



ErisX REST API v3.0



Please contact ErisX sales representatives or help desk personnel for more information on this documentation.

Contents

Change History	3
General Concepts	5
API Credentials	5
API Key permissions	5
Authentication	6
Funding Password Signing	7
Filters	8
Sorting	9
Pagination	9
Trade Date and Business Date	10
Rate Limiting	10
Clearing API Service	11
REST API Endpoint URL	11
Funds Designation	11
Balances Calculations	11
Account Method balance values	11
Balances Method, Opening Balance calculation	11
Accounts Method	12
Balances Method	14
Movements Method	16
Trades Method	19
Requests Method	22
Positions Method	24
Deposit Address Method	27
Linked Account Methods	28
Withdrawal Request Methods	30
Build Withdrawal Request	30
Submit Withdrawal Request	31
Block Trade API Service (NEW)	33
REST API Endpoint URL	33
Block Trade States	33
Error Codes	33
Duplicate Requests	34
Submit Block Trade Method	34
Request Block Trade Information Method	36
Block Trade Volume Method	38

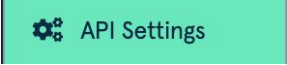
1 Change History

Date	Message(s) or Section	Description
20190731		Version 1
20190809	Filters	In python, filters should be specified in the json argument of requests.post function, not in the data argument. Some new filters have been added to the different methods. Each token will now be valid for 60 seconds, instead of the previous 30 seconds
20190819	Trades Response	The trades response will now include 3 new fields (tcr_id, client_order_id, fix_id)
20190925	Trades Response	The trades response will now include one new field: product_code
20200130		Version 1.5
	Messages updated; Account , Balances and Trades .	New fields introduced to expose Futures information to clients (highlighted in green).
	New message added; Positions	New positions endpoint to query the positions for a given account.
20200301		Version 2.0
	Movements & Trades Filter changes	Removed asset_type filter from Movements and fee_type filter from Trades.
	New methods: deposit_address , linked_accounts , build_withdrawal_request and submit_withdrawal_request	A set of new methods to allow users to process deposits and withdrawals.
	Authentication	Removed Python 2 authentication example
20200326		Version 3.0
	New Service: Block Trades API Service	Added new methods to interact with the Block Trade API
	New section: Balances Calculations	New section that details balances calculations
	Funding Password Signing	Modified the javascript example for easier use

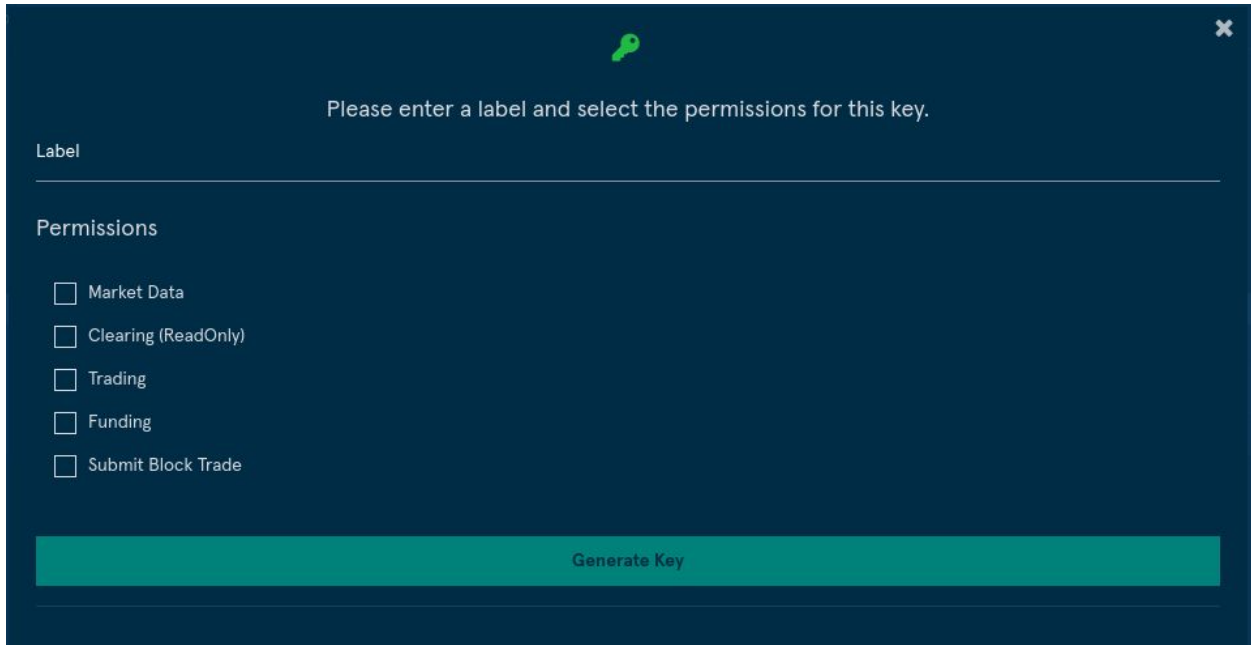
2 General Concepts

2.1 API Credentials

In order to sign your API requests, you will need to create a set of API Credentials.

From the Eris Member Portal, navigate to the dropdown next to your username in the top right of the page and select .

After clicking **Create New API Key** you will be asked to select the permissions you want to enable.



Please enter a label and select the permissions for this key.

Label

Permissions

- Market Data
- Clearing (ReadOnly)
- Trading
- Funding
- Submit Block Trade

Generate Key

API Key permissions

- **Market Data:** An API key can query historical data or subscribe to real time data.
- **Trading:** Allows an API key to enter, modify and cancel orders.
- **Clearing (ReadOnly):** Allows an API key to query information about their clearing accounts.
- **Funding:** Allows an API key to initiate withdrawal requests.
- **Submit Block Trade:** Allows an API key to submit Block Trades.

When ready click **Generate Key** and you will be presented with two pieces of information that must be kept safe as they will be needed for authentication of calls to the end points and will not be shown again:

- **API key**
- **Secret**



2.2 Authentication

A json web token should be generated using the HS256 algorithm on the API key, secret and timestamp as described in the examples below. This token should be included in the header of every request.

- **Timestamp:** The authentication token requires a Unix Epoch timestamp.
- **Token Age:** Each token will only be valid for 60 seconds after the timestamp.

Notes:

- In python use the **pyjwt** package to generate the token (<https://pyjwt.readthedocs.io/en/latest/>).
- Note that some jwt encoding functions may return a byte array rather than a string, and some languages require explicit conversion. For example, in Python, you must use the **decode()** function.
- Be aware that there must be a blank space between **Bearer** and the token.

