enum:

Array [ 3 ]

}]

session\_aftermarket SessionPeriod

{

sub\_type

[SubType

{

name string

sessionTime

[

Trading hours. Next sequence is used: sun, mon, tue, wed, thu, fri, sat, shortened day

SessionTime

{

endTime string

Trading period end time

startTime string

Trading period start time.

}]

}]

}

session\_beforemarket SessionPeriod

{

sub\_type

[SubType

{

name string

sessionTime

[

Trading hours. Next sequence is used: sun, mon, tue, wed, thu, fri, sat, shortened day

SessionTime

{

endTime string

Trading period end time

startTime string

Trading period start time.

}]

}]

}

session\_postclose SessionPeriod

{

sub\_type

[SubType

{

name string

sessionTime

[

Trading hours. Next sequence is used: sun, mon, tue, wed, thu, fri, sat, shortened day

SessionTime

{

endTime string

Trading period end time

startTime string

Trading period start time.

}]

}]

}

session\_preopen SessionPeriod

{

sub\_type

[SubType

{

name string

sessionTime

[

Trading hours. Next sequence is used: sun, mon, tue, wed, thu, fri, sat, shortened day

SessionTime

{

endTime string

Trading period end time

startTime string

Trading period start time.

}]

}]

}

session\_regular SessionPeriod

{

sub\_type

[SubType

{

name string

sessionTime

[

Trading hours. Next sequence is used: sun, mon, tue, wed, thu, fri, sat, shortened day

SessionTime

{

endTime string

Trading period end time

startTime string

Trading period start time.

}]

}]

}

timeZone\* string

Timezone of the exchange for this symbol. List of the available time zone ids is used from Java TimeZone class.

}

s\* string

example: ok

Status will always be ok.

}

TradeSessionStatusResponse

{

d\* Trade session status

{

allowedOperations

[

List of allowed operations. The sequence used is: new, modification, cancel.

integer($int32)]

allowedOrderTypes

[

List of allowed order types. The used sequence is: market, stop, limit

integer($int32)]

}

s\* string

example: ok

Status will always be ok.

}

TradingRules

{

supportBrackets boolean

Whether the brackets (take profit and stop loss) for orders and positions are supported

supportPartialClosePosition boolean

Supporting partial closing of a position.

supportSelfTrading boolean

Whether the user is allowed to trade on this account

supportTradingOutOfTradingHours boolean

Whether placing orders out of trading hours is allowed for the account

supportTrailingStop boolean

Whether the trailing stop orders are supported

}

Unlock your potential.