Cboe Silexx API User Docs [25.4]

Introduction

Cboe Silexx API offers a robust platform for traders and institutions to execute orders and manage portfolios. Our API provides a flexible and scalable solution to integrate trading capabilities into your applications.

By leveraging the Cboe Silexx API, traders can:

- Execute a wide range of order types across multiple asset classes.
- Manage and monitor a portfolio efficiently.
- Benefit from advanced order routing and execution capabilities.

This documentation provides comprehensive guidance on integrating with the CBOE Silexx API, including API endpoints, data structures, authentication, and best practices.

About Cboe Silexx

Cboe Silexx is a comprehensive order and execution management system (OEMS) offered by Cboe Global Markets. The platform empowers buy-side and sell-side traders with advanced tools for order management, order risk analysis, and portfolio monitoring.

Cboe Silexx Client API Platform

Cboe Silexx API provides a robust, flexible platform for traders to efficiently manage their portfolios and execute orders across options, FLEX, equities, ETFs, and futures. Built on a foundation of gRPC, our API mirrors the functionality of the Silexx front end, enabling developers to seamlessly integrate trading capabilities into their applications.

Architecture

The Silexx API is a core component of the platform, working in conjunction with the front-end OEMS, certification, and production environments. We offer multiple interface options (gRPC, REST, .NET) to accommodate diverse developer preferences and project requirements.

Streaming Services

The gRPC API and .NET API both support streaming services. With the gRPC API, users will need to subscribe to updates whereas the .NET API is automatically subscribed upon login and the user will need to add a listener.

API Updates & Upgrades

Cboe Silexx is dedicated to ongoing API development and improvement. We regularly introduce new features and enhancements, communicated through product release notes. To ensure a smooth transition, we provide ample notice for any changes in functionality which may impact existing workflows.

Installation

gRPC

1. Obtain proto files from Silexx-Support@cboe.com OR use server reflection
2. Helpful resources for interacting with a gRPC API
3. For quick start examples and tutorials in various languages, follow this guide: Supported languages | gRPC

REST

There is no explicit installation required to begin working with our REST API. Please continue to the Authentication section for logging into the REST API.

.NET API

Contact Silexx-Support@cboe.com for portfolio and order example code

---

Cboe Silexx Client API

About

The Cboe Silexx Client API offers a comprehensive set of endpoints for submitting, modifying, and managing orders across various asset classes. The API is designed to provide flexibility and efficiency for traders, allowing seamless integration into existing trading systems.

While the following examples primarily focus on the gRPC API, the underlying message structures are mostly identical for both REST and gRPC API calls. The differences are detailed in the REST quick start section.

Key features of the API include:

- **Comprehensive order support:** Support for a wide range of order types and crosses for options, FLEX instruments, equities, and futures
- **Streaming order and portfolio updates:** With gRPC, stream real-time updates on new orders, fills, portfolio changes, and other related events
- **Certification Environment:** Sandbox environment for testing

By leveraging the Cboe Silexx Client API, you can build sophisticated trading applications that benefit from the platform's advanced order management capabilities and low-latency execution.

Certification Environment

The Cboe Silexx API offers a dedicated **certification environment** designed to facilitate smooth and efficient testing before deploying your code to production. This environment provides a realistic testing ground with the following key features:

- **Simulated Trading:** The certification environment simulates real-world trading conditions allowing developers to test order placement, execution, and other functionalities without impacting live markets.
- **Sandbox Environment:** The environment operates as a secure **sandbox**, ensuring testing activities do not affect production data or order flow.
- **Identical API Endpoints:** The API endpoints within the certification environment mirror those in production, enabling developers to test code functionality without modifying endpoint URLs or request structures.
- **Unique Credentials:** Separate login credentials are provided for the certification environment to maintain a clear distinction from production access.
- **Configurable Accounts:** Accounts within the certification environment can be configured to mimic the production environment setup thereby enabling realistic workflow testing.
- **Early Access to Features:** New features and updates are first deployed to the certification environment, providing developers with early access for testing and code adjustments before the production rollout.

By leveraging the Cboe Silexx API certification environment, developers can gain valuable confidence in their code's functionality before deploying to the live platform. This proactive approach minimizes potential disruptions and ensures a smooth transition to real-world trading.