Javascript Example:

```
const jwt = require('jsonwebtoken');
const axios = require('axios').default;
const apiKey = '9106676d85f1163f.d1ba2efac8bc1e0a';
const secret = '31b6b61606588580';
var payload = {
  iat: Date.now(),
  sub: apiKey
};
var token = jwt.sign(payload, secret, { algorithm: 'HS256' });
axios({
  method: 'POST',
  url: `https://clearing.erisx.com/api/v1/${method_name}`,
  timeout: 30000, // 30 seconds
  headers: {
    Authorization: `Bearer ${token}`,
  },
  data: {}
});
```

Python 3 Example:

```
import jwt
import time
import requests

def gen_token(secret, api_key):
    unix_timestamp = int(round(time.time()))
    payload_dict = {'sub': api_key, 'iat': unix_timestamp}
```

```
return jwt.encode(payload_dict, secret, algorithm='HS256').decode('utf-8')

my_secret = '31b6b61606588580'
my_api_key = '9106676d85f1163fgd1ba2efac8bc1e0a'
url = 'https://clearing.erisx.com/api/v1/'
token = gen_token(my_secret, my_api_key)
requests.post(url= url + method_name, headers={'Authorization': 'Bearer ' +
token}, json={}) # Be aware that there is a blank space after Bearer
```

2.3 Funding Password Signing

In order to enhance security in funding related operations, some methods of the Clearing API require a two step process authentication. The first authentication is based on the API credentials and the token derivation described in the section above. The second authentication is based on the generation of an encrypted signature based on the member user's Funding Password. In order to correctly perform the signature, the clearing member should follow the following steps:

- Generate a key from the funding password using the Password-Based Key Derivation Function 2 (PBKDF2) with the following parameters:
 - Hashing algorithm: SHA-256
 - Password: Clearing member user's Funding Password.
 - Salt: Auth ID for the clearing member user, which can be found in the response from the method [Build Withdrawal Request](#) in the field 'auth_id'.
 - Iterations: 100,000
 - Derive Key Length: 32 bytes (256 bits).
- Generate a canonical signature for the appropriate message using the Elliptic Curve Digital Signature Algorithm (ECDSA) with the elliptic curve SECP256k1, where the key used in the signature should be the key generated in the previous step. Encoding of the signature should be in DER format.
- Encode the signature using Base58 encoding. Note that some base58 encoding functions may return a byte array rather than a string, and some languages require explicit conversion. For example, in Python, you must use the decode() function.

This signature will enable the ErisX ClearingHouse to validate the funding password without the clearing member having to expose the funding password at any moment over the internet, which provides a higher layer of security for the safe keeping of the clearing member's credentials.

Javascript Example:

```
const crypto = require('crypto');
const bs58 = require('bs58');
const ecdsa = require('ecdsa');
const pbkdf2 = require('pbkdf2-sha256');
```

```
const BigInteger = require('bigi');

function sha256(str) {
  return crypto.createHash('sha256').update(str).digest();
}

function privateKeyFromPassword(authId, password) {
  return pbkdf2(password, authId, 100000, 32);
}

function signMessage(message, privateKey) {
  let shaMsg = sha256(message);
  let signature = ecdsa.sign(shaMsg, BigInteger.fromBuffer(privateKey));
  return signature.toDER();
}

const privateKey = privateKeyFromPassword(auth_id, password);
const signature = bs58.encode(signMessage(request_data, privateKey));
```

Python3 Example:

```
import hashlib
import ecdsa
from ecdsa.util import sigencode_der_canonize
import base58

def privateKeyFromPassword(authId, password):
    return hashlib.pbkdf2_hmac(
        hash_name='sha256',
        password=password.encode(),
        salt=authId.encode(),
        iterations=100000,
        dklen=32)

def signMessage(message, authId, password):
    privateKey = privateKeyFromPassword(authId, password)
    sk = ecdsa.SigningKey.from_string(privateKey, curve=ecdsa.SECP256k1)
    signature = sk.sign_deterministic(
        message.encode(),
        sigencode=sigencode_der_canonize,
        hashfunc=hashlib.sha256)
    return base58.b58encode(signature).decode('ascii')

signature = signMessage(message, authId, password)
```



2.4 Filters

Some methods allow the use of filters. These filters provide a greater level of flexibility to queries. Ultimately, providing more efficient requests and a better API experience.

The filter query has the following json type format. Multiple filters can be applied in a single request to best tailor the query. In python, filters should be given under the **json** argument of the requests.post function.

```
"filter": [{ "attr": "attribute_name", "op": "eq", "value": "attribute_value" }]
```

Field	Value
filter	Name of the query parameter
attr	Name of the attribute that wants to be used in the query
op	Operations present in the query: 'eq' - equal 'ne' - not equal 'gt' - greater than 'gte' - greater than or equal 'lt' - less than 'lte' - less than or equal
value	Value or array of values of the attribute to which the query will compare.

2.5 Sorting

Queries also provide the ability to sort the results using the following format.

```
"sort": [{ "attr": "attribute_name", "value": "desc" }]
```

Field	Value
sort	Name of the query parameter
attr	Name of the attribute that wants to be used in the query
value	Direction of the sort: 'desc' - descending or 'asc' - ascending

2.6 Pagination

Some requests can be paginated. The offset and limit parameters on the request allows the user to choose how many results should be included in the return message and where the results should begin.

Maximum number of results per request is 100.

These two parameters are optional and available parameters in all methods except in the Balances method.

```
"offset":0, "limit":10
```

Field	Value
-------	-------

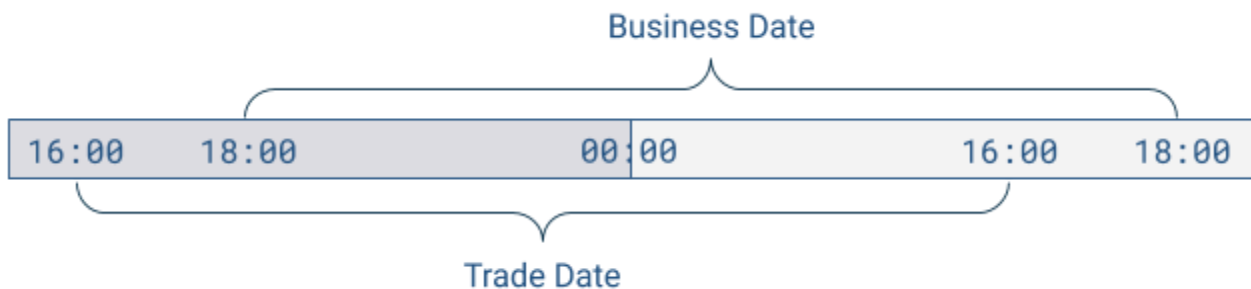


offset	Integer. The number of entries to skip (default: 0).
limit	Integer. Maximum number of results to be returned (default: 100).

