TECHNICAL SPECIFICATIONS

Document title

Euronext Clearing Application Programming Interfaces Specifications

Document type or subject

EURONEXT CLEARING API SPECIFICATIONS

Revision number	Date
2.0	27/12/2022

Number of pages

95

This publication is for information purposes only and is not a recommendation to engage in investment activities. This publication is provided "as is" without representation or warranty of any kind. Whilst all reasonable care has been taken to ensure the accuracy of the content, Euronext does not guarantee its accuracy or completeness. Euronext will not be held liable for any loss or damages of any nature ensuing from using, trusting or acting on information provided. No information set out or referred to in this publication shall form the basis of any contract. The creation of rights and obligations in respect of financial products that are traded on the exchanges operated by Euronext's subsidiaries shall depend solely on the applicable rules of the market operator. All proprietary rights and interest in or connected with this publication shall vest in Euronext. No part of it may be redistributed or reproduced in any form without the prior written permission of Euronext.

Preface

PURPOSE

This document sets out the Euronext Clearing Application Programming Interfaces specifications. It describes the different interfaces offered by Euronext Clearing through which Clients can connect to the clearing systems.

This document has the purpose of providing more details in relation to the APIs introduced in the document "Euronext Clearing Interfaces Overview and Reporting".

This version of the document contains additional details regarding the clearing data that members can query. The description of the commands available to perform operational requests will be provided in the next version of the document.

Details relating to Connectivity will be provided in dedicated separate document, to be released in due course.

TARGET AUDIENCE

All Euronext clients that will adopt Euronext Clearing as their clearing house.

WHAT'S NEW?

The following lists only the most recent modifications made to this revision/version. For the Document History table, see the Appendix.

REVISION NO./ VERSION NO.	DATE	AUTHOR	CHANGE DESCRIPTION
1.0	02/11/2022		First Version
2.0	27/12/2022		▪ Updated Fields and Types on: ○ List Instruments API ○ List Trades API Client ○ List Positions API ○ List Settlement Positions API ○ List Reports API ▪ Added List Accounts API ▪ Added List Participants API ▪ Added List Collateral Deposit API ▪ Added Export Collateral Deposit API ▪ Added Collateral Deposit Feed API ▪ Added List Collateral Balance API ▪ Added Export Collateral Balance API ▪ Added Collateral Balance Feed API ▪ Added List Operational Requests API ▪ Added List Operational Requests Feed API

REVISION NO./ VERSION NO.	DATE	AUTHOR	CHANGE DESCRIPTION
			▪ Added List Operational Request Details API ▪ Added List Notification API ▪ Added Notification Feed API

ASSOCIATED DOCUMENTS

The following lists the associated documents that either should be read in conjunction with this document or that provide other relevant information for the user:

- Euronext Clearing Interfaces Overview and Reporting

Contents

1. INTRODUCTION 7
1.1 Glossary 7
1.2 Overview 8
2. API ACCESS 9
2.1 API Manager for Client Credentials 9
2.2 Token Structure 11
2.3 Token Generation 12
2.4 Token Expiration 13
2.5 API Authorisation 15
3. API USAGE 16
3.1 Interaction via Curl 16
3.2 Formatting a query 16
4. GRAPHQL SCHEMA 18
5. QUERIES 43
5.1 List Instruments API 43
5.1.1 Sample List Instruments Request 43
5.1.2 Sample List Instruments Response 44
5.2 Export Instruments API 45
5.2.1 Sample Export Instruments Request 45
5.2.2 Sample Export Instruments Response 45
5.3 List Trades API 46
5.3.1 Sample List Trades Request 46
5.3.2 Sample List Trades Response 47
5.4 Export Trades API 48
5.4.1 Sample Export Trades Request 48
5.4.2 Sample Export Trades Response 48
5.5 List Trades Historical Requests API 48
5.5.1 Sample List Trades Historical Request 48
5.5.2 Sample List Trades Historical Response 49
5.6 List Positions API 50
5.6.1 Sample List Positions Request 50
5.6.2 Sample List Positions Response 51
5.7 Export Positions API 53
5.7.1 Sample Export Positions Request 53
5.7.2 Sample Export Positions Response 53
5.8 List Position Historical Requests API 54
5.8.1 Sample List Position Historical Request 54
5.8.2 Sample List Historical Positions Response 54
5.9 List Settlement Positions API 55
5.9.1 Sample List Settlement Positions Request 55

5.9.2 Sample List Settlement Positions Response 56
5.10 List Reports API 58
5.10.1 Sample List Reports Request 58
5.10.2 Sample List Reports Response 59
5.11 List Position Accounts API 59
5.11.1 Sample List Position Accounts Request........59
5.11.2 Sample List Position Accounts Response 60
5.12 List Participants API 60
5.12.1 Sample List Participants Request........60
5.12.2 Sample List Participants Response........61
5.13 List Collateral Deposits API 61
5.13.1 Sample List Collateral Deposit Request 61
5.13.2 Sample List Collateral Deposit Response 62
5.14 Export Collateral Deposits API 63
5.14.1 Sample Export Collateral Request 63
5.14.2 Sample Export Collateral Response 63
5.15 List Operational Requests API........63
5.15.1 Sample List Operation Request 64
5.15.2 Sample List Operation Response 64
5.16 Get Operational Request Details API........65
5.16.1 Sample Get Operational Request Details Request 65
5.16.2 Sample Get Operational Request Details Response 65
5.17 List Collateral Balance API 66
5.17.1 Sample List Collateral Balance Request........66
5.17.2 Sample List Collateral Balance Response........67
5.18 List Notifications API........67
5.18.1 Sample List Notifications Request........67
5.18.2 Sample List Notifications Response........68
6. MUTATIONS........ 69
6.1 Download Reports API........69
6.1.1 Sample Download Reports Request 69
6.1.2 Sample Download Reports Response 70
6.2 Submit Historical Positions Request API 70
6.2.1 Sample Submit Historical Positions Request........70
6.2.2 Sample Submit Historical Positions Response........71
6.3 Submit Historical Trades Request API 71
6.3.1 Sample Submit Historical Trades Request........72
6.3.2 Sample Submit Historical Trades Response 72
7. SUBSCRIPTIONS........ 74
7.1 Positions Feed API........74
7.1.1 Sample Positions Feed Request........74
7.1.2 Sample Positions Feed Response........75
7.2 Historical Positions Request Feed API........75

7.2.1 Sample Historical Positions Feed Request 75
7.2.2 Sample Historical Positions Feed Response 76
7.3 Reports Feed API 76
7.3.1 Sample Reports Feed Request 76
7.3.2 Sample Reports Feed Response 76
7.4 Settlement Positions Feed API 77
7.4.1 Sample Settlement Positions Feed Request 77
7.4.2 Sample Settlement Positions Feed Response 77
7.5 Trades Feed API 78
7.5.1 Sample Trades Feed Request 78
7.5.2 Sample Trades Feed Response 78
7.6 Historical Trades Request Feed API 79
7.6.1 Sample Historical Trades Request 79
7.6.2 Sample Historical Trades Response 79
7.7 Collateral Deposits Feed API 80
7.7.1 Sample Collateral Deposits Feed Request 80
7.7.2 Sample Collateral Deposits Feed Response 80
7.8 Operations Feed API 80
7.8.1 Sample Operations Feed Request 81
7.8.2 Sample Operations Feed Response 81
7.9 Collateral Balance Feed API 81
7.9.1 Sample Collateral Balance Feed Request 81
7.9.2 Sample Collateral Balance Feed Response 82
7.10 Notifications Feed API 82
7.10.1 Sample Notification Feed Request 82
7.10.2 Sample Notification Feed Response 83
8. LIST OF ATTRIBUTES 84
9. STATIC VALUES 92
10. ERROR REGISTER.... 94

1. INTRODUCTION

Euronext is extending its competitive European offer to include clearing services, thus completing the value chain operated by the Euronext Group.

As announced on 9 November 2021, Euronext intends to make Euronext Clearing (formerly CC&G) the CCP of choice for the Euronext cash, listed derivatives and commodities markets. It will continue to offer an open access CCP model for cash equity clearing.

Euronext Clearing is therefore building a new system to offer clearing services to the European markets. The content of this document focuses on clearing for cash markets, and aims to provide a technical overview of the new clearing system, which will be released in 2023.

As described in the Preface, this version of the document contains additional details regarding the clearing data that members can query. See the *What's New* section in the Preface for a detailed list of the updates of this version with respect to the previous one.

The description of the commands available to perform operational requests will be provided in the next version of the document.

The Euronext Clearing APIs are described in the GraphQL schema *(graphql.schema)* in section 4. The *graphql.schema* will be further updated and completed with additional API definitions, including operational commands, in a future version of this document.

1.1 Glossary

This section provides a list of some terms and abbreviations commonly used in this document. Please note that some of these terms are described in more detail in the dedicated sections within this document, or in the associated Euronext Clearing Systems specifications.

- IdP: Identity Provider used by Euronext Clearing to verify the identity of the end users and machine users
- Client application: application that requests resources from one of the Euronext Clearing services
- Client Credentials: credentials released by the IdP to the client to perform the first phase of the machine-to-machine authentication. Each set of credentials is made up of a pair of Client IDs and a Secret
- GUI: Graphical User Interface, web interface for end users
- API: Application Programming Interface
- Technical User: privileged user in the client's organisation in charge of managing the Client Credentials
- JSON: JavaScript Object Notation, textual data format used for communications
- JWT: Json Web Token, token used to share security information between two parties

- SFTP: Secure File Transfer Protocol, protocol for file transfer with secured connections
- Access Token: token used to authenticate with the IdP
- TTL: Time To Live

1.2 Overview

The Euronext Clearing systems manage the entire clearing process starting with the collection of market data and ending with the settlement phase.

Clearing Members can interact with the Euronext Clearing system to access clearing data, perform actions on trades and positions, and manage collateral, as well as interacting with the risk management systems to monitor margins and perform simulations. These actions can be carried out through several communication channels.

While providing a common information set, each channel addresses specific use cases and should therefore be deemed complementary to the others.

The following channels are available:

- **Graphical User Interface (GUI) channel:** displays the user’s real-time clearing data on a web browser. It also provides features that enable the Clearing Member to interact with the settlement and collateral management workflows. Additionally, it allows the user to interact with the risk management system for margin calculation and simulations on portfolios. Detailed information on the GUI will be provided in a dedicated User Guide.
- **Application Programming Interface (API) channel**: enables the interoperability of the clearing system with the Clearing Member’s own systems. It is based on a machine-to-machine protocol and provides all the informative and operational functions that are made available for human users via the GUI.
- **Secured File Transfer Protocol (SFTP) channel:** allows Clearing Members to retrieve reports generated by the Clearing System. For further details see *“Euronext Clearing Interfaces Overview and Reporting”,* section 3.3.
- **FIX connection:** provides real-time trade confirmation. For further details see *“Euronext Clearing Interfaces Overview and Reporting”,* section 1.3.2.

2. API ACCESS

This section explains the authentication method and authorisation procedure that a Client Application must perform to interact with Euronext Clearing APIs.

2.1 API Manager for Client Credentials

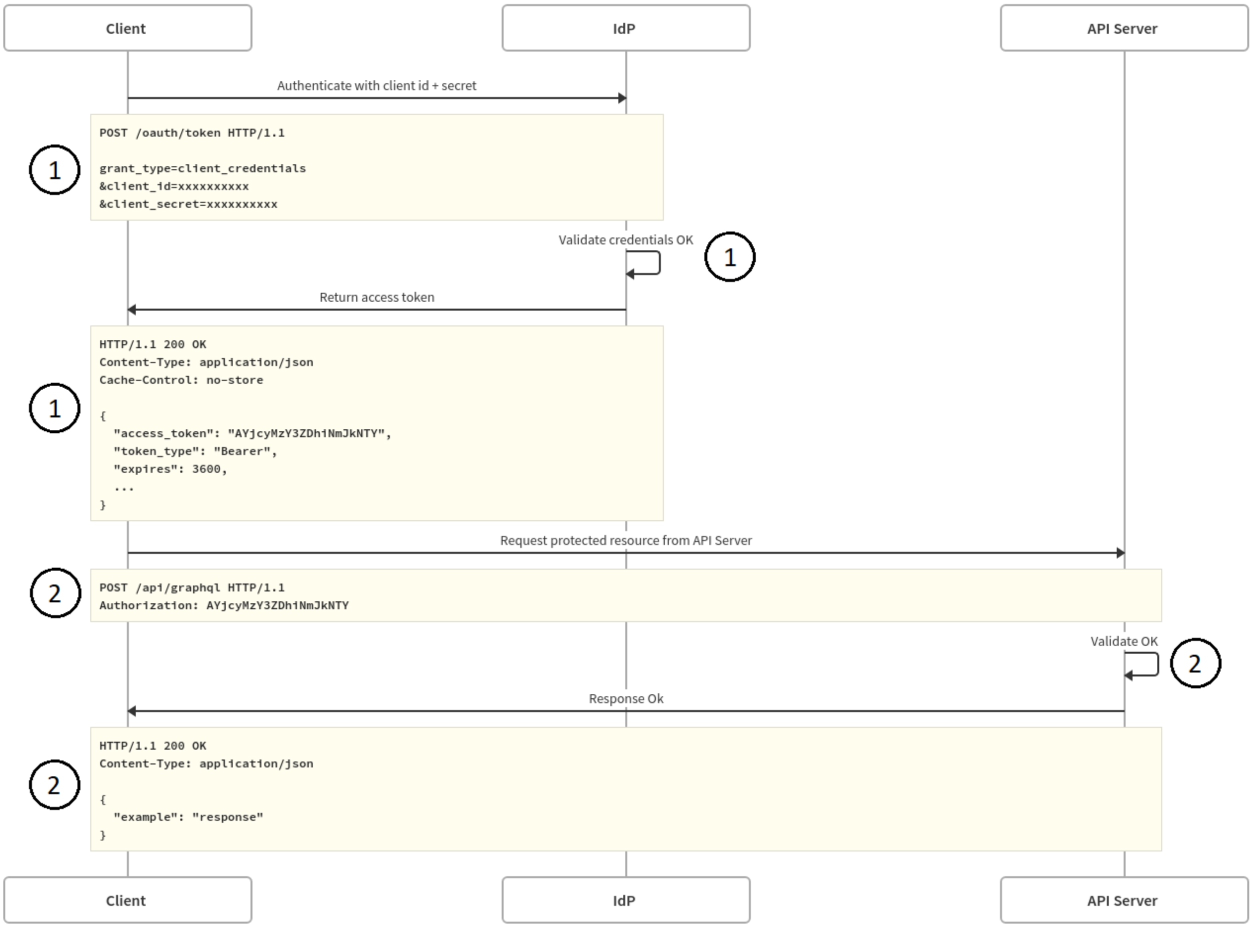
In order to obtain a token for machine-to-machine access, Clearing Members must generate private credentials. The Identity Provider (IdP) needs these private credentials to identify the client's application and return a valid token. More details on the token can be found in the section 2.22.2.