To access the Cboe Silexx API CERT Environment, please contact Silexx-support@cboe.com

Supported Asset Classes

The Cboe Silexx API provides comprehensive support for a wide range of asset classes, mirroring the capabilities of our front-end OEMS. This enables traders to execute a diverse array of trading strategies through a unified platform.

Supported Asset Classes:

- **Options:** Execute both single-leg and multi-leg option orders on equities and indices.
- **FLEX Options:** Create custom option strategies with flexibility in strike price, expiration date, and other parameters.
- **Equities:** Trade long or short on individual stocks.
- **Exchange-Traded Funds (ETFs):** Access a variety of ETF products for diversified investment strategies.
- **American Depository Receipts (ADRs):** Trade options and the underlying equity for listed ADRs.
- **Compound Assets:** Trade listed or FLEX options with an equity leg.
- **Futures:** Trade futures contracts on various underlying assets, including commodities, indices, and volatility.
- **Options on Futures:** Trade options contracts on futures contracts, allowing for complex hedging and speculative strategies.

By offering a broad spectrum of asset classes, the Cboe Silexx API empowers traders to efficiently manage their portfolios and capitalize on market opportunities across multiple asset categories.

Supported Orders and Routes

Supported Order Types

The Cboe Silexx API offers a comprehensive suite of order types to accommodate diverse trading strategies and market conditions. While the exact range of order types may vary by asset class and exchange, the API supports the following:

Order Type
ORD_TYPE_MARKET
ORD_TYPE_LIMIT
ORD_TYPE_STOP
ORD_TYPE_STOP_LIMIT

Crossing Orders

The Cboe Silexx API supports crossing orders, allowing clients to match buy and sell orders by specifying contra accounts. This feature provides efficient execution and cost savings for large block trades.

Supported Routes

The Cboe Silexx API offers a flexible routing infrastructure, enabling clients to execute orders across multiple venues to achieve optimal pricing, liquidity, and execution quality.

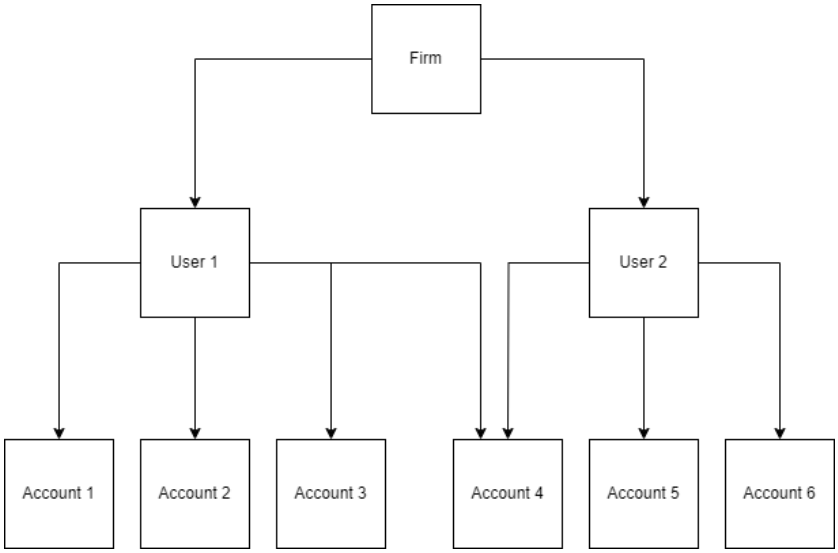
- **Direct Market Access (DMA):** Provides direct access to exchanges, allowing for high-speed, low-latency execution.
- **Broker Routes:** Leverages the broker's expertise and relationships to access a variety of venues, including exchanges, dark pools, and internal crossing networks.
- **Peer-to-Peer (P2P):** Facilitates trades between other Silexx users.
- **Staged Routing:** Orders can be staged and derived into child orders. Staged orders may be sent or received via FIX or used as an execution workflow tool.
- **Queued Routing:** Queuing an order puts it into a pending state and will become a live order when routed.
- **Manual Routing:** Provides traders with a workflow to submit orders that has been executed elsewhere.
- **Algorithmic Routing:** Employs sophisticated algorithms to dynamically route orders based on real-time market data and pre-defined parameters, optimizing execution quality.