2.7 Trade Date and Business Date

A new trade date starts at 4:00:00pm CST and finishes at 3:59:59pm CST the following day. All Exchange (trading) activity will be included in the appropriate trade date depending on the time of the activity. (I.e. trading activity at 2019-01-01 15:59:59 CST will be included in 2019-01-01 trade date but trading activity at 2019-01-01 16:00:00 CST will be included in 2019-01-02 trade date).

A new business date starts at 6:00pm CST and finishes at 5:59pm CST of the following day. All asset movement activity (Deposits, Withdrawals) will be included in the appropriate business date depending on the time of the asset movement. (I.e. a deposit made at 2019-01-01 17:59:59 CST will be included in 2019-01-01 business date but a deposit made at 2019-01-01 18:00:00 CST will be included in the 2019-01-02 business date).



The relevance of these two time frames is important for understanding the calculation of the [Opening Balance](#) in the Balances method.

2.8 Rate Limiting

Requests are throttled per IP address. Limit: 5 requests in a 10 second period.

When the rate limit is exceeded, a response with status **429 -> Too Many Requests** is returned.

If the limit is exceeded the IP address will be restricted from making new requests for 60 seconds.

We highly recommend adding logic to your application to gracefully process the 429 Too Many Requests message. We suggest that if the limit is breached, your application will pause for the required time in order to be within the rate limiting again. An application that repeatedly breaches the limit will keep extending the restricted period, thus, preventing your application to function correctly again.



3 Clearing API Service

This API service enables clients to interact with their Clearing accounts in order to extract data regarding their activity. All requests and responses are application/json content type.

All Clearing API methods are private and every request needs to be signed using the authentication method described in the [Authentication](#) section. Some methods require additional signing authentication as described in the [Funding Password Signing](#) section.

3.1 REST API Endpoint URL

- **Production:** <https://clearing.erisx.com/api/v1>
- **New Release (test):** <https://clearing.newrelease.erisx.com/api/v1>

3.2 Funds Designation

All customer funds for trading on designated contract markets (futures exchanges like ErisX) must be kept apart (“segregated”) from non customer funds.

ErisX currently supports three funds designations:

- N: Represents “non-segregated” funds held on behalf of members trading ErisX Spot products.
- P: Represents “member property” funds held on behalf of direct members trading ErisX futures products.
- S: Represents “segregated” funds held on behalf of the clients of Futures Commission Merchants (FCM’s) trading ErisX futures products.

3.3 Balances Calculations

As defined in the [Trade Date and Business Date](#) section, Trade Date has a different start time and end time than Business Date. This difference has certain implications in how balances are calculated and provided in response to the different Clearing API methods.

3.3.1 Account Method balance values

The response of the accounts method, provides a summary of balances for the account along with other account information. The balances reflected include all Exchange and Clearing House activities up to the moment when the API request is made. Therefore, this value will include the latest information known for the account.

3.3.2 Balances Method, Opening Balance calculation

The balances method, provides a more detailed view of the balances for a particular account. The opening balance is generated using a reference to the current business date.

Therefore, the following rules need to be considered to understand the value provided;

- Include all asset movements (Deposits, Withdrawals, etc.) prior to the beginning of the current business date.
- Include all trading activity for the trade date prior to the current business date.

Examples

Request is within the same trade date and business date:



The Opening balance for a request on Tuesday @ **15:50** CT will include:

- Trades prior to Monday @ 16:00 CT
- Asset movements prior to Monday @ 18:00 CT

Request is during a new trade date but still the same business date

The Opening balance for a request on Tuesday @ **16:30** CT will include:

- Trades up to Monday @ 16:00 CT
- Asset movements prior to Monday @ 18:00 CT

Request is during a new trade date and new business date (same calendar date)

The Opening balance for a request on a Tuesday @ **18:30** CT will include:

- Trades up to Tuesday @ 16:00 CT
- Asset movements prior to Tuesday @ 18:00 CT

3.4 Accounts Method

This method will return a list of all accounts a member has available to them, as well as basic balance information. More detailed balance information is returned in the getBalances method.

- **HTTP Request Type:** POST
- **Method:** /accounts
- **API security:** This API method requires an authentication token with Clearing API read permission.

Inputs:

Field	Value	
filter (optional)	Default: <code>"filter": [{ "attr": "account_id", "op": "eq", "value": member_account_id }]</code>	
	<table border="1"> <tr> <td>account_id</td> <td>Account ID</td> </tr> </table>	account_id
account_id	Account ID	
offset (optional)	Number of elements to be offset in the request for pagination purposes	
limit (optional)	Limit of elements returned in the request	

Example Requests:

```
requests.post(
  url="https://clearing.erisx.com/api/v1/accounts",
  headers={"Authorization": "Bearer " + token},
  json={})

requests.post(
  url="https://clearing.erisx.com/api/v1/accounts",
  headers={"Authorization": "Bearer " + token},
```

```

json={
  "filter": [{
    "attr":
      "account_id",
    "op":
      "eq",
    "value": [
      "27ff6d34-523d-476d-9ad5-edeb373b83dc"
    ]
  }],
  "offset": 0,
  "limit": 10
})

```

Outputs

Field	Value
count	Number of member accounts found
timestamp	Time of the request
accounts	List of all available accounts
account_id	Account ID
account_number	Account Number
fix_ids	List of all available FIX Trading IDs
member_users	Member users associated with the account
balances	Balances of the account at the time of the request
cti	Customer Type Indicator (For futures accounts)
origin	Origin (For futures accounts)

Example Response:

```

{
  "count": 1,
  "timestamp": "2018-01-01T06:00:00.000Z",
  "accounts": [
    {
      "account_id": "27ff6d34-523d-476d-9ad5-edeb373b83dc",
      "account_number": "DM-000001",
      "member_users": [
        "5c532a02f2530e906a9c065f"
      ],
      "balances": [
        {
          "asset_type": "USD",
          "amount": "100.5"
        }
      ],
    }
  ],
}

```

```

    {
      "asset_type": "TBTC",
      "amount": "1.5"
    }
  ],
  "fix_ids": [
    "trading_id"
  ],
  "cti": 1,
  "origin": 2
}
]
}

```

3.5 Balances Method

This method will return a detailed set of balance information for a given account.

- **HTTP Request Type:** POST
- **Method:** /balances
- **API security:** This API method requires an authentication token with Clearing API read permission.