The Client Credentials must be generated by the Client from the dedicated Client Credentials User Interface, to which technical users from the client's organisation have access. Client Credentials are composed of a pair of Client IDs and Secret. It is the responsibility of the technical users at the Client firm to manage the credentials on the client side.

At the time of generation, the system will return the values of the client credentials, but will not store them internally. The client's technical users must store the credentials in a safe place and integrate them in the client application.

The following diagram shows how Clearing Members interact with Euronext Clearing Systems:

Authentication workflow: Client credential grants



```
POST /oauth/token HTTP/1.1

grant_type=client_credentials
&client_id=xxxxxxxxxx
&client_secret=xxxxxxxxxx
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "access_token": "AYjcyMzY3ZDhiNmJkNTY",
  "token_type": "Bearer",
  "expires": 3600,
  ...
}
```

```
POST /api/graphql HTTP/1.1
Authorization: AYjcyMzY3ZDhiNmJkNTY
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "example": "response"
}
```

1. The client application contacts the dedicated URL of the IdP to log in using the Client Credentials. The credentials are validated by the IdP and if the authentication is successful, the IdP returns a JSON Web Token (JWT) with duration of 30 minutes. Please note that when the token expires, the client application must authenticate again with the IdP to obtain a new valid token (see section 2.4 for more details).
2. The JWT must be sent with each API call in the *Authorization* header and it will be validated by the API Server. If the validation is successful, the API Server will execute the API request and return the response to the client application.

Client Credentials only need to be generated once; however, they can be renewed when necessary.

Credentials can be revoked from the Client Credentials Interface by technical users, meaning that the IdP will no longer generate a JWT when the client application presents these credentials.

The Euronext Clearing Operations team can also manage clients' credentials via a dedicated management interface. Actions that Euronext Clearing Operations can carry out include:

- Deactivation/activation: this action suspends the validity of the credentials temporarily;
- Revocation: this action deletes the credentials permanently. The IdP will no longer generate a JWT with these credentials.

API Client Credentials are segregated per environment (EUA vs Prod).

2.2 Token Structure

The authentication of end users and machine users is based on the OpenID Connect Protocol.

In line with the protocol, Euronext Clearing uses the Access Token to allow the client application to access a resource. The token is issued by the authorisation server and is encoded as a JSON Web Token (JWT).

The JWT is composed of three parts, separated by a dot:

1. **Header**: specifies the type of the token and the algorithm that is used
2. **Payload**: the payload contains the claims. There is a set of registered claims, for example: iss (issuer), exp (expiration time), sub (subject), and aud (audience). The payload can also include extra attributes that define custom claims, such as employee role.
3. **Signature**: to create the signature part, the encoded header and encoded payload are signed using the signature algorithm from the header. The signature is used to verify that the token has not been corrupted along the way.

The claims are used to represent an identity and its associations. Euronext Clearing has customised the JWT in order to include claims that specify permissions and the Member Code associated with the user/machine.

Below is an example of a decoded JWT that could be proposed by Euronext Clearing:

```
{
 "ver": 1,
 "jti": "AT.Oxmd9AABJgivAypz9KjIVBL1GqIEGCzSSXL9qztAvmI",
 "iss": "https://euronextclearing.com/oauth2/aus3j5mpv9p0lpqx6417",
 "aud": "eu-core-h2m-audience",
 "iat": 1657197886,
 "exp": 1657198486,
 "cid": "0oa3j5zmmk03Q1B2p417",
 "uid": "00u3lp7ytdMe8pWge417",
 "scp": [
  "openid",
  "profile"
 ],
 "auth_time": 1657197886,
 "sub": "name.surname@company.com",
 "perms": [
  "auth.margindeltaposition.fetch",
  "auth.margindeltaposition.actions",
  "auth.clientcredentials.fetch",
  "auth.clientcredentials.actions",
  "auth.defaultfundlog.fetch",
  "auth.collateraleligibleinstruments.fetch",
  "auth.collateral.fetch",
  "auth.collateraloperations.cashrestitutionrequest",
  "auth.collateraloperations.securityrestitutionrequest",
  "auth.collateraloperations.sharerestitutionrequest",
  "auth.collateraloperations.uploadrequest",
  "auth.defaultfund.actions",
  "auth.defaultfund.fetch",
  "auth.instruments.fetch",
  "auth.marginamounts.fetch",
  "auth.marginmonitor.actions",
  "auth.marginmonitor.fetch",
  "auth.operations.fetch",
  "auth.participants.fetch",
  "auth.positions.actions",
  "auth.positions.fetch",
  "auth.positionslog.actions",
  "auth.positionslog.fetch",
  "auth.reporting.actions",
  "auth.reporting.fetch",
  "auth.settlementpositions.fetch",
  "auth.simulationengine.actions",
  "auth.simulationengine.fetch",
  "auth.trades.actions",
  "auth.trades.fetch",
  "auth.tradeslog.actions",
  "auth.tradeslog.fetch",
  "auth.virtualportfolio.actions",
  "auth.virtualportfolio.fetch"
 ],
 "mbr": "01030"
}
```

In addition to the above, the security of information in transit will be guaranteed by TLS encryption. Further details will be provided in a dedicated document.

2.3 Token Generation

To log in, the user should call the token generation Service offered by the IdP (the URL for the token generation will be communicated in a dedicated Connectivity document). The IdP exposes a REST Endpoint. To perform the log-in, use the following curl command:

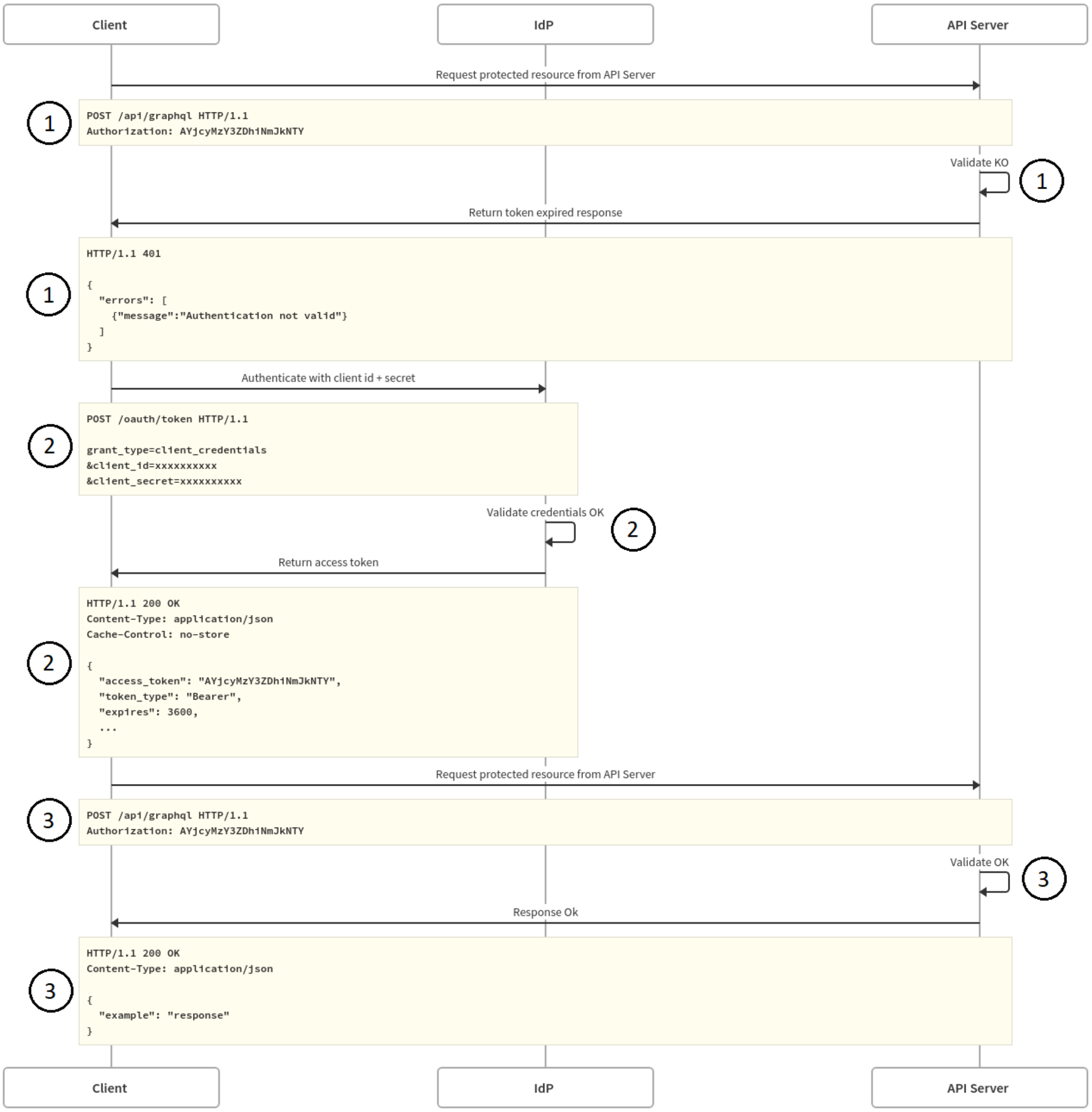
```
curl --location --request POST
'https://<IDP_BASE_URL>/<AUTH_SERVER_ID>/v1/token' \
--header 'Authorization: Basic <CLIENT_ID_CLIENT_SECRET_ENCODED>' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'scope=<AUTH_SERVER_SCOPE>' \
--data-urlencode 'grant_type=client_credentials'
```

- <IDP_BASE_URL>: Base URL of the Service for the token generation;
- <AUTH_SERVER_ID>: authorisation server that manages Authorisation for the API Server. This ID is specific to the environment that the client application is trying to reach, i.e. it will differ between EUA and PROD. The values will be provided in a dedicated Connectivity document;
- <CLIENT_ID_CLIENT_SECRET_ENCODED>: pair of Client ID and Secret encoded base64; and
- <AUTH_SERVER_SCOPE>: parameter that specifies what access privileges are being requested as part of the authorisation. The value will be provided in a dedicated Connectivity document.

2.4 Token Expiration

The diagram below shows how Clearing Members generate a new Access Token when their existing token expires:

Request protected resource with expired access token


Client
IdP
API Server
Request protected resource from API Server
1
Validate KO
1
Return token expired response
1
Authenticate with client id + secret
2
Validate credentials OK
2
Return access token
2
Request protected resource from API Server
3
Validate OK
3
Response Ok
3
Client
IdP
API Server

```
POST /api/graphql HTTP/1.1
Authorization: AYjcyMzY3ZDhiNmJkNTY
```

```
HTTP/1.1 401

{
  "errors": [
    {"message":"Authentication not valid"}
  ]
}
```

```
POST /oauth/token HTTP/1.1

grant_type=client_credentials
&client_id=xxxxxxxxxx
&client_secret=xxxxxxxxxx
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "access_token": "AYjcyMzY3ZDhiNmJkNTY",
  "token_type": "Bearer",
  "expires": 3600,
  ...
}
```

```
POST /api/graphql HTTP/1.1
Authorization: AYjcyMzY3ZDhiNmJkNTY
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "example": "response"
}
```

1. The client application tries to access the API Server using an expired Access Token. The API Server checks the Access Token and does not accept it. The Server responds to the client application with an Unauthorised code.
2. The client application submits a request to the IdP to obtain a new Access Token by providing the required pair of client IDs and a Secret as detailed in section 2.3. The IdP validates the credentials and returns a new Access Token to the requester.
3. The client application performs a new request providing the new Access Token to the API Server.

2.5 API Authorisation

After the authentication process, the system performs an authorisation check to verify whether the user is allowed to perform the requested command.

All requests that have a valid JWT are analysed by the API Server. Before executing the API request, the Server retrieves the information contained in the claims (for more details regarding the JWT structure please refer to section 2.2.

The JWT contains information regarding the membership of the client; this information is used to return only data that belongs to the Clearing Member specified in the JWT. The membership is contained in the *mbr* claim of the JWT.

The API Server checks if the client application has the required permissions needed for the requested operation. If not, the request is rejected. Permissions granted can be Read and/or Write and must be set for each data object (trade, instrument, etc…). The permissions are contained in the *perms* claim of the JWT.

3. API USAGE

This section explains how to interact with Euronext Clearing APIs.

The APIs are publicly exposed on the Internet and can be reached at a defined URL. The APIs are developed using the GraphQL Query Language. GraphQL is a query language for APIs and a runtime for fulfilling queries on the data source.

A GraphQL API has a single entry point instead of multiple resource addresses. Through the same API connection, Clearing Members will be able to retrieve reports, query the warehouse database and perform operational actions.

GraphQL allows the client application to specify the data required; the Server will return exactly the data requested by the client application. This prevents under-fetching or over-fetching. Data is returned in JavaScript Object Notation (JSON), the textual data format used for communications.

GraphQL protocol is widely used and related libraries are freely available for the most common programming language.

3.1 Interaction via Curl

To interact with a GraphQL query the client application can use a curl command like the following:

```
curl --location --request POST '<server_endpoint>/graphql' \
--header 'Authorization: eyJraWQiOiJLSnc5X1hDUXRsbERua...' \
--header 'Content-Type: application/json' \
--data-raw '{"query":"query <operation>($format: String!)
{exportParticipants(format:
$format)}","variables":{"format":"xlsx"},"operationName":"<operation>"}'
```

- `<server_endpoint>`: must be replaced with the actual endpoint of the GraphQL Server. This information will be shared in a dedicated Connectivity document.
- `--header 'Authorization: ...'`: must contain the JWT Token returned after the authentication process using the Participant’s Client Credentials.
- `--data-raw`: contains the GraphQL operation requested.
- `<operation>`: label for the operation requested. The value must be chosen by the Client Application as preferred.
- `variables`: contains the inputs populated by the Client Application. The input parameters vary based on the API.

3.2 Formatting a query

The samples provided in the following sections contain new lines to improve their readability.

When performing the request via CURL, the new lines must be removed and replaced with a space according to the GraphQL standard.