Note: The specific routes available may vary based on asset class, order type, and client permissions.

By offering a diverse range of routing options, the Cboe Silexx API empowers traders to tailor their execution strategies to meet specific trading objectives and market conditions.

Firm, Account, and User IDs

ID's are a key component of the Cboe Silexx API that developers and traders should familiarize themselves when developing applications.



Trading Firm ID

`trading_firms` are used to identify the firm in which a user is associated with.

Invoke: `UserDataService / ListTradingFirms`

```
    "trading_firms": [
        {
            "id": 1,
            "eq_mpid": "",
            "name": "SILEXX",
            "occ_clearing_number": ""
        }
    ],
```

A key use of the trading firm ID is for accessing trade confirms at the firm level.

Invoke: `OrderService / ListTradeConfirms`

```
{
    "trading_firm_ids": [
        1
    ],
    "start_time_utc": {
        "seconds": 1724025600,
        "nanos": 0
    },
    "end_time_utc": {
        "seconds": 1729523200,
        "nanos": 0
    }
}
```

The successful response of will contain trade information for all accounts under a firms umbrella.

User ID

The User ID is referring to who is currently logged into the API platform. Users are categorized under a firm and have associated accounts that they are permissioned to trade.

The current user ID can be accessed in a variety of ways

1. The `Login` response will contain the User ID `"id": 720`
2. The User ID can be accessed by invoking `UserDataService / GetActiveUser` and a successful message will contain the User ID `"id": 720`

While User ID is not explicitly used, all methods under `User Data Service` will show information based on the current active user.

Account ID

Account IDs are a key component of order and portfolio services.

Invoke: `UserDataService / ListAccounts`

```
{
        {
            "id": 1232,
            "name": "c-tmichalove",
            "trading_firm_id": 1,
        },
        {
            "id": 871,
            "name": "c-randerson",
            "trading_firm_id": 1,
        }
}
```

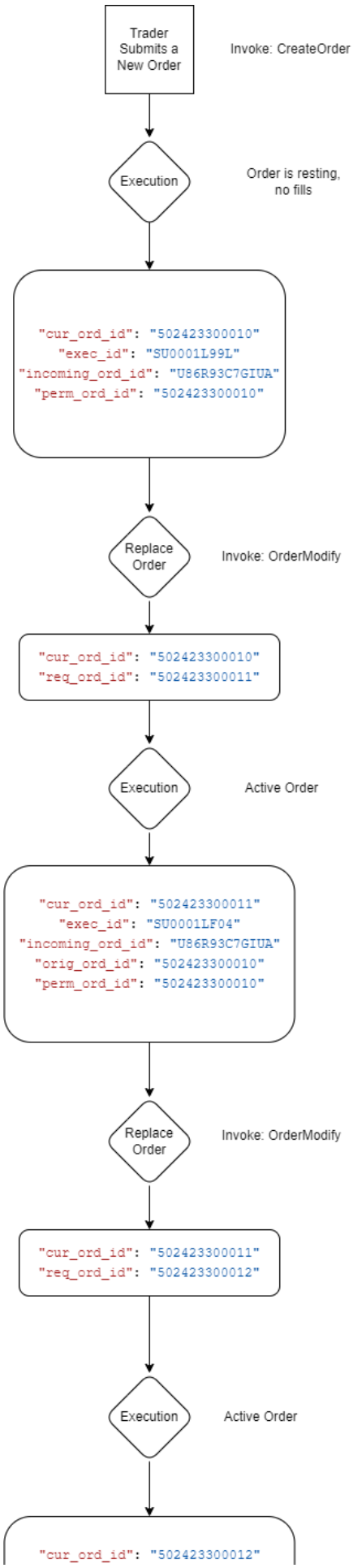
The above response has been paired down to show basic information. The actual response will contain many fields associated with each account such as EFIDs, MPIDs, Clearing Range and Account Type.

