EURONEXT CLEARING

TECHNICAL SPECIFICATIONS

Document title

Euronext Clearing Application Programming Interfaces Specifications

Document type or subject

EURONEXT CLEARING API SPECIFICATIONS

Revision number
1.0

Date
02 November 2022

Number of pages
70

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 details of a subset of Euronext Clearing Services. Future versions will include sections related to interfaces that allow Clients to perform commands and make operational requests to the Clearing House.

Details relating to Connectivity will be provided in dedicated separate documents, 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

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

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 plans 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.

The first version of this document describes how to manage the following topics:

- create Client Credentials and generate Access Tokens in order to interact with Euronext Clearing Systems;
- retrieve Reports produced by Euronext Clearing via API;
- perform API Queries on Referential Data (Instruments, Accounts) and Real-Time Data (Trades, Positions, Settlement Positions);
- perform API Mutations to download reports and request data restoration;
- perform Subscriptions to receive real-time data feeds;
- export data via API.

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

Any modification to this version will be clearly stated in the *What's New* section of the Preface to 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.

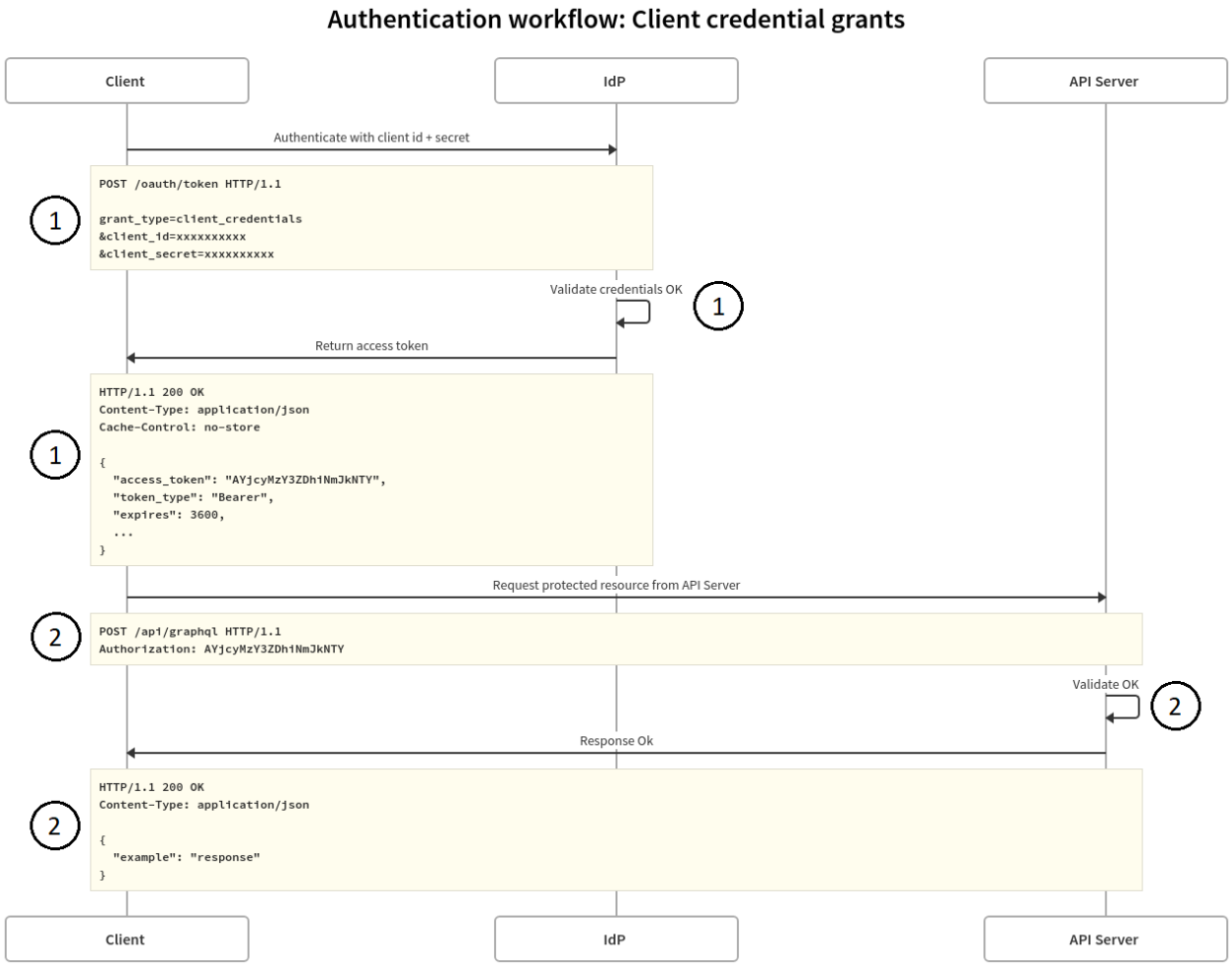
2.1 API Manager for Client Credentials

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:



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 *Authentication* 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:

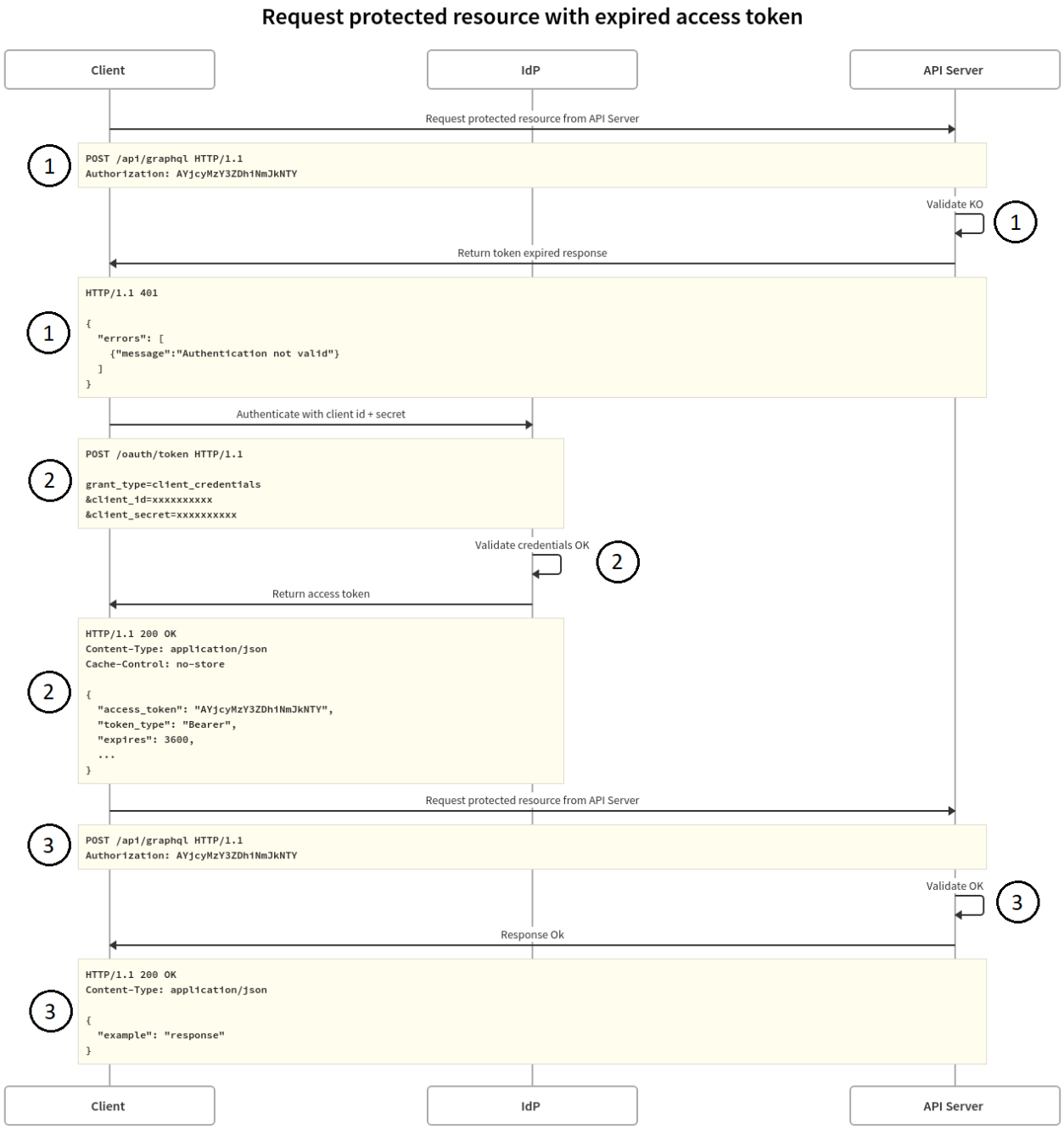
```
curl --location --request POST
'https://<IDP_BASE_URL>/<AUTH_SERVER_ID>/v1/token' \
--header 'Authorization: Basic <CLIENT_ID_CLIENT_SECRET_ENCODED>' \
--header 'Content-Type: application/x-www-form-urlencoded' \
```

```
    --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:



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.