4. GRAPHQL SCHEMA

This section contains the definition of available data and operations that can be performed by the client applications. The definitions are contained in the *graphql.schema* below.
The file provides definitions regarding Instruments, Trades, Positions, Settlement Positions, Reports as well as their related queries/ subscriptions/mutations.

```
directive @constraint (
  # String constraints
  minLength: Int
  maxLength: Int
  startsWith: String
  endsWith: String
  contains: String
  notContains: String
  pattern: String
  format: String

  # Number constraints
  min: Float
  max: Float
  exclusiveMin: Float
  exclusiveMax: Float
  multipleOf: Float
  uniqueTypeName: String
) on INPUT_FIELD_DEFINITION | FIELD_DEFINITION

directive @rateLimit (
  # Number of occurrences allowed over duration
  limit: Int!

  # Number of seconds before limit is reset
  duration: Int!
) on OBJECT | FIELD_DEFINITION

enum AccountCols {
  gcm
  mbr
  pos_acct_id
}

enum CollateralBalanceCols {
  cash_available
  cash_buffer
  cash_excess
  cash_required
  cash_used
  clearing_curncy
  coll_acct_id
  collateral_call
  collateral_evaluation_tmsp
  country_limit_excess
  gcm
  margin_requirement
  margin_requirement_tmsp
  ncb_used
  securities_after_limits
  securities_available
  securities_used
}

enum CollateralDepositCols {
  accr_int
  balance_hct
  balance_mtm
  clearing_curncy
  coll_acct_id
  collateral_curncy
  collateral_subtype
  created_at_tmsp
  curncy_hct
  curr_exch_rate
  deposit_id
  gcm
  isin
  isin_hct
  maturity_dt
  modified_at_tmsp
  price
  qty
  wwr_amt
}

enum CollateralEligibleCols {
  collateral_curncy
  curncy_hct
  isin
  maturity_dt
  tradable_amt
}

enum InstrumentCols {
  adjustment_factor
  asset_type
  cfi_code
  closing_price
  corporate_event
  country
  end_valid_days
  index_name
  instr_group_code
  instr_subtype
  instr_type
  isin
  listing_dt
  maturity_dt
  mic
  settl_curcy
  settle_delay
  strike_curncy
  strike_price
  symbol_index
  trade_curncy
}

enum OperationalRequestCols {
  created_at_tmsp
  disposal_id
  err_code
  gcm
  mbr
  modified_at_tmsp
  user_id
}

enum ParticipantCols {
  participant_code
}

enum PositionCols {
  accr_int
  adjustment_factor
  amt
  asset_type
  buy_in_dt
  cash_settl_dt
  created_at_tmsp
  end_valid_dt
  gcm
  haircut
  isin
  margin_acct_id
  mbr
  miti
  modified_at_tmsp
  mtm_amt
  mtm_price
  mtm_tmsp
  pos_acct_id
  position_id
  previous_settl_ref
  qty
  settl_curcy
  settl_dt
  settle_ref
  symbol_index
  trade_dt
  unsettled_amt
  unsettled_qty
}

enum ReportCols {
  added_tmsp
  agent
  expiration_dt
  gcm
  mbr
  report_code
  report_format
  report_name
  report_tmsp
  report_version
  restore_request_tmsp
}

enum SettlementPositionCols {
  agent
  amt
  bic_party_2
  bic_party_3
  buy_in_dt
  cash_settl_dt
  ccp_bic_code
  ccp_sec_acct
  corporate_action_fraction
  corporate_event
  corporate_msg_ref
  created_at_tmsp
  csd_bic_ccp
  csd_bic_cm
  csd_settle_acct
  delivery_acct_id
  effective_settl_dt
  end_valid_dt
  fail_acct
  gcm
  isin
  market_venue
  miti
  modified_at_tmsp
  previous_settl_ref
  qty
  reason_code
  settl_curcy
  settl_dt
  settle_ref
  symbol_index
  trade_dt
  transformation_fraction
  unsettled_amt
  unsettled_qty
}

enum SortDirection {
  ASC
  DESC
}

enum TradeCols {
  accr_int
  client_order_id
  counterparty_code
  ctv
  curr_exch_rate
  exec_id
  gcm
  isin
  market_price
  mbr
  mic
  order_id
  pos_acct_id
  position_id
  qty
  settle_amt
  settl_curcy
  settl_dt
  settle_per
  settle_ref
  symbol_index
  trade_curncy
  trade_dt
  trade_tm
}

input AccountFilterModel {
  acct: ModelAcctInputSet
  gcm: ModelStringInput
  mbr: ModelStringInput
  pos_acct_id: ModelStringInput
}

input CollateralBalanceColSort {
  column: CollateralBalanceCols!
  order: SortDirection!
}

input CollateralBalanceFilterModel {
  cash_available: ModelFloatInput
  cash_buffer: ModelFloatInput
  cash_excess: ModelFloatInput
  cash_required: ModelFloatInput
  cash_used: ModelFloatInput
  clearing_curncy: ModelStringInput
  coll_acct_id: ModelStringInput
  collateral_call: ModelFloatInput
  collateral_evaluation_tmsp: ModelDateTimeInput
  country_limit_excess: ModelFloatInput
  gcm: ModelStringInput
  margin_requirement: ModelFloatInput
  margin_requirement_tmsp: ModelDateTimeInput
  ncb_used: ModelFloatInput
  payment_flag: ModelBoolInputSet
  securities_after_limits: ModelFloatInput
  securities_available: ModelFloatInput
  securities_used: ModelFloatInput
}

input CollateralDepositColSort {
  column: CollateralDepositCols!
  order: SortDirection!
}
```

```
input CollateralDepositFilterModel {
  accr_int: ModelFloatInput
  balance_hct: ModelFloatInput
  balance_mtm: ModelFloatInput
  clearing_curncy: ModelStringInput
  coll_acct_id: ModelStringInput
  collateral_curncy: ModelStringInput
  collateral_subtype: ModelStringInput
  collateral_type: ModelCollateralTypeInputSet
  created_at_tmsp: ModelDateTimeInput
  curncy_hct: ModelFloatInput
  curr_exch_rate: ModelFloatInput
  deposit_id: ModelIntInput
  excluded_flag: ModelIntInput
  gcm: ModelStringInput
  isin: ModelStringInput
  isin_hct: ModelFloatInput
  main_depository: ModelMainDepositoryInputSet
  maturity_dt: ModelDateInput
  modified_at_tmsp: ModelDateTimeInput
  price: ModelFloatInput
  qty: ModelFloatInput
  qty_type: ModelQtyTypeInputSet
}

input CollateralEligibleColSort {
  column: CollateralEligibleCols!
  order: SortDirection!
}

input CollateralEligibleFilterModel {
  collateral_curncy: ModelStringInput
  collateral_type: ModelCollateralTypeInputSet
  curncy_hct: ModelFloatInput
  isin: ModelStringInput
  maturity_dt: ModelDateInput
  qty_type: ModelQtyTypeInputSet
  tradable_amt: ModelFloatInput
}

input InstrumentFilterModel {
  asset_type: ModelStringInput
  closing_price: ModelFloatInput
  corporate_event: ModelStringInput
  country: ModelStringInput
  coupon_freq: ModelIntInput
  end_valid_days: ModelIntInput
  guaranteed_flag: ModelIntInput
  index_name: ModelStringInput
  instr_group_code: ModelStringInput
  instr_status: ModelIntInput
  instr_subtype: ModelStringInput
  instr_type: ModelStringInput
  instr_unit: ModelIntInput
  isin: ModelStringInput
  listing_dt: ModelStringInput
  main_depository: ModelMainDepositoryInputSet
  maturity_dt: ModelDateInput
  mic: ModelStringInput
  qty_type: ModelQtyTypeInputSet
  settl_curcy: ModelStringInput
  settle_delay: ModelIntInput
  settle_system: ModelIntInput
  strike_curncy: ModelStringInput
  strike_price: ModelFloatInput
  symbol_index: ModelIntInput
  trade_curncy: ModelStringInput
}

input NotificationFilterModel {
  mbr: ModelStringInput
  notification_audience: ModelNotificationAudienceInputSet
  notification_category: ModelStringInput
  notification_level: ModelNotificationLevelInputSet
  notification_persistence: ModelNotificationPersistenceInputSet
  notification_status: ModelStringInput
  notification_title: ModelStringInput
  notification_tmsp: ModelDateTimeInput
  notification_type: ModelNotificationTypeInputSet
}

input OperationalRequestColSort {
  column: OperationalRequestCols!
  order: SortDirection!
}

input OperationalRequestFilterModel {
  created_at_tmsp: ModelDateTimeInput
  err_code: ModelStringInput
  gcm: ModelStringInput
  mbr: ModelStringInput
  modified_at_tmsp: ModelDateTimeInput
  operation: ModelOperationInputSet
  status: ModelStatusInputSet
  user_id: ModelStringInput
}

input ParticipantColSort {
  column: ParticipantCols!
  order: SortDirection!
}

input ParticipantFilterModel {
  participant_code: ModelStringInput
  segment: ModelSegmentInputSet
}

input PositionColSort {
  column: PositionCols!
  order: SortDirection!
}

input PositionFilterModel {
  accr_int: ModelFloatInput
  acct: ModelAcctInputSet
  amt: ModelFloatInput
  asset_type: ModelStringInput
  buy_in_dt: ModelDateInput
  buy_in_status: ModelBoolInputSet
  cash_settl_dt: ModelDateInput
  cash_settl_status: ModelBoolInputSet
  created_at_tmsp: ModelDateTimeInput
  end_valid_dt: ModelDateInput
  french_registered_flag: ModelBoolInputSet
  gcm: ModelStringInput
  isin: ModelStringInput
  main_depository: ModelMainDepositoryInputSet
  margin_acct_id: ModelStringInput
  mbr: ModelStringInput
  miti: ModelStringInput
  modified_at_tmsp: ModelDateTimeInput
  mtm_price: ModelFloatInput
  pos_acct_id: ModelStringInput
  position_id: ModelIntInput
  position_source: ModelSourceInputSet
  position_status: ModelPosStatusInputSet
  position_type: ModelPosTypeInputSet
  previous_settl_ref: ModelStringInput
  qty: ModelFloatInput
  qty_type: ModelQtyTypeInputSet
  settl_curcy: ModelStringInput
  settl_dt: ModelDateInput
  settle_ref: ModelStringInput
  settle_system: ModelIntInput
  side: ModelSideInputSet
  symbol_index: ModelIntInput
  trade_dt: ModelDateInput
  unsettled_amt: ModelFloatInput
  unsettled_qty: ModelFloatInput
}

input ReportColSort {
  column: ReportCols!
  order: SortDirection!
}

input ReportFilterModel {
  added_tmsp: ModelDateTimeInput
  agent: ModelStringInput
  expiration_dt: ModelDateInput
  gcm: ModelStringInput
  mbr: ModelStringInput
  report_code: ModelStringInput
  report_format: ModelStringInput
  report_name: ModelStringInput
  report_status: ModelReportStatusInputSet
  report_tmsp: ModelDateTimeInput
  report_version: ModelIntInput
  restore_request_tmsp: ModelDateTimeInput
}

input SettlementPositionColSort {
  column: SettlementPositionCols!
  order: SortDirection!
}

input SettlementPositionFilterModel {
  agent: ModelStringInput
  amt: ModelFloatInput
  buy_in_dt: ModelDateInput
  cash_settl_dt: ModelDateInput
  ccp_bic_code: ModelStringInput
  ccp_sec_acct: ModelStringInput
  corporate_event: ModelStringInput
  corporate_msg_ref: ModelStringInput
  created_at_tmsp: ModelDateTimeInput
  csd_settle_acct: ModelStringInput
  delivery_acct_id: ModelStringInput
  effective_settl_dt: ModelDateInput
  end_valid_dt: ModelDateInput
  fail_acct: ModelStringInput
  french_registered_flag: ModelBoolInputSet
  gcm: ModelStringInput
  hold_indicator: ModelBoolInputSet
  isin: ModelStringInput
  main_depository: ModelMainDepositoryInputSet
  market_venue: ModelStringInput
  matching_status: ModelMatchingStatusInputSet
  miti: ModelStringInput
  modified_at_tmsp: ModelDateTimeInput
  netting_rule: ModelNettingTypeInputSet
  partial_settl_status: ModelPartialSettleStatusInputSet
  previous_settl_ref: ModelStringInput
  qty: ModelFloatInput
  qty_type: ModelQtyTypeInputSet
  settl_curcy: ModelStringInput
  settl_dt: ModelDateInput
  settle_ref: ModelStringInput
  settle_source: ModelSettlementSourceInputSet
  settle_status: ModelSettlementStatusInputSet
  settle_system: ModelIntInput
  side: ModelSideInputSet
  symbol_index: ModelIntInput
  trade_dt: ModelIntInput
  unsettled_amt: ModelFloatInput
  unsettled_qty: ModelFloatInput
}

input TradeFilterModel {
  accr_int: ModelFloatInput
  acct: ModelAcctInputSet
  client_order_id: ModelIntInput
  counterparty_code: ModelStringInput
  ctv: ModelFloatInput
  curr_exch_rate: ModelFloatInput
  exec_id: ModelStringInput
  execution_type: ModelIntInput
  french_registered_flag: ModelBoolInputSet
  gcm: ModelStringInput
  guaranteed_flag: ModelIntInput
  isin: ModelStringInput
  main_depository: ModelMainDepositoryInputSet
  market_price: ModelFloatInput
  mbr: ModelStringInput
  mic: ModelStringInput
  order_id: ModelIntInput
  pos_acct_id: ModelStringInput
  position_id: ModelIntInput
  qty: ModelFloatInput
  qty_type: ModelQtyTypeInputSet
  settle_amt: ModelFloatInput
  settl_curcy: ModelStringInput
  settl_dt: ModelDateInput
  settle_per: ModelIntInput
  settle_ref: ModelStringInput
  settle_system: ModelIntInput
  side: ModelSideInputSet
  symbol_index: ModelIntInput
  trade_curncy: ModelStringInput
  trade_dt: ModelDateInput
  trade_tm: ModelTimeInput
}

input AccountColSort {
  column: AccountCols!
  order: SortDirection!
}

input InstrumentColSort {
  column: InstrumentCols!
  order: SortDirection!
}

input DateRange {
  from: String! @constraint(pattern: "^[0-9]*$", format: "date", minLength: 8,
maxLength: 8)
  to: String! @constraint(pattern: "^[0-9]*$", format: "date", minLength: 8,
maxLength: 8)
}

input SectionLabel {
  label: String! @constraint(pattern: "^(DESKS)|(SPA)|(TEMPL)$")
}

input DownloadReportInput {
  id: String!
  formats: [String!]!
}

input ModelDateInput {
  eq: String @constraint(pattern: "^[0-9]{4}-[0-9]{2}-[0-9]{2}$", minLength: 10,
maxLength: 10)
  le: String @constraint(pattern: "^[0-9]{4}-[0-9]{2}-[0-9]{2}$", minLength: 10,
maxLength: 10)
  ge: String @constraint(pattern: "^[0-9]{4}-[0-9]{2}-[0-9]{2}$", minLength: 10,
maxLength: 10)
  between: [String] @constraint(pattern: "^[0-9]{4}-[0-9]{2}-[0-9]{2}$", minLength:
10, maxLength: 10)
}

input ModelTimeInput {
  eq: String @constraint(pattern: "^[0-9]{2}:[0-9]{2}:[0-9]{2}$", minLength: 8,
maxLength: 8)
  le: String @constraint(pattern: "^[0-9]{2}:[0-9]{2}:[0-9]{2}$", minLength: 8,
maxLength: 8)
  ge: String @constraint(pattern: "^[0-9]{2}:[0-9]{2}:[0-9]{2}$", minLength: 8,
maxLength: 8)
  between: [String] @constraint(pattern: "^[0-9]{2}:[0-9]{2}:[0-9]{2}$", minLength:
8, maxLength: 8)
}

input ModelDateTimeInput {
  eq: String @constraint(pattern: "^[0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]{2}:[0-
9]{2}:[0-9]{2}$", minLength: 19, maxLength: 19)
  le: String @constraint(pattern: "^[0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]{2}:[0-
9]{2}:[0-9]{2}$", minLength: 19, maxLength: 19)
  ge: String @constraint(pattern: "^[0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]{2}:[0-
9]{2}:[0-9]{2}$", minLength: 19, maxLength: 19)
  between: [String] @constraint(pattern: "^[0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]{2}:[0-