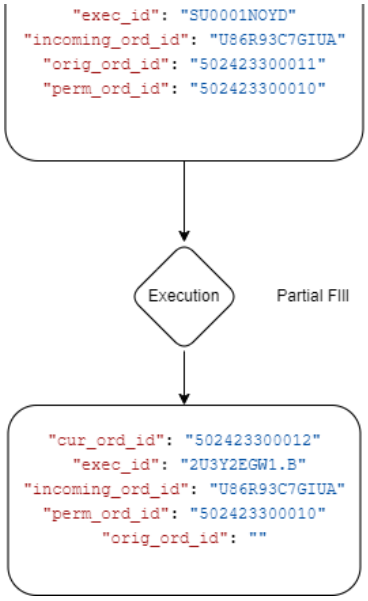
The account ID is required for placing, modifying, and cancelling orders. It can also be used when searching for trade confirms and order history.

Order IDs

ID	Description
`cur_ord_id`	This value represents the current order. A new value will be given every time an order is placed or modified
`id`	Returned when submitting a new order.
`parent_ord_id`	Used to identify the parent order for when route = STAGE
`trader_id`	text based id commonly used in the front end Example: c-tmichalove
`perm_ord_id`	Represents the first ID generated when a new order is created
`orig_ord_id`	When replacing an order, this shows the previous order ID
`req_ord_id`	returned in the replace order response. Used show identify what the new cur_ord_id will be
`exec_ord_id`	Unique identifier for each execution event

Tracing Order ID's Though the `ListOrders` Response


Trader Submits a New Order
Invoke: CreateOrder
Execution
Order is resting, no fills
"cur_ord_id": "502423300010"
"exec_id": "SU0001L99L"
"incoming_ord_id": "U86R93C7GIUA"
"perm_ord_id": "502423300010"
Replace Order
Invoke: OrderModify
"cur_ord_id": "502423300010"
"req_ord_id": "502423300011"
Execution
Active Order
"cur_ord_id": "502423300011"
"exec_id": "SU0001LF04"
"incoming_ord_id": "U86R93C7GIUA"
"orig_ord_id": "502423300010"
"perm_ord_id": "502423300010"
Replace Order
Invoke: OrderModify
"cur_ord_id": "502423300011"
"req_ord_id": "502423300012"
Execution
Active Order
"cur_ord_id": "502423300012"



Authentication

The Cboe Silexx API employs a secure token-based authentication mechanism to ensure the confidentiality and integrity of API interactions. This section outlines the authentication process and best practices.

API Hostname

All gRPC API requests are directed to the Cboe Silexx API hostname:

Environment	hostname
Chicago Certification	c-gw-api-ch-cboe.silexx.com
New York Certification	c-gw-api-ny-cboe.silexx.com
Chicago Production	gw-api-ch-cboe.silexx.com
New York Production	gw-api-ny-cboe.silexx.com

Base REST URL:

Environment	Base URL
Chicago Certification	https://c-gw-api-ch-cboe.silexx.com/api/v1
New York Certification	https://c-gw-api-ny-cboe.silexx.com/api/v1
Chicago Production	https://gw-api-ch-cboe.silexx.com/api/v1
New York Production	https://gw-api-ny-cboe.silexx.com/api/v1

Token-Based Authentication

Every private API call requires a valid authentication token. This token is obtained through a successful login request to either the certification or production environment. **Tokens have a limited lifespan of one hour** and require renewal upon expiration.

gRPC Login Process

To initiate a login session and obtain an authentication token, invoke the `ApplicationService/Login` method with the following request body:

```
{
    "application_name": "Your application name",
    "application_version": "Your application version",
    "domain": "silexx",
    "password": "{{password}}",
    "username": "{{username}}"
}
```

- Replace the placeholder values `{password}`, `{username}` with your actual credentials.
- The `application_name` and `application_version` fields are user defined and can be configured to best suit development needs.
- The domain field is a fixed value specific to the Silexx platform.

REST Login Process

The message structure is the same as gRPC but the endpoint is slightly different:

`/application/login`

Either the Chicago or New York URL can be used. A successful login will return a token that is valid for one hour. Users will need to log in again after the token expires to generate a new one.

Quick Start Examples for gRPC API

The following examples will cover the basics for placing an order, modifying, cancelling, and accessing account information. The underlying message structure for the gRPC API and REST APIs are identical. For additional information about the REST API, please review the Quick Start Example for REST API.

Example: Retrieving Account Information

To begin interacting with the CBOE Silexx API after logging in, retrieve a list of assigned accounts. This information is essential for subsequent API calls.