Inputs

Field	Value
account_id	Account ID

Example Request:

```

requests.post(
  url="https://clearing.erisx.com/api/v1/balances",
  headers={"Authorization": "Bearer " + token},
  json={"account_id": "27ff6d34-523d-476d-9ad5-edeb373b83dc"})

```

Outputs

Field	Value
account_id	Account ID
timestamp	Time of the request
report_date	Business date associated with the request
asset_type	Asset Type
opening_balance	Balance at the beginning of the corresponding business date
asset_movement	Amount of asset movements for the business date up to the time of the request.
spot_movement	Amount of Spot trade movements for the business date up to the time of the request.
closing_balance	Balance as of the time of the request.



change_in_balance	Change in balance between the beginning of the request's business date and the time of the request.
exchange_fees	Exchange fees paid during the request's trade date
clearing_fees	Clearing fees paid during the request's trade date
other_fees	Other fees paid during the request's business date
realized_p_and_l	Realized Profit and Loss in Futures trades
futures_delivery	Quantity of Futures contract delivered
total_equity	Total Equity
reserved_margin	Reserved Margin for Futures positions
total_excess_deficit	Total Excess Deficit
net_liquidating_value	Net Liquidating Value
available_to_trade	Balance available to trade (does not include working orders)
reserved_ote	Reserved OTE
fd	Funds designation
closing_price	Closing price of each asset at the end of the previous trade date
closing_price_date	Trade date to which the closing price belongs
usd_value	The USD equivalent balance for each asset based on the closing price of the previous trade date from the time of the request.

Example Response:

```
{
  "account_id": "3cfb773e-3a71-42c8-bab5-037ca4ae616f",
  "timestamp": "2020-01-30T16:52:41.900Z",
  "report_date": "2020-01-30",
  "balances": [
    {
      "opening_balance": "100621.4286",
      "spot_movement": "0.0",
      "exchange_fees": "-1.8",
      "clearing_fees": "-0.2",
      "other_fees": "0.0",
      "asset_movement": "0.0",
      "realized_p_and_l": "81.8",
      "futures_delivery": "0.0",
      "closing_balance": "100701.2286",
      "total_equity": "100701.2286",
      "reserved_margin": "-21294.6",
      "total_excess_deficit": "79406.6286",
      "net_liquidating_value": "100701.2286",
      "available_to_trade": "79406.6286",
      "reserved_ote": "1375.4",
      "fd": "P",
      "asset_type": "USD",
    }
  ]
}
```



```

    "closing_price": "1.0",
    "closing_price_date": "2020-01-28",
    "usd_value": "100701.2286",
    "change_in_balance": "79.8"
  },
  {
    "opening_balance": "0.79925",
    "spot_movement": "0.0",
    "exchange_fees": "0.0",
    "clearing_fees": "0.0",
    "other_fees": "0.0",
    "asset_movement": "0.0",
    "realized_p_and_l": "0.0",
    "futures_delivery": "0.0",
    "closing_balance": "0.79925",
    "total_equity": "0.79925",
    "reserved_margin": "0.0",
    "total_excess_deficit": "0.79925",
    "net_liquidating_value": "0.79925",
    "available_to_trade": "0.79925",
    "reserved_ote": "0.0",
    "fd": "P",
    "asset_type": "TBTC",
    "closing_price": "9068.0",
    "closing_price_date": "2020-01-28",
    "usd_value": "7247.599",
    "change_in_balance": "0.0"
  }
]
}

```

3.6 Movements Method

This method will return a detailed set of asset movements information for a given account.

- **HTTP Request Type:** POST
- **Method:** /movements
- **API security:** This API method requires an authentication token with Clearing API read permission.

Inputs

Field	Value
filters (optional)	Default: <code>"filter": [{ "attr": "account_id", "op": "eq", "value": member_account_id }]</code>
	account_id Account ID
	time Start time using "op":gte or gt and End time using "op":lte or lt. If no time query is made it will return all the available

	data (subject to the specified limit)
offset (optional)	Number of elements to be offset in the request for pagination purposes
limit (optional)	Limit of elements returned in the request
Sort (optional)	Default: "sort": [{ "attr": "time", "value": "desc" }]

Example Request:

```
requests.post(
    url="https://clearing.erisx.com/api/v1/movements",
    headers={"Authorization": "Bearer " + token},
    json={
        "filter": [{
            "attr": "account_id",
            "op": "eq",
            "value": "27ff6d34-523d-476d-9ad5-edeb373b83dc"
        }], {
            "attr": "time",
            "op": "lte",
            "value": "2018-01-01T05:59:30.000Z"
        }, {
            "attr": "time",
            "op": "gte",
            "value": "2017-12-01T05:59:30.000Z"
        }],
        "sort": [{
            "attr": "time",
            "value": "asc"
        }],
        "offset": 0,
        "limit": 10
    })
```

Outputs

Field	Value
count	Number of results returned
description	Description of the asset movement
time	Timestamp of the asset movement
date	Business date of the asset movement
type	Type of the asset movement
posting_summary	Details of the asset movement (account ID, Asset type, Key (specifies what the amount refers to), Amount and Report Date). List of available keys: <ul style="list-style-type: none"> • "amount": General movement amount. • "bank_fee": Bank Fees Charged • "clearing_fee": Clearing House Fees Charged

	<ul style="list-style-type: none">• "exchange_fee": Trading Fees Charged• "other_fees": Other Fees Charged
--	---

Example Response:

```
{
  "result": {
    "count": 1,
    "movements": [
      {
        "description": "DEPOSIT 0.13057719 TBTC",
        "time": "2018-01-01T06:00:00.000Z",
        "type": "deposit",
        "posting_summary": [
          {
            "account_id": "27ff6d34-523d-476d-9ad5-edeb373b83dc",
            "asset_type": "TBTC",
            "key": "notional",
            "amount": "0.25486",
            "report_date": "2018-01-01"
          },
          {
            "account_id": "27ff6d34-523d-476d-9ad5-edeb373b83dc",
            "asset_type": "TBTC",
            "key": "clearing_fee",
            "amount": "0.00002549",
            "report_date": "2018-01-01"
          },
          {
            "account_id": "27ff6d34-523d-476d-9ad5-edeb373b83dc",
            "asset_type": "TBTC",
            "key": "exchange_fee",
            "amount": "0.00022937",
            "report_date": "2018-01-01"
          }
        ]
      }
    ]
  }
}
```

3.7 Trades Method

This method will return a set of trade information for a given account.

- **HTTP Request Type:** POST
- **Method:** /trades

- **API security:** This API method requires an authentication token with Clearing API read permission.