9]{2}:[0-9]{2}$", minLength: 19, maxLength: 19)
}

input ModelFloatInput {
  eq: Float
  ne: Float
  lt: Float
  le: Float
  gt: Float
  ge: Float
  between: [Float]
}

input ModelIntInput {
  eq: Int
  ne: Int
  lt: Int
  le: Int
  gt: Int
  ge: Int
  between: [Int]
}

input ModelStringInput {
  contains: String @constraint(minLength: 1)
  notContains: String @constraint(minLength: 1)
  eq: String @constraint(minLength: 1)
  ne: String @constraint(minLength: 1)
  beginsWith: String @constraint(minLength: 1)
  endsWith: String @constraint(minLength: 1)
}

input ModelAcctInputSet {
  in: [String!] @constraint(pattern: "^(C)|(H)|(L)$")
}

input ModelCollateralTypeInputSet {
  in: [String!] @constraint(pattern: "^(C)|(B)|(S)$")
}

input ModelBoolInputSet {
  in: [String!] @constraint(pattern: "^(Y)|(N)$")
}

input ModelMainDepositoryInputSet {
  in: [String!] @constraint(pattern:
"^(00001)|(00002)|(00003)|(00004)|(00006)|(00010)$")
}

input ModelMatchingStatusInputSet {
  in: [String!] @constraint(pattern: "^(AM)|(TBM)$")
}

input ModelNettingTypeInputSet {
  in: [String!] @constraint(pattern: "^(AGGR)|(SING)$")
}

input ModelNotificationAudienceInputSet {
  in: [String!] @constraint(pattern: "^(A)|(P)|(U)$")
}

input ModelNotificationLevelInputSet {
  in: [String!] @constraint(pattern: "^(E)|(I)|(W)$")
}
```

```
input ModelNotificationPersistenceInputSet {
  in: [String!] @constraint(pattern: "^(E)|(P)$")
}

input ModelNotificationTypeInputSet {
  in: [String!] @constraint(pattern: "^(C)|(S)$")
}

input ModelOperationInputSet {
  in: [String!] @constraint(pattern: "^(D)|(S)|(W)$")
}

input ModelPartialSettleStatusInputSet {
  in: [String!] @constraint(pattern: "^(PARC)|(PAIN)$")
}

input ModelPosStatusInputSet {
  in: [String!] @constraint(pattern: "^(LIVE)|(CLSD)$")
}

input ModelPosTypeInputSet {
  in: [String!] @constraint(pattern: "^(S)|(F)$")
}

input ModelQtyTypeInputSet {
  in: [String!] @constraint(pattern: "^(F)|(U)$")
}

input ModelReportStatusInputSet {
  in: [String!] @constraint(pattern: "^(A)|(R)|(E)|(N)$")
}

input ModelSegmentInputSet {
  in: [String] @constraint(pattern: "^(Euronext Equity Markets)$")
}

input ModelSettlementSourceInputSet {
  in: [String!] @constraint(pattern: "^(T)|(CA)|(BP)|(PO)|(BI)|(OR)$")
}

input ModelSettlementStatusInputSet {
  in: [String!] @constraint(pattern: "^(CAND)|(REJT)|(FULL)|(PEND)|(PENF)$")
}

input ModelSideInputSet {
  in: [String!] @constraint(pattern: "^(B)|(S)$")
}
```

```
input ModelSourceInputSet {
  in: [String!] @constraint(pattern: "^(ST)|(CA)|(BP)$")
}

input ModelStatusInputSet {
  in: [String!] @constraint(pattern: "^(E)|(L)|(H)|(P)|(R)|(V)|(X)$")
}

input PaginationModel {
  offset: Int!
  limit: Int!
}

input TradeColSort {
  column: TradeCols!
  order: SortDirection!
}

type Account {
  acct: String
  crud: String
  gcm: String
  id: String
  mbr: String
  msg_sequence: Int
  participant_description: String
  pos_acct_description: String
  pos_acct_id: String
}

type ArgsCash {
  amt: Float
  collateral_curncy: String
  crud: String
  deposit_id: Int
  disposal_id: String
  msg_sequence: Int
  request_amt: Float
  ssi_id: String
  transfer_src_acct: String
  transfer_tgt_acct: String
}

type CollateralBalance {
  cash_available: Float
  cash_buffer: Float
  cash_excess: Float
  cash_required: Float
  cash_used: Float
  clearing_curncy: String
  coll_acct_id: String
  collateral_call: Float
  collateral_evaluation_tmsp: String
  country_limit_excess: Float
  crud: String
  gcm: String
  id: String
  margin_requirement: Float
  margin_requirement_tmsp: String
  msg_sequence: Int
  ncb_used: Float
  payment_flag: String
  securities_after_limits: Float
  securities_available: Float
  securities_used: Float
}

type CollateralDeposit {
  accr_int: Float
  balance_hct: Float
  balance_mtm: Float
  clearing_curncy: String
  coll_acct_id: String
  collateral_curncy: String
  collateral_subtype: String
  collateral_type: String
  created_at_tmsp: String
  crud: String
  curncy_hct: Float
  curr_exch_rate: Float
  deposit_id: Int
  excluded_flag: Int
  gcm: String
  id: String
  isin: String
  isin_hct: Float
  main_depository: String
  maturity_dt: String
  modified_at_tmsp: String
  msg_sequence: Int
  price: Float
  qty: Float
  qty_type: String
  wwr_amt: Float
}

type CollateralEligible {
  collateral_curncy: String
  collateral_description: String
  collateral_type: String
  crud: String
  curncy_hct: Float
  id: String
  isin: String
  maturity_dt: String
  msg_sequence: Int
  qty_type: String
  tradable_amt: Float
}

type HistoricalRequest {
  created_at_tmsp: String
  crud: String
  date_from: String
  date_to: String
  disposal_id: String
  err_code: String
  err_desc: String
  expiry_tmsp: String
  file_url: String
  gcm: String
  mbr: String
  modified_at_tmsp: String
  msg_sequence: Int
  size: String
  status: String
  user_id: String
}

type Instrument {
  adjustment_factor: Float
  asset_type: String
  cfi_code: String
  closing_price: Float
  corporate_event: String
  country: String
  coupon_freq: Int
  crud: String
  end_valid_days: Int
  guaranteed_flag: Int
  id: String
  index_name: String
  instr_desc: String
  instr_group_code: String
  instr_status: Int
  instr_subtype: String
  instr_type: String
  instr_unit: Int
  isin: String
  legal_form: Int
  liquid_indicator: String
  listing_dt: String
  main_depository: String
  maturity_dt: String
  mic: String
  msg_sequence: Int
  qty_type: String
  settl_curcy: String
  settle_delay: Int
  settle_system: Int
  strike_curncy: String
  strike_price: Float
  symbol_index: Int
  trade_curncy: String
}

type Notification {
  crud: String
  mbr: String
  msg_sequence: Int
  notification_audience: String
  notification_category: String
  notification_id: String
  notification_level: String
  notification_persistence: String
  notification_status: String
  notification_text: String
  notification_title: String
  notification_tmsp: String
  notification_type: String
  user_id: String
}

type OperationalRequest {
  additional_info: String
  args_cash: ArgsCash
  created_at_tmsp: String
  crud: String
  disposal_id: String
  err_code: String
  err_desc: String
  gcm: String
  mbr: String
  modified_at_tmsp: String
  msg_sequence: Int
  operation: String
  status: String
  user_id: String
}

type Participant {
  crud: String
  id: String
  msg_sequence: Int
  participant_code: String
  participant_description: String
  segment: String
}

type Position {
  accr_int: Float
  acct: String
  adjustment_factor: Float
  amt: Float
  asset_type: String
  buy_in_dt: String
  buy_in_status: String
  cash_settl_dt: String
  cash_settl_status: String
  created_at_tmsp: String
  crud: String
  end_valid_dt: String
  french_registered_flag: String
  gcm: String
  haircut: Float
  id: String
  isin: String
  main_depository: String
  margin_acct_id: String
  mbr: String
  miti: String
  modified_at_tmsp: String
  msg_sequence: Int
  mtm_amt: Float
  mtm_price: Float
  mtm_tmsp: String
  pos_acct_id: String
  position_id: Int
  position_source: String
  position_status: String
  position_type: String
  previous_settl_ref: String
  qty: Float
  qty_type: String
  settl_curcy: String
  settl_dt: String
  settle_ref: String
  settle_system: Int
  side: String
  symbol_index: Int
  trade_dt: String
  unsettled_amt: Float
  unsettled_qty: Float
}

type Report {
  added_tmsp: String
  agent: String
  crud: String
  expiration_dt: String
  gcm: String
  id: String
  mbr: String
  msg_sequence: Int
  report_code: String
  report_description: String
  report_format: String
  report_id: String
  report_name: String
  report_status: String
  report_tmsp: String
  report_version: Int
  restore_request_tmsp: String
}

type Settings {
  user_id: String
  section: String
  data: String
  upd_tmsp: String
}
```

```
type SettlementPosition {
  agent: String
  amt: Float
  bic_party_2: String
  bic_party_3: String
  buy_in_dt: String
  cash_settl_dt: String
  ccp_bic_code: String
  ccp_sec_acct: String
  corporate_action_fraction: Float
  corporate_event: String
  corporate_msg_ref: String
  created_at_tmsp: String
  crud: String
  csd_bic_ccp: String
  csd_bic_cm: String
  csd_settle_acct: String
  delivery_acct_id: String
  delivery_position_id: Int
  effective_settl_dt: String
  end_valid_dt: String
  fail_acct: String
  french_registered_flag: String
  gcm: String
  hold_indicator: String
  id: String
  isin: String
  main_depository: String
  market_venue: String
  matching_status: String
  miti: String
  modified_at_tmsp: String
  msg_sequence: Int
  netting_rule: String
  partial_settl_status: String
  previous_settl_ref: String
  qty: Float
  qty_type: String
  reason_code: String
  reason_descr: String
  settl_curcy: String
  settl_dt: String
  settle_ref: String
  settle_source: String
  settle_status: String
  settle_system: Int
  side: String
  symbol_index: Int
  trade_dt: String
  transformation_fraction: Float
  unsettled_amt: Float
  unsettled_qty: Float
}

type Trade {
  accr_int: Float
  acct: String
  client_order_id: Int
  counterparty_code: String
  crud: String
  ctv: Float
  curr_exch_rate: Float
  exec_id: String
  execution_type: Int
  french_registered_flag: String
  gcm: String
  guaranteed_flag: Int
  id: String
  isin: String
  main_depository: String
  market_price: Float
  mbr: String
  mic: String
  msg_sequence: Int
  order_id: Int
  pos_acct_id: String
  position_id: Int
  qty: Float
  qty_type: String
  settle_amt: Float
  settl_curcy: String
  settl_dt: String
  settle_per: Int
  settle_ref: String
  settle_system: Int
  side: String
  symbol_index: Int
  text: String
  trade_capacity: Int
  trade_curncy: String
  trade_dt: String
  trade_tm: String
}
```

```
type Query @rateLimit(limit: 1000, duration: 1) {
  exportCollateralBalances(filterModel: CollateralBalanceFilterModel, sortModel:
[CollateralBalanceColSort!], format: String!, separator: String): String
  exportCollateralDeposits(filterModel: CollateralDepositFilterModel, sortModel:
[CollateralDepositColSort!], format: String!, separator: String): String
  exportCollateralEligibles(filterModel: CollateralEligibleFilterModel, sortModel:
[CollateralEligibleColSort!], format: String!, separator: String): String
  exportInstruments(filterModel: InstrumentFilterModel, sortModel:
[InstrumentColSort!], format: String!, separator: String): String
  exportParticipants(filterModel: ParticipantFilterModel, sortModel:
[ParticipantColSort!], format: String!, separator: String): String
  exportPositions(filterModel: PositionFilterModel, sortModel: [PositionColSort!],
format: String!, separator: String): String
  exportReports(filterModel: ReportFilterModel, sortModel: [ReportColSort!],
format: String!, separator: String): String
  exportSettlementPositions(filterModel: SettlementPositionFilterModel, sortModel:
[SettlementPositionColSort!], format: String!, separator: String): String
  exportTrades(filterModel: TradeFilterModel, sortModel: [TradeColSort!], format:
String!, separator: String): String
  getOperationalRequestDetails(disposal_id: String): OperationalRequest
  listAccounts(filterModel: AccountFilterModel, sortModel: [AccountColSort!],
paginationModel: PaginationModel): [Account]
  listCollateralBalances(filterModel: CollateralBalanceFilterModel, sortModel:
[CollateralBalanceColSort!], paginationModel: PaginationModel): [CollateralBalance]
  listCollateralDeposits(filterModel: CollateralDepositFilterModel, sortModel:
[CollateralDepositColSort!], paginationModel: PaginationModel): [CollateralDeposit]
  listCollateralEligibles(filterModel: CollateralEligibleFilterModel, sortModel:
[CollateralEligibleColSort!], paginationModel: PaginationModel):
[CollateralEligible]
  listInstruments(filterModel: InstrumentFilterModel, sortModel:
[InstrumentColSort!], paginationModel: PaginationModel): [Instrument]
  listNotifications(filterModel: NotificationFilterModel, paginationModel:
PaginationModel): [Notification]
  listOperationalRequests(filterModel: OperationalRequestFilterModel,
paginationModel: PaginationModel): [OperationalRequest]
  listParticipants(filterModel: ParticipantFilterModel, sortModel:
[ParticipantColSort!], paginationModel: PaginationModel): [Participant]
  listPositions(filterModel: PositionFilterModel, sortModel: [PositionColSort!],
paginationModel: PaginationModel): [Position]
  listPositionsHistoricalRequests: [HistoricalRequest]
  listReports(filterModel: ReportFilterModel, sortModel: [ReportColSort!],
paginationModel: PaginationModel): [Report]
  listSettlementPositions(filterModel: SettlementPositionFilterModel, sortModel:
[SettlementPositionColSort!], paginationModel: PaginationModel):
[SettlementPosition]
```

```
  listTrades(filterModel: TradeFilterModel, sortModel: [TradeColSort!],
paginationModel: PaginationModel): [Trade]
  listTradesHistoricalRequests: [HistoricalRequest]
}

type Mutation @rateLimit(limit: 1000, duration: 1) {
  downloadReports(reports: [DownloadReportInput!]!): String
  submitPositionsHistoricalRequest(input: DateRange!): Boolean
  submitTradesHistoricalRequest(input: DateRange!): Boolean
}

type Subscription @rateLimit(limit: 1000, duration: 1) {
  onCollateralBalancesFeed: CollateralBalance
  onCollateralDepositsFeed: CollateralDeposit
  onNotificationsFeed: Notification
  onOperationalRequestsFeed: OperationalRequest
  onPositionsFeed: Position
  onPositionsHistoricalRequestsFeed: HistoricalRequest
  onReportsFeed: Report
  onSettlementPositionsFeed: SettlementPosition
  onTradesFeed: Trade
  onTradesHistoricalRequestsFeed: HistoricalRequest
}

schema {
  query: Query
  mutation: Mutation
  subscription: Subscription
}
```