Invoke: `UserDataService/ListAccounts`

This API call returns a list of accounts associated with the authenticated user. Each account includes an `ID` field, which is a unique identifier used to specify the account in other API requests.

Example Response:

```
{
            "routing_session_groups": [
                {
                    "id": 9,
                    "name": "C1.0001",
                    "type": "ROUTING_SESSION_GROUP_TYPE_EXCHANGE"
                },
                {
                    "id": 4,
                    "name": "BZX",
                    "type": "ROUTING_SESSION_GROUP_TYPE_EXCHANGE"
                }
            ],
            "id": 1232,

            "billing_code": "",
            "cti_code": "",
            "currency": "CURRENCY_USD",
            "equities_capacity": "CAPACITY_CODE_NOT_SET",
            "equities_give_up": "",
            "equities_mpid": "",
            "futures_clearing_account": "",
            "futures_clearing_range": "CLEARING_RANGE_NOTSET",
            "futures_cmta": "",
            "futures_execution_broker": "",
            "name": "c-tmichalove",
            "occ_actionable_id": "",
            "options_clearing_account": "",
            "options_clearing_range": "CLEARING_RANGE_NOTSET",
            "options_cmta": "",
            "options_efid": "",
            "options_give_up": "",
            "status": "ACCOUNT_STATUS_ACTIVE",
            "tag1": "",
            "trading_firm_id": 1,
            "type": "ACCOUNT_TYPE_CASH",
            "created_ts": {
                "seconds": "1707491480",
                "nanos": 492000000
            },
            "optional_data": ""
        },

```


Example: Submitting a Single Leg Option Order

Once the desired account `ID` is determined, construct a single-leg option order message. This message typically includes fields for:

- **Account ID:** Identifies the account to place the order.
- **Order Type:** Specifies the order type (e.g., Market, Limit, Stop).
- **Side:** Indicates whether it is a buy or sell order.
- **Symbol:** The securities unique identifier.
- **Expiration Date:** The option's expiration date.
- **Strike Price:** The option's strike price.

- **Quantity:** The number of contracts to trade.
- **Order Price:** The order price (for Limit or Stop Limit orders)
- **Aux Price**: The stop price for Stop or Stop Limit orders
- **Time in Force (TIF):** Specifies the order's validity period (e.g., Day, Good Till Canceled).

Example Order Message:

Invoke: `OrderService / CreateOrder`

```
{
    “account_id”: 1232,
    “ord_type”: “ORD_TYPE_LIMIT”,
    “position_effect”: “POSITION_EFFECT_OPEN”,
    “price”: {"value": 3.00},
    “price_type”: “PRICE_TYPE_PER_UNIT”,
    “qty”: 100,
    “route”: “CBOE”,
    “side”: “SIDE_BUY”,
    “symbol”: {"value": “SPY/240717/558C”},
    “tif”: “TIF_DAY”
}

```

With the provided order message, a limit order to buy 100 contracts of the SPY July 17, 2024 558 Call option at a price of 3.00 will be submitted.

Order Validation (Optional)

Before submitting an order, a risk check can be performed using the `OrderService / ValidateCreateOrderRisk` method. This verifies that the order complies with the account's buying power, quantity limits, and other risk parameters. Pass the same order message as above to the validate risk method.

Silexx admins can configure risk limit settings within the front-end application.

Order Monitoring

To track the status of the order, including fills, cancellations, and modifications, subscribe to order updates using the `OrderService/SubscribeToOrderUpdates` method. This service provides real-time notifications about order events.

Order Cancellation

To cancel the order, use the `OrderService/CancelOrder` method, providing the `cur_ord_id` obtained from the latest order creation response.

```
{
    “cur_ord_id”: “402419700009”
}
```

Example: Selling Short

Selling short is similar to placing a single leg order with the additional `locate_id` field. The below message will place an order to sell short NVDA at 131.30

Invoke: `OrderService / CreateOrder`

```
{

"clearing_info": {

    "locate_id": {"value": "TEST"}
},

    "account_id": 1232,
    "ord_type": "ORD_TYPE_LIMIT",
    "position_effect": "POSITION_EFFECT_AUTO",
    "price": {"value": 131.30},
    "price_type": "PRICE_TYPE_PER_UNIT",
    "qty": 1,
    "route": "MANUAL",
    "side": "SIDE_SELL_SHORT",
    "symbol": {"value": "NVDA"},
    "tif": "TIF_DAY"
}
```

Example: Submitting and Modifying a Multileg Option Order

To create a multi-leg option order, use the `OrderService/CreateMultiLegOrder` method. The request body includes details about the order's legs, order type, price, quantity, and routing instructions.