```
scalar Upload

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 PositionCols {
  accr_int
  acct
  adjustment_factor
  amt
  asset_type
  buy_in_dt
  buy_in_status
  cash_settl_dt
  cash_settl_status
```

```
  corporate_event
  corporate_event_indicator
  created_at_tmsp
  crud
  csd_bic_ccp
  csd_bic_cm
  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_settle_ref
  qty_type
  qty
  settl_curcy
  settl_dt
  settle_ref
  settle_system
  side
  symbol_index
  trade_dt
  unsettled_amt
  unsettled_qty
}

enum InstrumentCols {
  accr_int_start_dt
  adjustment_factor
  adt
  asset_type
  cfi_code
  corporate_event
  country
```

```
  coupon_freq
  coupon_rate
  crud
  end_valid_days
  exercise_style
  fixed_coupon
  guaranteed_flag
  index_name
  instr_desc
  instr_group_code
  instr_status
  instr_subtype
  instr_trading_code
  instr_type
  instr_unit
  isin
  legal_form
  liquid_indicator
  listing_dt
  main_depository
  closing_price
  maturity_dt
  mic
  minimum_settl_amt
  msg_sequence
  par_value
  qty_type
  redemption_freq
  settl_curcy
  settle_delay
  settle_system
  strike_curncy
  strike_price
  symbol_index
  trade_curncy
}

enum ReportCol {
  mbr
  gcm
  agent
  report_tmsp
  expiration_dt
  report_type
  report_name
  report_code
  report_version
```

```
  report_format
  report_status
  added_tmsp
  restore_request_tmsp
  crud
}

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
  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_settle_ref
  qty_type
  qty
  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
}

enum SortDirection {
  ASC
  DESC
}

enum TradeCols {
  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_type
  qty
  settl_curcy
  settl_dt
  settle_amt
  settle_per
  settle_ref
```

```
  settle_system
  side
  symbol_index
  text
  trade_capacity
  trade_curncy
  trade_dt
  trade_tm
}

input AcctValue {
  val: String! @constraint(pattern: "^(C)|(H)|(L)$", minLength: 1, maxLength: 1)
}

input BoolValue {
  val: String! @constraint(pattern: "^(Y)|(N)$", minLength: 1, maxLength: 1)
}

input PosTypeValue {
  val: String! @constraint(pattern: "^(S)|(F)$", minLength: 1, maxLength: 1)
}

input PosStatusValue {
  val: String! @constraint(pattern: "^(LIVE)|(CLSD)$", minLength: 4, maxLength: 4)
}

input QtyTypeValue {
  val: String! @constraint(pattern: "^(F)|(U)$", minLength: 1, maxLength: 1)
}


input ReportStatusValue{
  val: String! @constraint(pattern: "^(A)|(R)|(E)|(N)$", minLength: 1, maxLength:
1)
}


input SignValue {
  val: String! @constraint(pattern: "^(B)|(S)$", minLength: 1, maxLength: 1)
}
input PositionSourceValue {
  val: String! @constraint(pattern: "^(ST)|(CA)|(BP)$", minLength: 2, maxLength: 2)
}

input SettlementSourceValue {
  val: String! @constraint(pattern: "^(T)|(CA)|(BP)|(PO)|(BI)|(OR)$", minLength: 1,
maxLength: 2)
}
```