5. QUERIES

This section focuses on the queries that can be executed via API. The sections below describe each API and provide request and response examples.

All the APIs that execute queries allow the user to filter data according to different fields and to sort the results based on several criteria.

Besides data retrieval, queries allow data export of relevant data types. The output of the export operation are files containing filtered/sorted clearing data.

5.1 List Instruments API

The List Instruments API returns the listed instruments and their attributes.

5.1.1 Sample List Instruments Request

```
query Query {
  listInstruments {
    adjustment_factor
    asset_type
    cfi_code
    closing_price
    corporate_event
    country
    coupon_freq
    crud
    end_valid_days
    guaranteed_flag
    index_name
    instr_desc
    instr_group_code
    instr_status
    instr_subtype
    instr_type
    instr_unit
    isin
    legal_form
    liquid_indicator
    listing_dt
    main_depository
    maturity_dt
    mic
    msg_sequence
    qty_type
    settl_curcy
    settle_delay
    settle_system
    strike_curncy
    strike_price
    symbol_index
    trade_curncy
  }
}
```

5.1.2 Sample List Instruments Response

```
{
    "data": {
      "listInstruments": [
        {
          "adjustment_factor": 0.000,
          "asset_type": "E",
          "cfi_code": "ESVUFX",
          "closing_price": 1400000000000.000000,
          "corporate_event": "00",
          "country": null,
          "coupon_freq": 0,
          "crud": "U",
          "end_valid_days": 1,
          "guaranteed_flag": 1,
          "index_name": null,
          "instr_desc": "Euronext Paris 02",
          "instr_group_code": "S9",
          "instr_status": 1,
          "instr_subtype": "ORDINARY SHARES",
          "instr_type": "SHARE",
          "instr_unit": 1,
          "isin": "ENXC00000012",
          "legal_form": 0,
          "liquid_indicator": "0",
          "listing_dt": "2015-12-10",
          "main_depository": "00001",
          "maturity_dt": null,
          "mic": "XMLI",
          "msg_sequence": 999,
          "qty_type": "U",
          "settl_curcy": "EUR",
          "settle_delay": 2,
          "settle_system": 60,
          "strike_curncy": null,
          "strike_price": 0,
          "symbol_index": 1722876,
          "trade_curncy": "EUR"
        }
      ]
    }
  }
```

5.2 Export Instruments API

The Export Instruments API allows the client to export the listed instruments and their attributes that are stored in the database, in a CSV/XML/XLSX file.

The client application must provide a value for the *format* field (i.e. the field is mandatory). The format value is a String, so if the client needs different formats it needs to send different export requests to the Clearing House.

The separator field is optional; if the client does not provide a value the API server will use the ";" as default value. The only allowed value as separator for the csv format is ";".

The API returns a temporary link (TTL of 2 minutes) that the client can use to download the file. It is possible to apply filters and sorting criteria.

5.2.1 Sample Export Instruments Request

```
query Query($format: String!, $separator: String, $filterModel: 
InstrumentFilterModel) {
  exportInstruments(format: $format, separator: $separator, filterModel: 
$filterModel)
}

variables:
{
  "format": "csv",
  "separator": ";",
  "filterModel": {
    "isin": {
      "contains": "FR"
    }
  }
}
```

5.2.2 Sample Export Instruments Response

```
{
  "data": {
    "exportInstruments": "https://<URL-TO-DOWNLOAD-DOCUMENT> "
  }
}
```

5.3 List Trades API

The List Trades API allows client applications to retrieve the list of Trades and their description.

5.3.1 Sample List Trades Request

```
query Query {
    listTrades {
      accr_int
      acct
      client_order_id
      counterparty_code
      crud
      ctv
      curr_exch_rate
      exec_id
      execution_type
      french_registered_flag
      gcm
      guaranteed_flag
      isin
      main_depository
      market_price
      mbr
      mic
      msg_sequence
      order_id
      pos_acct_id
      position_id
      qty
      qty_type
      settle_amt
      settl_curcy
      settl_dt
      settle_per
      settle_ref
      settle_system
      side
      symbol_index
      text
      trade_capacity
      trade_curncy
      trade_dt
      trade_tm
    }
  }
```

5.3.2 Sample List Trades Response

```
{
  "data": {
   "listTrades": [
    {
     "accr_int": 0,
     "acct": "H",
     "client_order_id": 56234558,
     "counterparty_code": null,
     "crud": "I",
     "ctv": 120000000,
     "curr_exch_rate": 1,
     "exec_id": "27193807",
     "execution_type": 1,
     "french_registered_flag": "N",
     "gcm": "1000",
     "guaranteed_flag": 0,
     "isin": "ENXTCALXP043",
     "main_depository": "00006",
     "market_price": 12000,
     "mbr": "01030",
     "mic": "ALXP",
     "msg_sequence": 366,
     "order_id": 16861875,
     "pos_acct_id": "A-004-PAC002",
     "position_id": 221222003965,
     "qty": 10000,
     "qty_type": "U",
     "settle_amt": 120000000,
     "settl_curcy": "USD",
     "settl_dt": "2022-12-27",
     "settle_per": 2,
     "settle_ref": "EX221222AAAC5N01",
     "settle_system": 51,
     "side": "B",
     "symbol_index": 1725382,
     "text": "ABCDEFGHIJKLMNOPQRSTUVWXYZ",
     "trade_capacity": 695,
     "trade_curncy": "USD",
     "trade_dt": "2022-12-22",
     "trade_tm": "17:39:53"
    }
   ]
  }
 }
```

5.4 Export Trades API

The Export Trades API allows the client to export the Trades and their attributes that are stored in the database, in a CSV/XML/XLSX file.

The client application must provide a value for the *format* field (i.e. the field is mandatory). The format value is a String, so if the client needs different formats it needs to send different export requests to the Clearing House.

The separator field is optional; if the client does not provide a value the API server will use the ";" as default value. The only value permitted as separator for the csv format is ";".

The API returns a temporary link (TTL of 2 minutes) that the client can use to download the file. It is possible to apply filters and sorting criteria.

5.4.1 Sample Export Trades Request

```
query Query($format: String!, $separator: String) {
  exportTrades(format: $format, separator: $separator)
}
variables:
{
  "format": "csv"
}
```

5.4.2 Sample Export Trades Response

```
{
  "data": {
    "exportTrades": "https://< URL-TO-DOWNLOAD-DOCUMENT > "
  }
}
```

5.5 List Trades Historical Requests API

The List Trades Historical Requests API allows client applications to retrieve the restore operational request executed on Trades sent by the Client.

5.5.1 Sample List Trades Historical Request

```
query ListTradesHistoricalRequests{
    listTradesHistoricalRequests {
      created_at_tmsp
      crud
      date_from
      date_to
      disposal_id
      err_code
      err_desc
      expiry_tmsp
      file_url
      gcm
      mbr
      modified_at_tmsp
      msg_sequence
      size
      status
      user_id
    }
  }
```

5.5.2 Sample List Trades Historical Response

```
{
    "data": {
      "listTradesHistoricalRequests": [
        {
          "created_at_tmsp": "2022-12-22 14:18:25",
          "crud": "I",
          "date_from": "2022-04-16",
          "date_to": "2022-11-21",
          "disposal_id": "319901b0-e004-47ef-9be1-4c988cc431f7",
          "err_code": null,
          "err_desc": null,
          "expiry_tmsp": null,
          "file_url": null,
          "gcm": "1030",
          "mbr": "1030",
          "modified_at_tmsp": "2022-12-22 14:18:25",
          "msg_sequence": 0,
          "size": null,
          "status": "L",
          "user_id": "email@organization.com"
        }
      ]
    }
  }
```

5.6 List Positions API

The List Positions API allows client applications to retrieve the Positions managed by the Client.

5.6.1 Sample List Positions Request

```
query Query {
  listPositions {
    accr_int
    acct
    adjustment_factor
    amt
    asset_type
    buy_in_dt
    buy_in_status
    cash_settl_dt
    cash_settl_status
    created_at_tmsp
    crud
    end_valid_dt
    french_registered_flag
    gcm
    haircut
    isin
    main_depository
    margin_acct_id
    mbr
    miti
    modified_at_tmsp
    msg_sequence
    mtm_amt
    mtm_price
    mtm_tmsp
    pos_acct_id
    position_id
    position_source
    position_status
    position_type
    previous_settl_ref
    qty
    qty_type
    settl_curcy
    settl_dt
    settle_ref
    settle_system
    side
    symbol_index
    trade_dt
    unsettled_amt
    unsettled_qty
  }
}
```

5.6.2 Sample List Positions Response

```
  {
    "data": {
      "listPositions": [
        {
          "accr_int": 0,
          "acct": "C",
          "adjustment_factor": 0,
          "amt": 148500000000,
          "asset_type": "E",
          "buy_in_dt": null,
          "buy_in_status": "N",
          "cash_settl_dt": null,
          "cash_settl_status": "N",
          "created_at_tmsp": "2022-12-22 10:12:56",
          "crud": "U",
          "end_valid_dt": "2022-12-29",
          "french_registered_flag": "N",
          "gcm": "0001",
          "haircut": 0,
          "isin": "ENXTCXAMS003",
          "main_depository": "00003",
          "margin_acct_id": "A-001-MAC",
          "mbr": "01030",
          "miti": null,
          "modified_at_tmsp": null,
          "msg_sequence": 999,
          "mtm_amt": 0,
          "mtm_price": 0,
          "mtm_tmsp": null,
          "pos_acct_id": "A-001-PAC01",
          "position_id": 221222003965,
          "position_source": "ST",
          "position_status": "LIVE",
          "position_type": "S",
          "previous_settl_ref": null,
          "qty": 1000000,
          "qty_type": "U",
          "settl_curcy": "EUR",
          "settl_dt": "2022-12-27",
          "settle_ref": "EX221222AAAAZN01",
          "settle_system": 60,
          "side": "B",
          "symbol_index": 1723872,
          "trade_dt": "2022-12-22",
          "unsettled_amt": -148500000000,
          "unsettled_qty": 1000000
        },
        {
          "accr_int": 0,
          "acct": "H",
          "adjustment_factor": 0,
          "amt": 148500000000,
          "asset_type": "E",
          "buy_in_dt": null,
          "buy_in_status": "N",
          "cash_settl_dt": null,
          "cash_settl_status": "N",
          "created_at_tmsp": "2022-12-22 10:12:56",
          "crud": "U",
          "end_valid_dt": "2022-12-29",
          "french_registered_flag": "N",
          "gcm": "0005",
          "haircut": 0,
          "isin": "ENXTCXAMS003",
          "main_depository": "00003",
          "margin_acct_id": "B-MACLIEN",
          "mbr": "01030",
          "miti": null,
          "modified_at_tmsp": null,
          "msg_sequence": 999,
          "mtm_amt": 0,
          "mtm_price": 0,
          "mtm_tmsp": null,
          "pos_acct_id": "B-003-PAH01",
          "position_id": 221222003966,
          "position_source": "ST",
          "position_status": "LIVE",
          "position_type": "S",
          "previous_settl_ref": null,
          "qty": -1000000,
          "qty_type": "U",
          "settl_curcy": "EUR",
          "settl_dt": "2022-12-27",
          "settle_ref": "EX221222AAAA0N01",
          "settle_system": 60,
          "side": "S",
          "symbol_index": 1723872,
          "trade_dt": "2022-12-22",
          "unsettled_amt": 148500000000,
          "unsettled_qty": -1000000
        }
    ]
}
}
```

5.7 Export Positions API

The Export Positions API allows the client to export the Positions that are stored in the database, in a CSV/XML/XLSX file.

The client application must provide a value for the *format* field (i.e. the field is mandatory). The format value is a String, so if the client needs different formats it needs to send different export requests to the Clearing House.

The separator field is optional; if the client does not provide a value the API server will use ";" as the default value. The only value permitted as separator for the csv format is ";".

The API returns a temporary link (TTL of 2 minutes) that the client can use to download the file. It is possible to apply filters and sorting criteria.

5.7.1 Sample Export Positions Request

```
query Query($format: String!, $separator: String) {
  exportPositions(format: $format, separator: $separator)
}
variables:
{
  "format": "xml"
}
```

5.7.2 Sample Export Positions Response

```
{
  "data": {
    "exportPositions": "https://< URL-TO-DOWNLOAD-DOCUMENT > "
  }
}
```

5.8 List Position Historical Requests API

The List Positions Historical Requests API allows client applications to retrieve the restore operational request executed on Positions sent by the Client.

5.8.1 Sample List Position Historical Request

```
query ListPositionsHistoricalRequests {
    listPositionsHistoricalRequests {
      created_at_tmsp
      crud
      date_from
      date_to
      disposal_id
      err_code
      err_desc
      expiry_tmsp
      file_url
      gcm
      mbr
      modified_at_tmsp
      msg_sequence
      size
      status
      user_id
    }
  }
```

5.8.2 Sample List Historical Positions Response

```
{
    "data": {
      "listPositionsHistoricalRequests": [
        {
          "created_at_tmsp": "2022-12-22 14:18:25",
          "crud": "I",
          "date_from": "2022-04-16",
          "date_to": "2022-11-21",
          "disposal_id": "319901b0-e004-47ef-9be1-4c988cc431f7",
          "err_code": null,
          "err_desc": null,
          "expiry_tmsp": null,
          "file_url": null,
          "gcm": "1030",
          "mbr": "1030",
          "modified_at_tmsp": "2022-12-22 14:18:25",
          "msg_sequence": 0,
          "size": null,
          "status": "L",
          "user_id": "email@organization.com"
        }
      ]
    }
  }
```

5.9 List Settlement Positions API

The List Settlement Positions API allows client applications to retrieve the Settlement Positions managed by the Client.