Inputs

Field	Value
filters (optional)	Default: <code>"filter": [{ "attr": "account_id", "op": "eq", "value": member_account_id }]</code>
account_id	Account ID
time	Start time using "op":gte or gt and End time using "op":lte or lt. If no time query is made it will return all the available data (subject to the specified limit)
trade_id	Trade ID
side	Side of the trade (BUY, SELL)
aggressor	Aggressor in the trade (Y, N)
qty	Quantity of the trade
px	Price of the trade
qty_type	Base currency
px_type	Quoted currency
type	Types: futures, spot, delivery or reversal
offset (optional)	Number of elements to be offset in the request for pagination purposes
limit (optional)	Limit of elements returned in the request
Sort (optional)	Default: <code>"sort": [{ "attr": "time", "value": "desc"}]</code>

Example Request:

```
requests.post(
  url="https://clearing.erisx.com/api/v1/trades",
  headers={"Authorization": "Bearer " + token},
  json={
    "filter": [{
      "attr": "account_id",
      "op": "eq",
      "value": "27ff6d34-523d-476d-9ad5-edeb373b83dc"
    }]
  }
)
```

Outputs

Field	Value
count	Number of results returned
trade_id	Trade ID of the trade
tcr_id	Trade Capture Report ID



client_order_id	Client Order ID
fix_id	FIX ID
time	Timestamp of the trade
description	Description of the trade
side	Side of the trade (BUY, SELL)
account_id	Account ID
aggressor	Aggressor of the trade (Y, N)
qty	Quantity
px	Price
clearing_fee	Clearing fee of the trade
exchange_fee	Exchange fee of the trade
product_code	Product code
qty_type	Base currency
px_type	Quote currency
fee_type	Fee currency
report_date	Business date of the trade
contract_symbol	Contract Symbol
asset_type	Asset Type
trader_type	Trade Type
record_type	Record Type
notional	Notional Amount
total_amount	Total Amount charged to the Account
trade_report_id	Trade Report ID
customer_account_ref	Customer Account Reference
product_suffix	Product Type: SP, FUT
state	State of the Trade
expiration_time	Expiration date and Time of the futures contract involved in the trade
cti	CTI
origin	Origin

Example Response:

```
{
  "count": 1,
  "trades": [
    {
      "account_id": "27ff6d34-523d-476d-9ad5-edeb373b83dc",

```

```
"contract_symbol": "TBTCZ9",
"asset_type": "TBTC",
"px_type": "USD",
"side": "BUY",
"trade_type": "REGULAR",
"record_type": "T",
"qty": "1.0",
"px": "6994.0",
"notional": "699.4",
"aggressor": "Y",
"fee_type": "USD",
"exchange_fees": "0.001",
"clearing_fees": "0.001",
"total_amount": "699.402",
"tcr_id": "477188150",
"trade_report_id": "1125899907429878",
"trade_id": "B2019196081HP00",
"customer_account_ref": "buy_side",
"fix_id": "1",
"product_suffix": "FUT",
"state": "posted",
"time": "2018-01-01T06:00:00.000000Z",
"expiration_time": "2030-01-01T06:00:00Z",
"cti": 1,
"origin": 1,
"product_code": "TBTC/USD",
"client_order_id": "1",
"description": "BUY 1.0 TBTCZ9 @ 6994.0 USD"
}
]
}
```

3.8 Requests Method

This method will return the asset movements requests made by the appropriate account and their current status.

- **HTTP Request Type:** POST
- **Method:** /requests
- **API security:** This API method requires an authentication token with Clearing API read permission.

Inputs

Field	Value										
filters (optional)	Default: "filter": [{ "attr": "account_id", "op": "eq", "value": member_account_id }]										
	<table border="1"> <tr> <td>account_id</td> <td>Account ID</td> </tr> <tr> <td>time</td> <td>Start time using "op":gte or gt and End time using "op":lte or lt. If no time query is made it will return all the available data (subject to the specified limit)</td> </tr> <tr> <td>asset_type</td> <td>Asset type (BTC, BCH, ETH, LTC)</td> </tr> <tr> <td>amount</td> <td>Amount of the request</td> </tr> <tr> <td>transaction_type</td> <td>Request type (withdrawal, deposit)</td> </tr> </table>	account_id	Account ID	time	Start time using "op":gte or gt and End time using "op":lte or lt. If no time query is made it will return all the available data (subject to the specified limit)	asset_type	Asset type (BTC, BCH, ETH, LTC)	amount	Amount of the request	transaction_type	Request type (withdrawal, deposit)
account_id	Account ID										
time	Start time using "op":gte or gt and End time using "op":lte or lt. If no time query is made it will return all the available data (subject to the specified limit)										
asset_type	Asset type (BTC, BCH, ETH, LTC)										
amount	Amount of the request										
transaction_type	Request type (withdrawal, deposit)										
offset (optional)	Number of elements to be offset in the request for pagination purposes										
limit (optional)	Limit of elements returned in the request										
Sort (optional)	Default: "sort": [{ "attr": "time", "value": "desc" }]										

Example Request :

```
requests.post(
  url="https://clearing.erisx.com/api/v1/requests",
  headers={"Authorization": "Bearer " + token},
  json={
    "filter": [{
      "attr": "account_id",
      "op": "eq",
      "value": "27ff6d34-523d-476d-9ad5-edeb373b83dc"
    }],
    "sort": [{
      "attr": "time",
      "value": "asc"
    }],
    "offset": 0,
    "limit": 10
  })
```

Outputs

Field	Value
count	Number of results returned
account_id	Account ID
dest_address	Destination Address of the request (Digital Assets Only)
time	Timestamp of the asset movement request
asset_type	Asset type
amount	Amount of the request
fee	Fees
fee_type	Fee currency
transaction_type	Transaction type (deposit, withdrawal)
state	State of the request (pending, rejected, posting)

Example Response:

```
{
  "result": {
    "count": 1,
    "requests": [
      {
        "account_id": "27ff6d34-523d-476d-9ad5-edeb373b83dc",
        "dest_address": "2MvQsanz92K5DmKe9fd2GHPz4oRKJXoAR1m4",
        "time": "2018-01-01T06:00:10.000Z",
        "asset_type": "TBTC",
        "amount": "-0.0001",
        "fee": "-0.0000001",
        "fee_type": "TBTC",
        "transaction_type": "withdrawal",
        "state": "pending"
      }
    ]
  }
}
```

3.9 Positions Method

This method will return the list of open positions for each account.

- **HTTP Request Type:** POST
- **Method:** /positions
- **API security:** This API method requires an authentication token with Clearing API read permission.

Inputs

Field	Value
-------	-------



filters (optional)	Default: "filter": [{ "attr": "account_id", "op": "eq", "value": member_account_id }]	
	account_id	Account ID
	contract_symbols	Contract symbol
offset (optional)	Number of elements to be offset in the request for pagination purposes	
limit (optional)	Limit of elements returned in the request	
Sort (optional)	Default: "sort": [{ "attr": "time", "value": "desc"}]	

Example Request:

```
requests.post(
    url="https://clearing.erisx.com/api/v1/positions",
    headers={"Authorization": "Bearer " + token},
    json={
        "filter": [
            {
                "attr": "account_id",
                "op": "eq",
                "value": "27ff6d34-523d-476d-9ad5-edeb373b83dc"
            },
            {
                "attr": "contract_symbols",
                "op": "eq",
                "value": "BTCW44"
            }
        ],
        "sort": [{
            "attr": "time",
            "value": "asc"
        }],
        "offset": 0,
        "limit": 10
    }
)
```