Example Request:

Invoke: `OrderService/CreateMultiLegOrder`

```
{
    "account_id": 1232,
    "legs": [
        {
            "position_effect": "POSITION_EFFECT_OPEN",
            "ratio": 1,
            "side": "SIDE_BUY",
            "symbol": {"value": "SPY/240830/575C"}
        },
        {
            "position_effect": "POSITION_EFFECT_OPEN",
            "ratio": 1,
            "side": "SIDE_SELL",
            "symbol": {"value": "SPY/240830/575P"}
        }
    ],

    "ord_type": "ORD_TYPE_LIMIT",
    "price": {"value": -44.00},
    "price_type": "PRICE_TYPE_PER_UNIT",
    "qty": 100,
    "route": "CBOE",
    "tif": "TIF_DAY"
```

```
}
```

A successful order submission will return a `cur_ord_id` which uniquely identifies the order.

```
"cur_ord_id": "40242150001A",
```

Modifying a Multi-Leg Order

To modify an existing multi-leg order, use the `OrderService/ReplaceMultiLegOrder` method. The request body should include the `cur_ord_id` of the order to be modified, along with updated order parameters.

Example Request:

Invoke: `OrderService/ReplaceMultiLegOrder`

```
{
    "cur_ord_id": "40242150001A",
    "ord_type": "ORD_TYPE_LIMIT",
    "price": {"value": -44.10},
    "qty": 200,
    "tif": "TIF_DAY"
}
```

Example: Submitting and Routing a Staged Order

Creating a Staged Order

Staged orders provide a mechanism to temporarily hold orders before deriving child orders and routing them to specific execution venues. To create a staged order, use the `OrderService/CreateOrder` method with the route parameter set to `"STAGE"`.

Example Request:

Invoke: `OrderService/CreateOrder`

```
{
    "account_id": 1232,
    "ord_type": "ORD_TYPE_LIMIT",
    "position_effect": "POSITION_EFFECT_OPEN",
    "price": {"value": 0.12},
    "price_type": "PRICE_TYPE_PER_UNIT",
    "qty": 100,
    "route": "STAGE",
    "side": "SIDE_BUY",
    "symbol": {"value": "SPXW/240830/5330C"},
    "tif": "TIF_DAY"
}
```

The successful creation of a staged order will return an order ID.

Routing a Staged Order

To route a staged order to a specific execution venue, create a new order with the following parameters:

- **Parent Order ID:** The ID of the staged order to be routed.

- **Route:** The desired execution venue (e.g., "CBOE").

Example Request:

Invoke: `OrderService/CreateOrder`

```
{
    "parent_ord_id": "402422700002",
    "account_id": 1232,
    "ord_type": "ORD_TYPE_LIMIT",
    "position_effect": "POSITION_EFFECT_OPEN",
    "price": {"value": 0.12},
    "price_type": "PRICE_TYPE_PER_UNIT",
    "qty": 100,
    "route": "CBOE",
    "side": "SIDE_BUY",
    "symbol": {"value": "SPXW/240830/5330C"},
    "tif": "TIF_DAY"
}

```

This request will route the staged order with ID "402422700002" to the CBOE exchange.

By utilizing staged orders, order flow and execution logic can be effectively managed, providing greater flexibility in your trading strategies.

Example: Trade Confirms and Order History

Trade Confirms

To retrieve trade confirms, use the `OrderService/ListTradeConfirms` method. Results can be filtered as follows:

Field	Description	Example
user_ids	List of user IDs	Unique identifier for a user: “710”
trading_firm_ids	List of trading firm IDs	Unique identifier for a firm: “7”
account_ids	List of account IDs	Unique identifier for an account: “1232”
start_time_utc	UNIX Format	seconds: nanos:
end_time_utc	UNIX Format	seconds: nanos:

Example Request:

Invoke: `OrderService/ListTradeConfirms`

```
{
    "account_ids": [
       1232, 871
    ],
```

```
    "end_time_utc": {
        "nanos": 0,
        "seconds": 0
    },
    "start_time_utc": {
        "nanos": 0,
        "seconds": 0
    }
}
```

The response will contain a list of trade confirms matching the specified criteria.