5.9.1 Sample List Settlement Positions Request

```
query Query {
  listSettlementPositions {
    agent
    amt
    bic_party_2
    bic_party_3
    buy_in_dt
    cash_settl_dt
    ccp_bic_code
    ccp_sec_acct
    corporate_action_fraction
    corporate_event
    corporate_msg_ref
    created_at_tmsp
    crud
    csd_bic_ccp
    csd_bic_cm
    csd_settle_acct
    delivery_acct_id
    delivery_position_id
    effective_settl_dt
    end_valid_dt
    fail_acct
    french_registered_flag
    gcm
    hold_indicator
    isin
    main_depository
    market_venue
    matching_status
    miti
    modified_at_tmsp
    msg_sequence
    netting_rule
    partial_settl_status
    previous_settl_ref
    qty
    qty_type
    reason_code
    reason_descr
    settl_curcy
    settl_dt
    settle_ref
    settle_source
    settle_status
    settle_system
    side
    symbol_index
    trade_dt
    transformation_fraction
    unsettled_amt
    unsettled_qty
  }
}
```

5.9.2 Sample List Settlement Positions Response

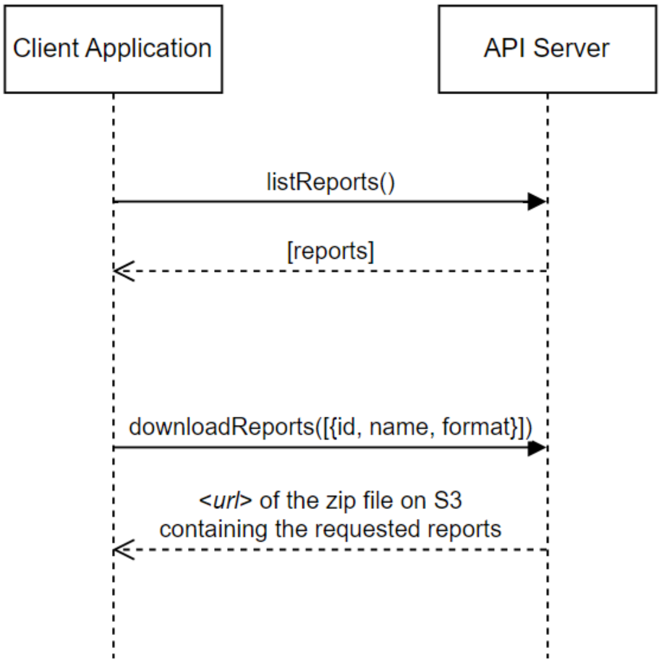
```
{
    "data": {
      "listSettlementPositions": [
        {
          "agent": null,
          "amt": 1000000,
          "bic_party_2": null,
          "bic_party_3": null,
          "buy_in_dt": null,
          "cash_settl_dt": null,
          "ccp_bic_code": "CCEGITRR001",
          "ccp_sec_acct": "13300",
          "corporate_action_fraction": 0,
          "corporate_event": null,
          "corporate_msg_ref": null,
          "created_at_tmsp": "2022-12-22 16:17:05",
          "crud": "U",
          "csd_bic_ccp": "MOTIITMMXXX",
          "csd_bic_cm": "NECINL2AXXX",
          "csd_settle_acct": "SECUACC30",
          "delivery_acct_id": "B-H-DA300",
          "delivery_position_id": 2917,
          "effective_settl_dt": null,
          "end_valid_dt": "2022-12-29",
          "fail_acct": "1",
          "french_registered_flag": "N",
          "gcm": "1030",
          "hold_indicator": "N",
          "isin": "ENXTCXAMS003",
          "main_depository": "00003",
          "market_venue": "VARI",
          "matching_status": "AM",
          "miti": "VLOPM394F3R18J6S9KH9",
          "modified_at_tmsp": "2022-12-22 16:17:05",
          "msg_sequence": 939,
          "netting_rule": "SING",
          "partial_settl_status": "PAIN",
          "previous_settl_ref": null,
          "qty": 1000000,
          "qty_type": "U",
          "reason_code": null,
          "reason_descr": "ABCDEFGHIJKLMNOPQRS",
          "settl_curcy": "EUR",
          "settl_dt": "2022-12-27",
          "settle_ref": "EX221222AAAC5N01",
          "settle_source": "T",
          "settle_status": "PEND",
          "settle_system": 60,
          "side": "B",
          "symbol_index": 579,
          "trade_dt": "2022-12-22",
          "transformation_fraction": 0,
          "unsettled_amt": null,
          "unsettled_qty": null
        }
      ]
    }
  }
```

5.10 List Reports API

The List Reports API allows client applications to query the list of available reports that are available to Clients.

The API returns a list of attributes for each available report. The information returned can be used in a second step to download the reports, as explained in section 6.1.

The diagram below shows the flow that the client applications must implement to download reports:


Client Application
API Server
listReports()
[reports]
downloadReports([{id, name, format}])
<url> of the zip file on S3
containing the requested reports

5.10.1 Sample List Reports Request

```
query Query {
  listReports {
    added_tmsp
    agent
    crud
    expiration_dt
    gcm
    mbr
    msg_sequence
    report_code
    report_description
    report_format
    report_id
    report_name
    report_status
    report_tmsp
    report_version
    restore_request_tmsp
  }
}
```

5.10.2 Sample List Reports Response

```
{
  "data": {
    "listReports": [
      {
        "added_tmsp": "20221008T095003Z",
        "agent": "1000",
        "crud": "I",
        "expiration_dt": 20221230,
        "gcm": "1000",
        "mbr": "1000",
        "msg_sequence": 9185,
        "report_code": "DP01",
        "report_description": "Report description",
        "report_format":  "csv,xml,xlsx",
        "report_id": 1723066454,
        "report_name": "20221003-DP01-1111-1",
        "report_status": "A",
        "report_tmsp": "20221008T095003Z",
        "report_version": 1,
        "restore_request_tmsp": "20221008T095003Z"
      }
    ]
  }
}
```

5.11 List Position Accounts API

The List Position Accounts API allows client applications to retrieve the sponsorship links and the Position Accounts owned by the Client.

5.11.1 Sample List Position Accounts Request

```
query ListAccounts {
      listAccounts {
        acct
        crud
        gcm
```

```
    mbr
    participant_description
    pos_acct_description
    pos_acct_id
   }
  }
```

5.11.2 Sample List Position Accounts Response

```
{
  "data": {
   "listAccounts": [
    {
     "acct": "H",
     "crud": "I",
     "gcm": "1030",
     "mbr": "1030",
     "participant_description": "Participant Description",
     "pos_acct_description": "HOUSE POS ACCT DESCR",
     "pos_acct_id": "1234567890123456789Y"
    },
    {
     "acct": "C",
     "crud": "I",
     "gcm": "1030",
     "mbr": "1000",
     "participant_description": "Participant’s Client Description",
     "pos_acct_description": "CLIENT POS ACCT DESCR",
     "pos_acct_id": "8394228947628849YO393"
    }
   ]
  }
 }
```

5.12 List Participants API

The List Participants API allows client applications to retrieve the list of Participants managed by the Clearing House.

5.12.1 Sample List Participants Request

```
query ListParticipants {
 listParticipants {
  participant_code
  participant_description
```

```
    segment
  }
}
```

5.12.2 Sample List Participants Response

```
{
    "data": {
      "listParticipants": [
        {
          "participant_code": "1030",
          "participant_descr": "PARTICIPANT_DESCR",
          "segment": "Euronext Equity Markets"
        }
      ]
    }
  }
```

5.13 List Collateral Deposits API

The List Collateral Deposits API allows client applications to retrieve the deposited collateral on all Collateral Accounts owned by the Client.

5.13.1 Sample List Collateral Deposit Request

```
query CollateralDeposit {
    listCollateralDeposits {
        accr_int
        balance_hct
        balance_mtm
        clearing_curncy
        coll_acct_id
        collateral_curncy
        collateral_subtype
        collateral_type
        created_at_tmsp
        crud
        curncy_hct
        curr_exch_rate
        deposit_id
        excluded_flag
        gcm
        isin
        isin_hct
        main_depository
        maturity_dt
        modified_at_tmsp
        msg_sequence
        price
        qty
        qty_type
        wwr_amt
  }
}
```

5.13.2 Sample List Collateral Deposit Response

```
{
    "data": {
      "listCollateralDeposits": [
       {
        "accr_int": 0,
        "balance_hct": 0,
        "balance_mtm": 0,
        "clearing_curncy": "EUR",
        "coll_acct_id": "A-007-CAC006",
        "collateral_curncy": "EUR",
        "collateral_subtype": null,
        "collateral_type": "C",
        "created_at_tmsp": "2022-12-22 10:12:56",
        "crud": "I",
        "curncy_hct": 1.00,
        "curr_exch_rate": 1.00,
        "deposit_id": 8274364829,
        "excluded_flag": 0,
        "gcm": "1000",
        "isin": "ENXTCALXP043",
        "isin_hct": 1.00,
        "main_depository": "00006",
        "maturity_dt": "2023-16-22",
        "modified_at_tmsp": null,
        "msg_sequence": 98,
        "price": 12000,
        "qty": 1000,
        "qty_type": "U",
        "wwr_amt": 50
       }
     ]
   }
}
```

5.14 Export Collateral Deposits API

The Export Collateral Deposits API allows the client applications to export in a CSV/XML/XLSX file the Deposited Collateral on Collateral Accounts owned by the Client.

The client application must provide a value for the *format* field (i.e. the field is mandatory). The format value is a String, so if the client needs different formats, it needs to send different export requests to the Clearing House.

The separator field is optional; if the client does not provide a value the API server will use ";" as the default value. The only value permitted as separator for the csv format is ";".

The API returns a temporary link (TTL of 2 minutes) that the client can use to download the file. It is possible to apply filters and sorting criteria.

5.14.1 Sample Export Collateral Request

```
query Query($format: String!, $separator: String) {
  exportCollateralDeposits(format: $format, separator: $separator)
}
variables:
{
  "format": "xlsx"
}
```

5.14.2 Sample Export Collateral Response

```
{
    "data": {
      "exportCollateralDeposits": "https://< URL-TO-DOWNLOAD-DOCUMENT > "
    }
  }
```

5.15 List Operational Requests API

The List Operational Requests API allows client applications to retrieve the list of operational requests performed by the Client. This specific API will return the main characteristics that provide an overview of the different requests.

To obtain the details of a request, the client application can perform an API call to the Get Operational Request Details API (see section 5.16) providing the request identifier; the API will return the same fields as the List Operational Request and, in addition to this, the details of the operation.

5.15.1 Sample List Operation Request

```
query ListOperationalRequests{
    listOperationalRequests {
      additional_info
      args_cash
      created_at_tmsp
      crud
      disposal_id
      err_code
      err_desc
      gcm
      mbr
      modified_at_tmsp
      msg_sequence
      operation
      status
      user_id
    }
  }
```

5.15.2 Sample List Operation Response

```
{
    "data": {
      "listOperationalRequests": [
        {
          "additional_info": "Additional information sent by the Client",
          "args_cash": null,
          "created_at_tmsp": "2022-12-22 14:18:25",
          "crud": "I",
          "disposal_id": "319901b0-e004-47ef-9be1-4c988cc431f7",
          "err_code": null,
          "err_desc": null,
          "gcm": "1030",
          "mbr": "1030",
          "modified_at_tmsp": "2022-12-22 14:18:25",
          "msg_sequence": 0,
          "operation": "D",
          "status": "L",
          "user_id": "email@organization.com"
        }
      ]
    }
  }
```

5.16 Get Operational Request Details API

The Get Operational Requests API allows client applications to retrieve the main characteristics and the details of specific operational requests performed by the Client.

The client application must provide, as an input to the request, the request identifier (i.e. *disposal_id*) that can be obtained via List Operational Requests API (see section 5.15).

5.16.1 Sample Get Operational Request Details Request

```
query ListOperationalRequests{
    listOperationalRequests {
      additional_info
      args_cash
      created_at_tmsp
      crud
      disposal_id
      err_code
      err_desc
      gcm
      mbr
      modified_at_tmsp
      msg_sequence
      operation
      status
      user_id
    }
  }
```

5.16.2 Sample Get Operational Request Details Response

```
{
    "data": {
      "listOperationalRequests": [
        {
          "additional_info": "Additional information sent by the Client",
          "args_cash": {
                 "amt": null,
                 "collateral_curncy": "EUR",
                 "crud": "I",
                 "deposit_id": 928,
                 "disposal_id": "319901b0-e004-47ef-9be1-4c988cc431f7",
                 "disposal_args_id": "829394749fddjd329201nfd",
                 "request_amt": 1000,
                 "ssi_id": "ssi-0929833",
                 "transfer_src_acct": null,
                 "transfer_tgt_acct": null
                 },
          "created_at_tmsp": "2022-12-22 14:18:25",
          "crud": "I",
          "disposal_id": "319901b0-e004-47ef-9be1-4c988cc431f7",
          "err_code": null,
          "err_desc": null,
          "gcm": "1030",
          "mbr": "1030",
          "modified_at_tmsp": "2022-12-22 14:18:25",
          "msg_sequence": 0,
          "operation": "D",
          "status": "L",
          "user_id": "email@organization.com"
        }
      ]
    }
  }
```

5.17 List Collateral Balance API

The List Collateral Balance API allows client applications to retrieve the variables considered during the collateral evaluation performed by the Clearing House, and the results of the collateral evaluation. The API also returns the margin exposure that the deposited collateral should cover and the cash call required, if any, to cover the liabilities.

5.17.1 Sample List Collateral Balance Request

```
query ListCollateralBalances {
  listCollateralBalances {
    cash_available
    cash_buffer
    cash_excess
    cash_required
    cash_used
    clearing_curncy
    coll_acct_id
    collateral_call
    collateral_evaluation_tmsp
    country_limit_excess
    gcm
    margin_requirement
    margin_requirement_tmsp
    ncb_used
    payment_flag
    securities_after_limits
    securities_available
    securities_used
  }
}
```

5.17.2 Sample List Collateral Balance Response

```
{
  "data": {
    "listOperationalRequests": [
     {
      "cash_available": 50000,
      "cash_buffer": 10000,
      "cash_excess": 0,
      "cash_required": 32000,
      "cash_used": 50000,
      "clearing_curncy": "EUR",
      "coll_acct_id": "A-007-CAC006",
      "collateral_call": 72000,
      "collateral_evaluation_tmsp": "2022-12-22 14:20:07",
      "country_limit_excess": 0,
      "gcm": "1000",
      "margin_requirement":  160000.00,
      "margin_requirement_tmsp": "2022-12-22 14:18:25",
      "ncb_used": 48000,
      "payment_flag": "Y",
      "securities_after_limits": null,
      "securities_available": null,
      "securities_used": 0
    }
  ]
}
}
```

5.18 List Notifications API

The List Notifications API allows client applications to retrieve the different type of notifications generated by the Clearing House.