Outputs

Field	Value
account_id	Account ID
positions	List of Open positions
contract_symbol	Exchange Contract Symbol
contract_code	Clearing House Contract Symbol
product_code	Product Code
closing_px_date	Date utilize for the closing price
total_long	Total Long positions open for a certain contract



total_short	Total short positions open for a certain contract
total_reserve_ote	Total reserved OTE for a certain contract
expiration_time	Expiration time of the contract
position_id	Position ID
qty	Quantity
px	Price
notional	Notional
reserve_margin_s	Reserve Margin for short position
reserve_margin_l	Reserve Margin for long position
et	Expiration Time of position
customer_account_ref	Customer Account Reference
cl_ord_id	Client Order ID

Example Response:

```
[
  {
    "account_id": "27ff6d34-523d-476d-9ad5-edeb373b83dc",
    "positions": [
      {
        "contract_symbol": "BTCUSD1W",
        "contract_code": "BTCW44",
        "product_code": "BTC",
        "closing_px_date": "2018-01-01",
        "total_long": "400.0",
        "total_short": "0.1",
        "total_reserve_ote": "0.0",
        "expiration_time": "2019-01-01T06:00:00.000Z",
        "positions": [
          {
            "position_id": "t2",
            "qty": "-2",
            "px": "6000.0",
            "notional": "6000",
            "reserve_margin_s": "0.2",
            "reserve_ote": "-2000.0",
            "et": "2018-01-01T06:00:00.000Z",
            "customer_account_ref": "customer_account_ref",
            "cl_ord_id": "cl_ord_id"
          },
          {
            "position_id": "t1",
            "qty": "1",
            "px": "4000.0",
            "notional": "-4000",

```



```

    "reserve_margin_l": "400.0",
    "reserve_ote": "1000.0",
    "et": "2018-01-01T06:00:00.000Z",
    "customer_account_ref": "customer_account_ref",
    "cl_ord_id": "cl_ord_id"
  }
]
}
]
}
]

```

3.10 Deposit Address Method

This method will return the address to which a client can deposit funds for a specified digital asset.

- **HTTP Request Type:** POST
- **Method:** /deposit_address
- **API security:** This API method requires an authentication token with Clearing API read permission.

Inputs

Field	Value
account_id	Account ID
asset_type	Asset Type
funds_designation	Types: N, P, S. (See Funds Designation section for reference)

Example Request:

```

requests.post(url="https://clearing.erisx.com/api/v1/deposit_address",
              headers={"Authorization": "Bearer " + token},
              json={"account_id": "27ff6d34-523d-476d-9ad5-edeb373b83dc",
                    "asset_type": "BTC",
                    "funds_designation": "N"})

```

Outputs

Field	Value
address	Address hash
asset_type	Digital Asset type
account_id	Account ID
state	Status of the request
funds_designation	Types: N, P, S. (See Funds Designation section for reference)

Example Response:

```

{
  "address": "2NFVP4gnh4j6GtW8bz2wpXijnWEJ8EAySRq",
  "asset_type": "TBTC",
  "account_id": "ac171a7c-a0de-4e8a-9ce6-8a83d7e3ddd8",
  "state": "succeeded",
  "funds_designation": "N"
}

```

3.11 Linked Account Methods

This method will return information regarding any digital asset or bank accounts linked to the appropriate clearing member.

- **HTTP Request Type:** POST
- **Method:** /linked_accounts
- **API security:** This API method requires an authentication token with Clearing API read permission.

Inputs

Field	Value
filters (optional)	Default: <code>"filter": [{ "attr": "account_id", "op": "eq", "value": member_account_id }]</code>
	id ID identifying a particular digital asset address or bank account.
	member_id Clearing Member ID
	asset_type Symbol corresponding to the appropriate asset
	label Label given to the linked account when it was input into the system
	state State of the request to add a new linked account. Valid values: pending, approved, rejected
	usable_at Time at which the linked account is ready for the user to use
	type Type of account. Valid values: bank, crypto, collateral
offset (optional)	Number of elements to be offset in the request for pagination purposes
limit (optional)	Limit of elements returned in the request

Example Request :

```

requests.post(
  url="https://clearing.erisx.com/api/v1/linked_accounts",
  headers={"Authorization": "Bearer " + token},
  json={
    "filter": [{
      "attr": "id",

```

```

    "op": "eq",
    "value": ["5e4bef801ef35c09af0b42ce", "5e4bef911ef35c2fbf0b42d0"]
  }],
})

```

Outputs

Field	Value
count	Count of linked accounts returned in the response
accounts	List of linked accounts
label	Label of the linked account as input in the system by the user
asset_type	Asset type of the linked account
usable_at	Time from which the linked account is usable to use
member_id	ID of the clearing member to which the linked account belongs
state	Status of the request (pending, approved, rejected)
id	ID of the linked account. This id will be the one used in the withdrawal endpoint to specify to which linked accounts the funds be withdrawn
created_at	Time at which the linked account was added to the system
type	Type of the linked account (crypto, bank or collateral)
address	For digital asset's linked accounts, address hash of the wallet
account_number	For bank accounts, last 4 digits of account number
routing_number	For bank accounts, routing number
bank_name	For bank accounts, bank name

Example Response:

```

{ "count": 2, "accounts": [ {
  "label": "External ETH Wallet",
  "asset_type": "ETH",
  "usable_at": "2020-02-21T14:07:14.855Z",
  "member_id": "5e2b07559228bfd8841fd0ad",
  "state": "pending",
  "address": "bbbbbbb",
  "id": "5e4bef911ef35c2fbf0b42d0",
  "created_at": "2020-02-18T14:07:14.859Z",
  "type": "crypto"
},{
  "label": "External Checking 0000",
  "asset_type": "USD",
  "usable_at": "2020-02-21T14:07:00.363Z",
  "member_id": "5e2b07559228bfd8841fd0ad",
  "state": "pending",
  "account_number": "0000",

```



```
"routing_number": "011401533",  
"bank_name": "Chase",  
"id": "5e4bef801ef35c09af0b42ce",  
"created_at": "2020-02-18T14:07:00.365Z",  
"type": "bank"  
}]}
```



3.12 Withdrawal Request Methods

This section describes the procedure for a clearing member to request a withdrawal via the Clearing API. It is composed of two endpoints. First a request to an endpoint will be required where the clearing member specifies the details of the transaction, this request will return a response including all the necessary information that composes a valid transaction. A second request to the second endpoint is then required, where the user will specify the transaction message, which is the response from the first requests, signed by a secure hash of the funding password following the procedure indicated in the Funding Password Signature section.

3.12.1 Build Withdrawal Request

This method will enable member users to retrieve all necessary information in order to submit a withdrawal request via the Clearing API.