Order History

To retrieve order history, use the `OrderService/SearchOrderHistory` method. Target accounts can be specified using `account_ids` and a date range defined by `from_dt` and `to_dt`.

Example Request:

Invoke: `OrderService/SearchOrderHistory`

```
{
    "account_ids": [
        1232
    ],
    "from_dt": {
        "day": 30,
        "month": 7,
        "year": 2024
    },
    "to_dt": {
        "day": 1,
        "month": 8,
        "year": 2024
    }
}
```

Example: Positions

Accessing Positions

The `PortfolioService/ListPositions` method provides three ways to retrieve position information:

Retrieving All Positions for one Account

To retrieve all positions for a specific account, use the following request format:

```
{
    "account_id": 1232,
}
```

Retrieving All Positions for All Accounts

The following message will return all positions across all accounts a user is authorized for.

```
{
    "all": true
}
```

Retrieving Positions by Account and Symbol

To retrieve positions for a specific account and symbol, use the following request format:

```
{
    "account_id_and_symbol": {
        "account_id": 1232,
        "symbol": "SPY/240823/558C"
    }
}
```

Gain valuable insights into portfolio composition and performance by effectively utilizing the `ListPositions` method.

This Quick Start Guide provides foundational examples to begin interacting with the Cboe Silexx API. It covers essential tasks such as order placement, modification, and retrieval of trade confirms and positions. While this guide offers a solid starting point, the API's full capabilities extend far beyond these examples. For a comprehensive understanding of all available request/response fields, please refer to the proto files or swagger documentation.

Quick Start Examples for REST API

The underlying message structure for REST API calls are identical to those of a gRPC call. For example messages, please review the gRPC Quick Start Examples.

There are two REST endpoints which do not take messages and require the use of parameters:

- `GET /portfolio/positions`
- `GET /orders/tradeconfirms`

`GET /positions`

Users can retrieve positions based on an account ID, or an account ID and symbol.

Parameter	Value	Description
accountIdAndSymbol.accountId	1232	account ID to be used with a symbol
accountIdAndSymbol.symbol	SPY	asset symbol to be used with account ID
accountID	SPY	account ID when used without a symbol

Example Request:

Retrieving positions based on account and symbol:

`{{baseUrl}}/portfolio/positions?accountIdAndSymbol.accountId=1232&accountIdAndSymbol.symbol=SPY`

`GET /tradeconfirms`

Trade confirms require the `startTimeUtc` and `endTimeUtc` parameters and one of the following: `accountIds`, `tradingFirmIds`, or `userIds`. The UTC timestamp that is used here differs from the Unix timestamp that the gRPC API uses.

Parameter	Value	Description
startTimeUtc	2024-08-30T19:20:20Z	Starting UTC timestamp
endTimeUtc	2024-09-03T19:20:20Z	Ending UTC timestamp
accountIds	1232	Unique account identifier
tradingFirmIds	1	Unique trading firm identifier
userIds	710	Unique user identifier

Example Request:

To define multiple `accountIds` use the following format: `accountIds=1&accountIds=2&accountIds=3`

`{{baseUrl}}/orders/tradeconfirms?accountIds=1232&accountIds=810&endTimeUtc=2024-09-03T19:20:20Z&startTimeUtc=2024-08-30T19:20:20Z`

Rate Limits

All services have a rate limit of 200 calls per minute. For example, a user can make 200 Order Service calls and 200 Securities Service calls within a 1-minute period. Limits are assessed within a rolling 1 minute window. Streaming gRPC responses only count as a single call when first subscribing.

Service	Limit
Application	200 /min
Orders	200 /min*
Portfolio	200 /min
Securities	200 /min
User Data	200 /min
Utility	200 /min

*We do not limit cancel order requests. Users will always be able to cancel orders regardless of if a limit is hit.

Additional Resources

- For help please contact silexx-support@cboe.com
- If you are using postman, we can provide a postman specific proto file for gRPC or a JSON file for REST endpoints.