5.18.1 Sample List Notifications Request

```
query ListNotifications {
  listNotifications {
    mbr
    msg_sequence
    notification_audience
    notification_category
    notification_id
    notification_level
    notification_persistence
    notification_status
    notification_text
    notification_title
    notification_tmsp
    notification_type
    user_id
  }
}
```

5.18.2 Sample List Notifications Response

```
{
    "data": {
      "listNotifications": [
        {
          "mbr": "1030",
          "msg_sequence": 900,
          "notification_audience": "U",
          "notification_category": "A",
          "notification_id": "NOTIFICATIONID123456",
          "notification_level": "I",
          "notification_persistence": "P",
          "notification_status": "N",
          "notification_text": "This is a system info notification for participant 1030.",
          "notification_title": "System INFO",
          "notification_tmsp": "2022-11-22 10:30:00",
          "notification_type": "C",
          "user_id": "user@organization.com"
        }
      ]
    }
  }
```

6. MUTATIONS

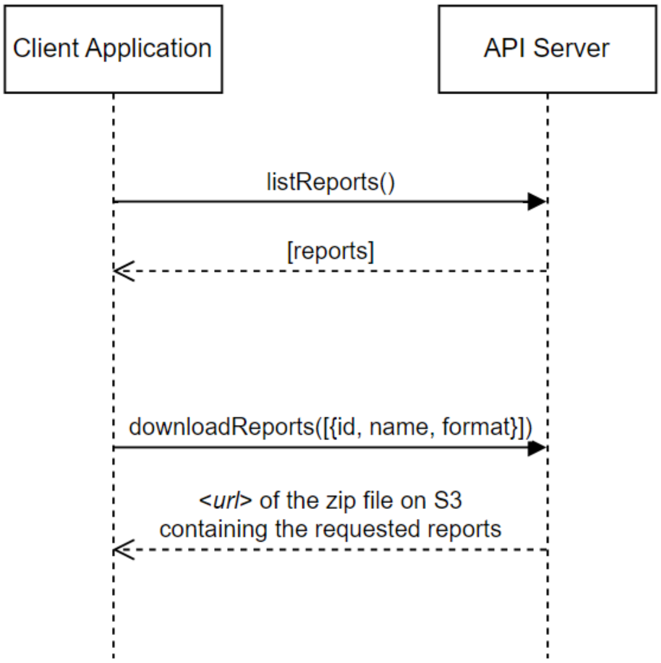
This section focuses on the mutations that can be executed via API. Mutations are used to perform operational requests (e.g. posting amendments).

The sections below describe each API and provide request and response examples.

6.1 Download Reports API

The Download Reports API returns a temporary link that client applications must use to download the .zip file containing the requested reports. The API requires as input an array of reports that the client application wants to download. For each report the client must specify ID, desired formats and report name to be downloaded. To obtain this information the client must first retrieve the list of available reports, as explained in section 5.10.

The diagram below shows the flow that the client applications must implement to download reports:


Client Application
API Server
listReports()
[reports]
downloadReports([{id, name, format}])
<url> of the zip file on S3
containing the requested reports

6.1.1 Sample Download Reports Request

```
mutation Mutation($reports: [DownloadReportInput]!) {
  downloadReports(reports: $reports)
}

variables:
{
  "reports": [
    {
      "id": 1723066454,
      "formats": ["csv","xml"]
    }
  ]
}
```

6.1.2 Sample Download Reports Response

```
{
  "data": {
    "downloadReports": "https://< SIGNED-URL-TO-DOWNLOAD-DOCUMENT > "
  }
}
```

6.2 Submit Historical Positions Request API

The Clearing System allows the immediate retrieval of the Positions for a certain period of time (details of time period to be provided in a future version of this document). To obtain Positions older than the specified period, the client applications must request a restore operation. The client application must specify the time range for which the restore operation is requested; only the Positions generated in that time range will be restored. The response of the Submit Historical Positions Request API will return information regarding the request.

The time range to be specified in the request is limited (details to be provided in a future version).

The restored data must be retrieved performing a *listPositions* query, detailed in section 5.6.

The data for which the client applications can request the restore must be more recent than 10 years.

The client application can subscribe to the *onPositionsHistoricalRequestsFeed* subscription described in section 7.2 to receive updates on the restore request.

6.2.1 Sample Submit Historical Positions Request

```
mutation SubmitPositionsHistoricalRequest($input: DateAsInput!) {
  submitPositionsHistoricalRequest(input: $input) {
    disposal_id
    mbr
    gcm
    user_id
    created_at_tmsp
    modified_at_tmsp
    status
    err_code
    err_desc
    crud
    date_from
    date_to
  }
}
variables:
{
  “input”: {
    “dateFrom”: 20220810,
    “dateTo”: 20220820
  }
}
```

6.2.2 Sample Submit Historical Positions Response

```
{
  “data”: {
    “submitPositionsHistoricalRequest”: {
      “disposal_id”: "09E01C5DE92B411A942CB52FE46DB391",
      “mbr”: "1000",
      “gcm”: "1000",
      «user_id»: "email@email.com",
      “created_at_tmsp”: "20221026T185329Z",
      “modified_at_tmsp”: "20221026T185329Z",
      „status“: "L",
      „err_code“: null,
      „err_desc“: null,
      “crud”: "I",
      “date_from”: 20220810,
      “date_to”: 20220820
    }
  }
}
```

6.3 Submit Historical Trades Request API

The Clearing System allows the immediate retrieval of the Trades for a certain period of time (details of time period to be provided in a future version of this document). To obtain Trades older than the above mentioned period, the client applications must request a restore operation. The client application must specify the time range for which the restore operation is requested; only the Trades generated in that time range will be restored. The response of the Submit Historical Trades Request API will return information regarding the request.

The time range to be specified in the request is limited (to be provided in a future version).

The restored data must be retrieved performing a *listTrades* query, detailed in section 5.3.

The data for which the client applications can request the restore must be more recent than 10 years.

The client application can subscribe to the *onTradesHistoricalRequestsFeed* subscription described in section 7.2 to receive updates on the restore request.

6.3.1 Sample Submit Historical Trades Request

```
mutation Mutation($input: DateAsInput!) {
  submitTradesHistoricalRequest(input: $input) {
    disposal_id
    mbr
    gcm
    user_id
    created_at_tmsp
    modified_at_tmsp
    status
    err_code
    err_desc
    crud
    date_from
    date_to
  }
}
variables:
{
  "input": {
    "dateFrom": 20220810,
    "dateTo": 20220820
  }
}
```

6.3.2 Sample Submit Historical Trades Response

```
{
  "data": {
    "submitTradesHistoricalRequest": {
      "disposal_id": "E7742A011F6E4F099DA6C52B2B64935B",
      "mbr": "1000",
      "gcm": "1000",
      "user_id": "email@email.com",
      "created_at_tmsp": "20221026T192836Z",
      "modified_at_tmsp": "20221026T192836Z",
      "status": "L",
      "err_code": null,
      "err_desc": null,
      "crud": "I",
      "date_from": 20220810,
      "date_to": 20220820
    }
  }
}
```

7. SUBSCRIPTIONS

This section focuses on the subscriptions that can be executed via API. Mutations are used to activate real-time data flows.

The sections below describe each API and provide request and response examples.

Client applications that subscribe to Euronext Clearing services must use the same WebSocket subprotocol as the API Server. The library adopted is *graphql-ws*; more information on the usage of this library is documented online: Apollo graphql guide.

7.1 Positions Feed API

The Positions Feed API allows client applications to subscribe to real-time feeds and receive the new positions generated by the Clearing House, or position updates, without the need to send multiple queries to the clearing system.

7.1.1 Sample Positions Feed Request

```
subscription Subscription {
  onPositionsFeed {
    member
    input {
      acct
      amt
      asset_type
      end_valid_dt
      gcm
      isin
      margin_acct_id
      mbr
      miti
      pos_acct_id
      position_id
      qty
      qty_type
      settl_curcy
      settl_dt
      settle_ref
      side
      trade_dt
    }
  }
}
```

7.1.2 Sample Positions Feed Response

```
{
    "data": {
      "onPositionsFeed": {
        "member": "1000",
        "input": [
          {
            "acct": "C",
            "amt": 14000000000.00,
            "asset_type": null,
            "end_valid_dt": "2023-02-07",
            "gcm": "2000",
            "isin": "ENXC00000004",
            "margin_acct_id": "A-MAHOUSE",
            "mbr": "1000",
            "miti": null,
            "pos_acct_id": "A-004-PAH01",
            "position_id": 221221003600,
            "qty_type": "U",
            "qty": 1000000.000,
            "settl_curcy": "EUR",
            "settl_dt": "2022-12-23",
            "settle_ref": "EX221221AAAC7N01",
            "side": "B",
            "trade_dt": "2022-12-21"
          }
        ]
      }
    }
  }
```

7.2 Historical Positions Request Feed API

The Historical Positions Feed Request API allows client applications to subscribe to real-time feeds on Historical Positions Restore Requests. This subscription sends updates on the restore requests performed by client applications (as described in section 6.2).

7.2.1 Sample Historical Positions Feed Request

```
subscription OnPositionsHistoricalRequestsFeed {
  onPositionsHistoricalRequestsFeed {
    member
    input {
      disposal_id
      status
    }
  }
}
```

7.2.2 Sample Historical Positions Feed Response

```
{
    "data": {
      "onPositionsHistoricalRequestsFeed": {
        "member": "1000",
        "input": [
          {
            "disposal_id": "09E01C5DE92B411A942CB52FE46DB391",
            "status": "H"
          }
        ]
      }
    }
  }
```

7.3 Reports Feed API

The Reports Feed API allows client applications to subscribe to real-time feeds on Reports in order to be notified when new reports become available.

7.3.1 Sample Reports Feed Request

```
subscription OnReportsFeed {
  onReportsFeed {
    member
    input {
      gcm
      report_name
      report_id
      report_version
    }
  }
}
```

7.3.2 Sample Reports Feed Response

```
{
  "data": {
    "onReportsFeed": {
      "member": "1000",
      "input": [
        {
          "gcm": "1000",
          "report_name": "20221003-DP01-1111-1",
          "report_id": 1723066454,
          "report_version": 1
        }
      ]
    }
  }
}
```

7.4 Settlement Positions Feed API

The Settlement Positions Feed API allows client applications to subscribe to real-time feeds on Settlement Positions.

7.4.1 Sample Settlement Positions Feed Request

```
subscription OnSettlementPositionsFeed {
  onSettlementPositionsFeed {
    member
    input {
      gcm
      delivery_acct_id
      unsettled_qty
      unsettled_amt
      settle_ref
    }
  }
}
```

7.4.2 Sample Settlement Positions Feed Response

```
{
  "data": {
    "onSettlementPositionsFeed": {
      "member": "1000",
      "input": [
        {
          "gcm": "1000",
          "delivery_acct_id": "DA-159875",
          "unsettled_qty": 0,
          "unsettled_amt": 0,
          "settle_ref": "BX220924AAAA2"
        }
      ]
    }
  }
}
```

7.5 Trades Feed API

The Trades Feed API allows client applications to subscribe to real-time feeds on Trades in order to receive trade confirmations in real time.

7.5.1 Sample Trades Feed Request

```
subscription OnTradesFeed {
  onTradesFeed {
    member
    input {
      gcm
      mbr
      isin
      mic
      qty
      side
    }
  }
}
```

7.5.2 Sample Trades Feed Response

```
{
  "data": {
    "onSettlementPositionsFeed": {
      "member": "1000",
      "input": [
        {
          "gcm": "2000",
          "mbr": "1000",
          "isin": "BE0003008019",
          "mic": "XBRU",
          "qty": 24,
          "side": "B"
        }
      ]
    }
  }
}
```

7.6 Historical Trades Request Feed API

The Historical Trades Request API allows client applications to subscribe to real-time feeds on Historical Trades Restore Requests. This subscription sends updates on the requests performed by client applications (as described in section 6.3).

7.6.1 Sample Historical Trades Request

```
subscription OnTradesHistoricalRequestsFeed {
  onTradesHistoricalRequestsFeed {
    member
    input {
      disposal_id
      status
    }
  }
}
```

7.6.2 Sample Historical Trades Response

```
{
  "data": {
    "onTradesHistoricalRequestsFeed": {
      "member": "1000",
      "input": [
        {
          "disposal_id": "E7742A011F6E4F099DA6C52B2B64935B",
          "status": "H"
        }
      ]
    }
  }
}
```

7.7 Collateral Deposits Feed API

The Collateral Deposits Feed API allows client applications to subscribe to real-time feeds related to deposit movements on Collateral Accounts owned by the Client.

7.7.1 Sample Collateral Deposits Feed Request

```
subscription OnCollateralDepositsFeed{
  onCollateralDepositsFeed{
    member
    input {
      coll_acct_id
      isin
      qty
    }
  }
}
```

7.7.2 Sample Collateral Deposits Feed Response

```
{
  "data": {
    "onCollateralDepositsFeed": {
      "member": "1000",
      "input": [
        {
          "coll_acct_id ": "A-007-CAC006",
          "isin": “BE0003008019”,
          "qty": 24
        }
      ]
    }
  }
}
```

7.8 Operations Feed API

The Operations Feed API allows client applications to subscribe to real-time feeds on operational requests. The subscription will send feeds providing only the main characteristics of the operational request; to retrieve the details of the operational request the client application must perform a Get Operational Request Details API call (see section 5.16)

7.8.1 Sample Operations Feed Request

```
subscription OnOperationalRequestsFeed{
  onOperationalRequestsFeed{
    member
    input {
      disposal_id
      operation
      status
    }
  }
}
```

7.8.2 Sample Operations Feed Response

```
{
  "data": {
    "onOperationalRequestsFeed": {
      "member": "1000",
      "input": [
        {
          "disposal_id": "319901b0-e004-47ef-9be1-4c988cc431f7",
          "operation": "D",
          "status": "L"
        }
      ]
    }
  }
}
```

7.9 Collateral Balance Feed API

The Collateral Balance Feed API allows client applications to subscribe to real-time feeds related to the balance on Collateral Accounts owned by the Client.

7.9.1 Sample Collateral Balance Feed Request

```
subscription OnCollateralBalancesFeed{
  onCollateralBalancesFeed{
    member
    input {
      coll_acct_id
      margin_requirement
      collateral_call
    }
  }
}
```

7.9.2 Sample Collateral Balance Feed Response

```
{
  "data": {
    "onCollateralBalancesFeed": {
      "member": "1000",
      "input": [
        {
          "coll_acct_id ": "A-007-CAC006",
          "margin_requirement": 160000.00,
          "collateral_call": 72000
        }
      ]
    }
  }
}
```

7.10 Notifications Feed API

The Notifications Feed API allows client applications to subscribe to real-time feeds on Notifications.