- **HTTP Request Type:** POST
- **Method:** /build_withdrawal_request
- **API security:** This API method requires an authentication token with Clearing API Funding permissions.
- **Notes:**
 - This request will not initialize a withdrawal request. It will only provide the data required to initialize a withdrawal.
 - The asset of the withdrawal will be inferred based on the Linked Account ID provided. I.e. if the clearing member specified a Linked Account ID that corresponds to a BTC linked account, the method will infer that the clearing member wants to withdraw BTC.

Inputs

Field	Value
account_id	Account ID from which the withdrawal will be made
linked_account_id	Linked Account ID to which the withdrawal will be sent. This value can be found in the field 'id' from the linked_accounts endpoint response.
funds_designation	Types: N, P, S. (See Funds Designation section for reference)
amount	Amount that will be withdrawn

Example Request:

```
requests.post(
  url="https://clearing.erisx.com/api/v1/build_withdrawal_request",
  headers={"Authorization": "Bearer " + token},
  json={'account_id': '48b7d9c5-55c5-4693-b5ec-10a97f7b2333',
        'linked_account_id': '5e4bef4b1ef35c96160b42cb',
        'funds_designation': 'S',
        'amount': '0.001'})
```

Outputs

Field	Value
account_id	Account ID from which account the withdrawal will be made
auth_id	Authentication ID needed to generate the signature in the Submit Withdrawal Request method
linked_account_id	Linked Account ID to which the withdrawal will be sent.
asset_type	Asset Type of the withdrawal. Inferred based on the linked_account_id provided
funds_designation	Types: N, P, S. (See Funds Designation section for reference)
amount	Amount of the withdrawal request
request_data	Base64-encoded withdrawal transaction as specified with the parameters above.

Example Response :

```
{
  "auth_id": "auth0|5e2b2eaeb9f8b40eaf22ec20",
  "account_id": "48b7d9c5-55c5-4693-b5ec-10a97f7b2333",
  "linked_account_id": "5e4bef4b1ef35c96160b42cb",
  "asset_type": "TBTC",
  "amount": "0.001",
  "funds_designation": "N",
  "request_data":
  "WyI1ZTJiMDc1NTkyMjhiZmQ4ODQxZmQwYWQiLCJhdXR0MHw1ZTJiMmVhZWl5ZjhiNDBlYWYyMmVjMjAiLCJhYWZhYWFhYSIsI1RCVEMiLCIwLjAwMSIsIjE1ODMxODk2NTk0OTYiLCI0OGI3ZDljNS01NWw1LTQ2OTMtYjVlYy0xMGE5N2Y3YjIzMzMiLCI1ZTRiZWY0YjFlZjZjM1Yzk2MTYwYjQyY2IiLCJTI10="
}
```

3.12.2 Submit Withdrawal Request

This method enables member users to submit a withdrawal request.

- **HTTP Request Type:** POST
- **Method:** /submit_withdrawal_request
- **API security:** This API method requires an authentication token with Clearing API Funding permissions as well as Funding Password signature security.

Inputs

Field	Value
request_data	Base64-encoded transaction data for the withdrawal that will be submitted. This value can be obtained from the response of the method 'build_withdrawal_request' in the field 'transaction_data'
signature	Signature created using the member user's funding password as described in the section Funding Password Signing

Example Request:

```
requests.post(  
    url="https://clearing.erisx.com/api/v1/submit_withdrawal_request",  
    headers={"Authorization": "Bearer " + token},  
    json={"request_data":  
        'WyI1ZTJiMDc1NTkyMjhiZmQ4ODQxZmQwYWQlLCJhdXRoMHw1ZTJiMmVhZWl5ZjhiNDBlYWYyMmVjMjAiLC  
        JhYWZhYWFhYSIsI1RCVEMiLCIwLjAwMSIsIjE1ODMxODk2NTk0OTYiLCI0OGI3ZDljNS01NWM1LTQ2OTMtY  
        jVlYy0xMGE5N2Y3YjIzZmZiLCI1ZTRiZWY0YjF1ZjM1Yzk2MTYwYjQyY2IiLCJTI10=',  
        "signature":  
        'AN1rKvtNrCF7mWSmbQ3hc4gUtY57C8CREkRNVU3cDdE8QibHgrcvTRmnmjU1SZpX7hDcWr9r8E6Q4Z7HWF  
        U3JGnTuMDdHPyG4 '})
```

Output

Field	Value
request_data	Transaction data for the submitted withdrawal

Example Response:

```
{  
  "request_data":  
  "WyI1ZTJiMDc1NTkyMjhiZmQ4ODQxZmQwYWQlLCJhdXRoMHw1ZTJiMmVhZWl5ZjhiNDBlYWYyMmVjMjAiLC  
  JhYWZhYWFhYSIsI1RCVEMiLCIwLjAwMSIsIjE1ODMxODk2NTk0OTYiLCI0OGI3ZDljNS01NWM1LTQ2OTMtY  
  jVlYy0xMGE5N2Y3YjIzZmZiLCI1ZTRiZWY0YjF1ZjM1Yzk2MTYwYjQyY2IiLCJTI10="  
}
```

4 Block Trade API Service (NEW)

This API service enables members to submit for processing negotiated Spot and Futures Block Trades in a programmatic way. All requests and responses are application/json content type.

All Block Trade API methods are private and every request needs to be signed using the authentication method described in the [Authentication](#).

4.1 REST API Endpoint URL

- **Production:** <https://clearing.erisx.com/api/v1>
- **New Release (test):** <https://clearing.newrelease.erisx.com/api/v1>

4.2 Block Trade States

A Block Trade submission can have different states during its processing life:

- **Accepted:** Block Trade has been accepted and processed in The Match Engine.
- **Cleared:** Block Trade has been cleared by The Clearing House.
- **Rejected:** Block Trade has been rejected.

4.3 Error Codes

Upon submission of a Block Trade through the Submit Block Trade Method, the Block Trade can be rejected for different reasons. In that case, the system will respond with a HTTP error, the response will contain “**state**” equal to “**rejected**” and it will reflect an “**error_code**” specifying the cause of the rejection. The table below explains what each error code represents for easier interpretation:

Error Code	Rejection Type
1	Unknown Symbol
2	Invalid Quantity
3	Invalid Price
4	Invalid Negotiated Block Trade Time
5	Instrument Closed
6	Trading is Halted
7	Invalid Buyer Account ID
8	Invalid Seller Account ID
9	Buyer Account Not Enabled for Futures
10	Seller Account Not Enabled for Futures
11	Buyer Account Not Enabled for Spot



12	Seller Account Not Enabled for Spot
13	Duplicate Request
14	Buyer Missing Regulatory Field
15	Seller Missing Regulatory Field
16	Buyer Insufficient Purchasing Power
17	Seller Insufficient Purchasing Power
18	Rejected

4.4 Duplicate Requests

The Match Engine will identify a request as a duplicate if all values specified in the request are identical to a previously sent request. If any of the values specified in the request change, it will not be considered as a duplicate anymore.