7.10.1 Sample Notification Feed Request

```
subscription Subscription {
    onNotificationsFeed {
      mbr
      notification_audience
      notification_category
      notification_id
      notification_level
      notification_persistence
      notification_status
      notification_text
      notification_title
      notification_tmsp
      notification_type
      user_id
    }
  }
```

7.10.2 Sample Notification Feed Response

```
{
    "data": {
      "onNotificationsFeed": {
        "member": "1030",
        "input": [
          {
            "mbr": "1030",
            "notification_audience": "U",
            "notification_category": "A",
            "notification_id": "NOTIFICATIONID123456",
            "notification_level": "I",
            "notification_persistence": "P",
            "notification_status": "N",
            "notification_text": "This is a system info notification for
participant 1030.",
            "notification_title": "System INFO",
            "notification_tmsp": "1669113000000",
            "notification_type": "C",
            "user_id": "user@organization.com"
          }
        ]
      }
    }
  }
```

8. LIST OF ATTRIBUTES

Field name	Description
accr_int	Accrued Interest
acct	Account Type. Please refer to the section 9 for the possible values.
added_tmsp	Added Timestamp
additional_info	Additional information sent in the operational request. Free text that can be filled with useful information.
adjustment_factor	Adjustment Factor
agent	Settlement Agent
amt	Amount
args_cash	Detail of the operational requests executed on collateral cash. Represents an object.
asset_type	Asset Type of the instrument. Follows the ISO 10962 and represents the first character of the CFI Code
balance_hct	Balance after application of Currency/ISIN Haircut and Wrong Way Risk
balance_mtm	It's the result of the application of the MTM process to the position in its own denomination currency before applying haircut and concentration limits.
bic_party_2	BIC Party 2
bic_party_3	BIC Party 3
buy_in_dt	Buy In Date. This date is available only on failed positions and represents the eventual date
buy_in_status	Buy In Status. Indicates whether the position is flagged for a buy-in. Possible values: ▪ 'N': 'No' ▪ 'Y': 'Yes'
cash_available	Collateral Cash available on the collateral account
cash_buffer	Permanent reserve of cash EUR at collateral account level
cash_excess	Excess of available cash collateral in regards to calculated exposure
cash_required	Minimum Cash amount required to cover Margin requirement
cash_used	Amount of cash used

cash_settl_dt	Cash Settlement Date. This date is available only on failed positions and represents the eventual settlement date
cash_settl_status	Cash Settlement Status. Indicates whether the position is flagged for a cash settlement. Possible values: ▪ 'N': 'No' 'Y': 'Yes'
ccp_bic_code	The CCP BIC Code
ccp_sec_acct	The CCP Security Account Identifier associated with the CSD of settlement
cfi_code	CFI Code
clearing_curncy	Currency in which the collateral call is done in ISO4217
client_order_id	Client Order ID
closing_price	Last adjusted closing price of the instrument
coll_acct_id	Unique ID of the account Collateral Account
collateral_call	Calculated collateral call per collateral account
collateral_curncy	Instrument denomination currency expressed following ISO 4217
collateral_description	Asset description
collateral_evaluation_tmsp	Timestamp of the collateral evaluation
collateral_subtype	Represents the sub-class of the instrument asset type
collateral_type	Represents the instrument asset type. Please refer to the section 9 for the possible values.
corporate_action_fraction	Fractional quantity left as a remainder after a Corporate Action Transformation
corporate_event	Corporate Event Type
corporate_event_indicator	Corporate event indicator. Indicates whether the position is undergoing a corporate event
corporate_msg_ref	Corporate message reference. Reference to the transformation/claim instruction
counterparty_code	Counterparty Code
country	Country
country_limit_excess	Security collateral excess due to country limit per collateral account
coupon_freq	Coupon Frequency
coupon_rate	Coupon Rate
created_at_tmsp	Timestamp of the request/position/settlement position creation

crud	CRUD, field used to track the operation executed on the record. Please refer to the section 9 for the possible values.
csd_bic_ccp	CSD BIC Code of the CCP
csd_bic_cm	CSD Bic Code of the Clearing Member
csd_settle_acct	CSD Settlement Account ID
ctv	Countervalue
curncy_hct	Currency Haircut
curr_exch_rate	Exchange Rate
date_from	Date From, used to specify the time range for historical data restore requests
date_to	Date To, used to specify the time range for historical data restore requests
delivery_acct_id	Delivery Account, internal account to Euronext Clearin
delivery_position_id	Delivery Position ID, identifier of one specific settlement position
deposit_id	Unique ID of the deposit on the Collateral Account
disposal_args_id	Unique ID of the detail of operational requests.
disposal_id	Disposal ID, identifier of an operational request
effective_settl_dt	Date when a transaction is effectively settled
end_valid_days	End Validity Days, days between the Intended Settlement Date and the End Validity Date
end_valid_dt	The final date of validity for the instruction
err_code	Error Code
err_desc	Error Description
excluded_flag	The field indicates if an asset is temporarily excluded due to a corporate action or because it is near to maturity. Possible values: ▪ '0': 'False' ▪ '1': 'True'
exec_id	Exec ID, corresponds to the Trade Unique Identifier (TUI)
execution_type	Execution Type, indicates if the trade is an insertion or a cancellation. Please refer to the section 9 for the possible values.
expiration_dt	Expiration Date, last day of the availability of a report
expiry_tmsp	Expiration time of the file generated as outcome of the restore data request
fail_acct	Fail Account

file_url	File URL generated as outcome of the restore data request
french_registered_flag	French Registered Security flag that indicates if the settlement must follow specific rules of Euroclear France. This field is set to True if the Main Depository is Euroclear France and if the Legal form field is purely registered. Possible values: ▪ 'N': 'No' ▪ 'Y': 'Yes'
gcm	Clearing Member Code, internal to Euronext Clearing
guaranteed_flag	Guaranteed Indicator, specifies if the trade or the instrument is guaranteed or not. Possible values: ▪ '0': 'False' ▪ '1': 'True'
haircut	Haircut applied to the position (in percentage terms) as a consequence of the exchange rate conversion. When Settlement Currency and Clearing currency are equal, haircut is zero
hold_indicator	Hold Indicator, indicates the status of the instruction. Possible values: ▪ 'N': 'No' ▪ 'Y': 'Yes'
id	ID used by the UI. Client applications should ignore this field.
index_name	Index Name
instr_desc	Instrument Description
instr_group_code	Instrument Group Code
instr_status	Instrument Status. Please refer to the section 9 for the possible values.
instr_subtype	Instrument Subtype
instr_trading_code	Instrument Trading Code
instr_type	Instrument Type
instr_unit	Instrument Unit. Please refer to the section 9 for the possible values.
isin	ISIN
isin_hct	Haircut applied to instrument used as collateral
legal_form	Legal Form. Please refer to the section 9 for the possible values.
liquid_indicator	Liquidity Indicator, indicates whether the instrument is liquid or not, as defined per MiFID II. Possible values: ▪ '0': 'False' ▪ '1': 'True'

listing_dt	Listing Date, first day of trading for the instrument
main_depository	Main Depository. Please refer to the section 9 for the possible values.
margin_acct_id	Margin Account ID, internal to Euronext Clearing
margin_requirement	Total margin requirement calculated at margin account level
margin_requirement_tmsp	Total margin requirement timestamp
market_price	Transaction price
market_venue	Market Venue
matching_status	Matching Status. Indicates whether the SI is already matched or to be matched. Please refer to the section 9 for the possible values.
maturity_dt	Maturity Date of the Instrument, when relevant
mbr	Member Code, internal to Euronext Clearing
mic	MIC
miti	CSD Settlement Reference
modified_at_tmsp	Timestamp of last update to the operational request/position/settlement position
msg_sequence	Message Sequence
mtm_amt	Mark To Market Amount
mtm_price	Price used to calculate the mtm_amt
mtm_tmsp	Mark to Market calculation Timestamp
ncb_used	Amount of NCB Guarantee used as collateral
netting_rule	Indicates the netting rule applied. Please refer to the section 9 for the possible values.
notification_audience	Indicates to whom the notification is addressed. Please refer to the section 9 for the possible values.
notification_category	Category of the notification
notification_id	Unique ID of the notification
notification_level	Indicates the severity of the notification. Please refer to the section 9 for the possible values.
notification_persistence	Indicates if the notification needs Client's acknowledgment. Please refer to the section 9 for the possible values.
notification_text	Body of the notification
notification_title	Title of the notification

notification_tmsp	Notification Timestamp
notification_type	Indicates the type of the notification. Please refer to the section 9 for the possible values.
operation	Operational Request type. Please refer to the section 9 for the possible values.
order_id	Order ID
partial_settl_status	Partial settlement indicator. Is null if the position is not a partial settlement, otherwise contains the status of the partial settlement status. Please refer to the section 9 for the possible values.
participant_code	Participant Code, internal to Euronext Clearing
participant_description	Participant Description
payment_flag	It shows if a cash call or restitution should be done. Values could be: • 'Y': 'Yes' (payment instruction is generated) • 'N': 'No' (no payment instruction is generated) The payment instruction generation depends on the threshold set by the Client and by the Clearing House. The value is always set to "Y" Only valid for EOD margin call.
pos_acct_description	Description of the Position Account
pos_acct_id	Position Account ID, internal to Euronext Clearing
position_id	Position ID
position_source	Position Source. Please refer to the section 9 for the possible values.
position_status	Position Status. Please refer to the section 9 for the possible values.
position_type	Position Type. Please refer to the section 9 for the possible values.
previous_settl_ref	Previous Settlement Reference
price	Price applied to security. Null in case of cash deposits
qty	Quantity
qty_type	Quantity type. Please refer to the section 9 for the possible values.
reason_code	The reason code for the last change in settlement status (e.g.: the reason for cancellation)
reason_descr	The reason for the last change in status (e.g.: the reason for cancellation)
report_code	Report Description

report_description	Report Description
report_format	Report Format
report_id	Report ID
report_name	Report Name
report_status	Report Status. Please refer to the section 9 for the possible values.
report_tmsp	Report Timestamp
report_version	Report Version
request_amt	Amount of cash/securities/bonds requested in an operational request executed on collateral
restore_request_tmsp	Restore Request Timestamp
securities_after_limits	Overall amount of collateral (securities), after limit application
securities_available	Overall amount of collateral (securities), after currency conversion and haircut and before limit application.
securities_used	Amount of securities, expressed in clearing currency, used as collateral
segment	Participant's Trading Segment. Please refer to the section 9 for the possible values.
settl_curcy	Settlement Currency
settl_dt	Settlement Date
settle_amt	Settlement Amount
settle_delay	Settlement Delay
settle_per	Settlement Period
settle_ref	Settlement Reference
settle_source	Settlement Source. Please refer to the section 9 for the possible values.
settle_status	Settlement Status. Please refer to the section 9 for the possible values.
settle_system	Settlement System. Please refer to the section 9 for the possible values.
side	Side. Please refer to the section 9 for the possible values.
size	Size of the file generated as outcome of the data restore request
ssi_id	Internal identification code of the external collateral account
status	Status of the operational request. Please refer to the section 9 for the possible values.
strike_curncy	Strike Currency
strike_price	Strike Price

symbol_index	Symbol Index
text	Text, sent by the Client when inserting the order
tradable_amt	Minimum tradable amout
trade_capacity	Trading Capacity. Please refer to the section 9 for the possible values.
trade_curncy	Trading Currency
trade_dt	Trade Date
trade_tm	Trade Time
transfer_src_acct	Source Collateral account from which the assets are transferred. Relevant for collateral transfer operational requests.
transfer_tgt_acct	Target Collateral account to which the assets are transferred. Relevant for collateral transfer operational requests.
transformation_fraction	Fraction resulting from a transformation instruction
unsettled_amt	Unsettled Amount
unsettled_qty	Unsettled Quantity
user_id	User that has requested the operation
wwr_amt	Measure of the wrongway risk for collateral

9. STATIC VALUES

Field	Enum Values
acct	'C': 'Client' 'H': 'House' 'L': 'Liquidity Provider’
collateral_type	'B': 'Bond' 'C': 'Cash' 'S': 'Security'
crud	'D': 'Delete' 'H': 'Detach' 'I': 'Insert' 'S': 'Switch' 'U': 'Update'
execution_type	'1': 'Trade Insertion' '2': 'Trade Cancellation'
instr_status	'1': 'Traded' '2': 'Temporary Suspended' '3': 'Permanently Suspended’
instr_unit	'1': 'Unit' '2': 'Percentage Clean' '4': 'Percentage Mixed' '5': 'Percentage Dirty' '7': 'Yield' '8': 'Per kilogram' '9': 'Per ounce'
legal_form	'0': 'Bearer or Registered' '2': 'Purely registered' '3': 'Purely bearer' '8': 'Not applicable'
main_depository	'00001': 'Euroclear France' '00002': 'Euroclear Belgium' '00003': 'Euroclear Nederland' '00010': 'Euronext Securities Porto' '00004': 'NBB-SSS' '00006': 'Euroclear Bank'
matching_status	'AM':'Already Matched' 'TBM':'To Be Matched'
netting_rule	'AGGR': 'Aggregation' 'SING': 'Single Netting'
notification_audience	'A': 'All' 'P': 'Participant' 'U': 'User'
notification_level	'E':'Error' 'I':'Info' 'W':'Warning'
notification_persistence	'E':'Ephemeral' 'P':'Persistent'

notification_type	'C':'Clearing' 'S':'System'
operation	'D':'Deposit' 'S':'Substitution' 'W':'Withdrawal'
partial_settle_status	'PARC: 'Partially Confirmed' 'PAIN': 'Partial Settlement '
position_source	'ST': 'Trading' 'CA': 'Corporate Action' 'BP': 'Buyer Protection'
position_status	'LIVE': 'Open' 'CLSD': 'Closed'
position_type	'S': 'Active' 'F': 'Failed'
qty_type	'F': 'Face Value' 'U': 'Unit'
report_status	'A': 'Available' 'R': 'Restoring' 'E': 'Error' 'N': 'Not Available'
segment	'Euronext Equity Markets':'Euronext Equity Markets'
settle_source	'T': 'Trading' 'CA': 'Corporate Action' 'BP': 'Buyer Protection' 'PO': 'Pair Off' 'BI': 'Buy In' 'OR': 'Other Reason'
settle_status	'CAND': 'Cancelled' 'FULL': 'Settled' 'PEND': 'Pending' 'PENF': 'Failed'
settle_system	'60': 'T2S' '51': 'Euroclear Bank'
side	‘B’: ‘Buy’ ‘S’: ‘Sell’
status	'E':'Error' 'L':'Loading' 'H':'Hold 'P':'Processing' 'R':'Rejected' 'V':'Valid' 'X':'Running'
trade_capacity	'1': 'Agent' '2': 'Principal' '3': 'Other'

10. ERROR REGISTER

List of Error messages returned to the Client Application:

- Authentication not valid: error returned when the client application has not provided a token or it has provided a bad token (bad formatting, expired, etc).
- File format not valid: error returned when the format specified in the request is not acceptable.
- Input not valid: error returned when the input provided by the client application is not valid.
- Internal server error: error returned when there is an issue inside the Clearing System that does not depend on the client request.
- Operation not allowed: error returned when the client application is requesting an operation for which it does not have sufficient permissions.
- Operation not found: error returned when the client application is requesting an operation that does not exist or that the server cannot expose.
- Separator character not specified: error returned when the CSV delimiter to use for the export functionality is not provided by the Client Application.
- Separator character not valid: error returned when the CSV delimiter to use for the export functionality is not valid.
- Too many requests for client: error returned when the client application has exceeded the rate limit of requests per interval.

Error codes for Operational requests (for Submit Historical Positions Request API, Submit Historical Trades Request API):

- E001: Unable to find the operation definition (Request Validation KO).
- E002: The operation is badly formatted (Request Validation KO).
- E003: Unable to execute the operation (Generic Exception).
- E004: Operation timed out.
- E0010: Error while executing the operation (SQL Exception).
- E0020: The operation is badly formatted (Request Validation KO).
- E0030: Unable to find the operation definition (IO Exception).
- E0040: Unable to execute the operation (Operation disabled).
- E0050: The requested operation is temporarily disabled (Operation Out of Time, this is valid for time limited requests).
- E0099: Unable to execute the operation (Generic Exception).
- ERROR: Unable to execute the operation (Internal Server Error).