Note: If a user sends a request, which gets rejected with any error code and then the user attempts to send the exact same request again, the second request will always be rejected due to a duplicate request.

To avoid generating duplicate requests, it is recommended that the users generate new Client Order Ids on every new request, even if the request is a resubmission due to the original request being rejected.

4.5 Submit Block Trade Method

This method enables member users to submit a Block Trade.

- **HTTP Request Type:** POST
- **Method:** /submit_block_trade
- **API security:** This API method requires an authentication token with Block Trade API permissions.

Inputs

Field	Value
contract_symbol	Contract Symbol
price	Price
quantity	Quantity
negotiated_time	Time at which the Block Trade was negotiated between the Buyer and Seller
sell_side	Details regarding the Seller Account
buy_side	Details regarding the Buyer Account
account_label	Account Label
cl_ord_id	Client Order ID

Customer_account_ref (conditional)	Customer Account Reference
sender_sub_id	Sender Sub ID that identifies the Trader that negotiated the Block Trade
sender_location	Sender Location that identifies the location of the Trader that negotiated the Block Trade

Example Request:

```
requests.post(
  url="https://clearing.erisx.com/api/v1/submit_block_trade",
  headers={"Authorization": "Bearer " + token},
  json={
    "contract_symbol": "BTC/USD",
    "price": "1000.0",
    "quantity": "3.0",
    "negotiated_time": "2020-01-01T14:27:00.000Z",
    "sell_side": {
      "account_label": "DM-122221",
      "cl_ord_id": "sell_side_order",
      "customer_account_ref": "sell_side",
      "sender_sub_id": "sell_fix",
      "sender_location": "US,IL"
    },
    "buy_side": {
      "account_label": "DM-77661",
      "cl_ord_id": "buy_side_order",
      "customer_account_ref": "buy_side",
      "sender_sub_id": "buy_fix",
      "sender_location": "US,IL"
    }
  }
})
```

Output

Field	Value
request_id	Request ID of the Block Trade
state	State of the Block Trade submission
error_code	Rejection Error Code. See Error Codes for reference

Example Responses:

```
{
  "request_id": "AmZuyvUdJvoa3HshZuhyCosBSvnrXVbZfJr5RtA92EQT",
  "state": "accepted"
}

{
  "request_id": "AmZuyvUdJvoa3HshZuhyCosBSvnrXVbZfJr5RtA92EQT",
```

```

    "state": "rejected",
    "error_code": "1"
  }

```

4.6 Request Block Trade Information Method

This method enables member users to request information regarding a previously submitted Block Trade.

- **HTTP Request Type:** POST
- **Method:** /block_trade_requests
- **API security:** This API method requires an authentication token with Submit Block Trade API permissions.

Inputs

Field	Value
filters (optional)	Default: "filter": [{ "attr": "account_id", "op": "eq", "value": member_account_id }]
request_id	Block Trade submission request ID. Value provided in the response of the Submit Block Trade Method .
contract_symbol	Contract Symbol
trade_date	Trade Date
state	State of the Block Trade. Values: accepted, cleared, rejected.
account_labels	Account Label(s)
offset (optional)	Number of elements to be offset in the request for pagination purposes
limit (optional)	Limit of elements returned in the request
sort (optional)	Default: "sort": [{ "attr": "time", "value": "desc" }]

Example Request:

```

requests.post(
  url="https://clearing.erisx.com/api/v1/block_trade_requests",
  headers={"Authorization": "Bearer " + token},
  json={
    "filter": [{
      "attr": "request_id",
      "op": "eq",
      "value": "5c267f0ee4b0974b5367fd35"
    }], {
      "attr": "contract_symbol",
      "op": "eq",
      "value": "BTC/USD"
    }, {

```

```

    "attr": "trade_date",
    "op": "eq",
    "value": "2020-01-01"
  }, {
    "attr": "state",
    "op": "eq",
    "value": "pending"
  }, {
    "attr": "account_labels",
    "op": "eq",
    "value": "27ff6d34-523d-476d-9ad5-edeb373b83dc"
  }
}]
})

```

Output

Field	Value
count	Count of Block Trades that meet the query filter
block_trade_requests	List of Block Trades Requests
request_id	Block Trade submission request ID
contract_symbol	Contract Symbol
qty	Quantity
px	Price
trade_date	Trade Date
negotiated_time	Time at which the Block Trade was negotiated between the Buyer and Seller
submitted_time	Time at which the Block Trade has submitted
created_by	Member user who submitted the Block Trade
state	State of the Block Trade
buy_side	Details regarding the Seller Account
sell_side	Details regarding the Buyer Account
account_id	Account ID
account_label	Account Label
side	Side
customer_account_ref	Customer Account Reference
cl_ord_id	Client Order ID
sender_sub_id	Sender Sub ID that identifies the Trader that negotiated the Block Trade
sender_location	Sender Location that identifies the location of the Trader that negotiated the Block Trade

Example Response:

```
{
  "count": 1,
  "block_trade_requests": [
    {
      "request_id": "AmZuyvUdJvoa3HshZuhyCosBSvnrXVbZfJr5RtA92EQT",
      "contract_symbol": "BTC/USD",
      "qty": "5.0",
      "px": "8900.0",
      "trade_date": "2020-01-01",
      "negotiated_time": "2020-01-01T06:00:00.000Z",
      "submitted_time": "2020-01-01T06:00:00.000Z",
      "created_by": "carl.doe@email.com",
      "state": "pending",
      "buy_side": {
        "account_id": "27ff6d34-523d-476d-9ad5-edeb373b83dc",
        "side": "BUY",
        "customer_account_ref": "buy_side",
        "cl_ord_id": "buy_side",
        "sender_sub_id": "buy_side_fix",
        "sender_location": "IL,US"
      },
      "sell_side": {
        "account_id": "73e36d47-0fe6-4bba-84d7-d981d9f9459d",
        "side": "SELL",
        "customer_account_ref": "sell_side",
        "cl_ord_id": "sell_side",
        "sender_sub_id": "buy_side_fix",
        "sender_location": "IL,US"
      }
    }
  ]
}
```

4.7 Block Trade Volume Method

This method enables member users to obtain information regarding the total volume that the different products have traded via Block Trades in the ErisX market for the current trade date.

- **HTTP Request Type:** POST
- **Method:** /block_trade_volume
- **API security:** This API method requires an authentication token with Market Data API permissions.

Inputs

No inputs are required.

Example Request:

```
requests.post(  
    url="https://clearing.erisx.com/api/v1/block_trade_volume",  
    headers={"Authorization": "Bearer " + token},  
    json={})
```

Output

Field	Value
result	List of Products
contract	Contract Symbol
volume	Total Volume
trade_date	Trade Date

Example Response:

```
[  
  {  
    "contract": "BTC/USD",  
    "volume": "20.0",  
    "trade_date": "2020-01-01"  
  }  
]
```