```
input MainDepositoryValue {
  val: String! @constraint(pattern: "^(00001)|(00002)|(00004)|(00006)|(00010)$",
minLength: 5, maxLength: 5)
}

input SettlementSystemValue {
  val: String! @constraint(pattern: "^(60)|(01)|(1)|(2)$", minLength: 1, maxLength:
2)
}

input TradeCapacityValue {
  val: String! @constraint(pattern: "^(1)|(2)|(3)$", minLength: 1, maxLength: 1)
}

input ExecutionTypeValue {
  val: String! @constraint(pattern: "^(1)|(2)$", minLength: 1, maxLength: 1)
}

input GuaranteedFlagValue {
  val: String! @constraint(pattern: "^(0)|(1)$", minLength: 1, maxLength: 1)
}

input InstrumentStatusValue {
  val: String! @constraint(pattern: "^(1)|(2)|(3)$", minLength: 1, maxLength: 1)
}

input InstrumentUnitValue{
  val: String! @constraint(pattern: "^(1)|(2)|(4)|(5)|(7)|(8)|(9)$", minLength: 1,
maxLength: 1)
}

input AssetTypeValue {
  val: String! @constraint(pattern: "^(E)|(D)|(R)|(F)|(O)$", minLength: 1,
maxLength: 1)
}

input SettlementStatusValue {
  val: String! @constraint(pattern: "^(CAND)|(REJT)|(FULL)|(PEND)|(PENF)$",
minLength: 4, maxLength: 4)
}

input CorporateEventValue {
  val: String! @constraint(pattern: "^(00)|(10)$", minLength: 2, maxLength: 2)
}

input ExerciseStyleValue {
```

```
  val: String! @constraint(pattern: "^(0)|(1)|(2)|(3)|(4)|(5)$", minLength: 1,
maxLength: 1)
}

input NetRuleValue {
   val: String! @constraint(pattern: "^(AGGR)|(SING)$", minLength: 4, maxLength: 4)
}

input PartialSettlementStatusValue {
   val: String! @constraint(pattern: "^(PARC)|(PAIN)$", minLength: 4, maxLength: 4)
}

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

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

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

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

input InstrumentFilterModel {
  asset_type: ModelStringAssetTypeInputSet
  closing_price: ModelFloatInput
  corporate_event: ModelStringCorporateEventInputSet
  country: ModelStringCountryInput
  coupon_freq: ModelStringCouponFreqInputSet
  end_valid_days: ModelIntInput
  exercise_style: ModelStringExerciseStyleInputSet
  guaranteed_flag: ModelGuaranteedFlagInputSet
  index_name: ModelStringInput
  instr_desc: ModelStringInput
  instr_group_code: ModelStringInput
  instr_status: ModelInstrumentStatusInputSet
```

```
  instr_subtype: ModelStringInput
  instr_trading_code: ModelStringInput
  instr_type: ModelStringInput
  instr_unit: ModelStringInstrumentUnitInputSet
  isin: ModelStringIsinInput
  listing_dt: ModelIntInput
  main_depository: ModelMainDepositoryInputSet
  maturity_dt: DateStringFilterModel
  mic: ModelStringCode4Input
  minimum_settl_amt: ModelFloatInput
  qty_type: ModelStringQtyTypeInputSet
  settl_curcy: ModelStringInput
  settle_delay: ModelStringInput
  settle_system: ModelSettlementSystemInputSet
  strike_curncy: ModelStringInput
  strike_price: ModelFloatInput
  symbol_index: ModelFloatInput
  trade_curncy: ModelStringInput
}

input ModelFloatInput {
  eq: Float @constraint(pattern: "^[0-9.,]*$", minLength: 1)
  ne: Float @constraint(pattern: "^[0-9.,]*$", minLength: 1)
  lt: Float @constraint(pattern: "^[0-9.,]*$", minLength: 1)
  le: Float @constraint(pattern: "^[0-9.,]*$", minLength: 1)
  gt: Float @constraint(pattern: "^[0-9.,]*$", minLength: 1)
  ge: Float @constraint(pattern: "^[0-9.,]*$", minLength: 1)
  between: [Float] @constraint(pattern: "^[0-9.,]*$", minLength: 1)
}

input ModelIntInput {
  eq: Int @constraint(pattern: "^[0-9]*$", minLength: 1)
  ne: Int @constraint(pattern: "^[0-9]*$", minLength: 1)
  lt: Int @constraint(pattern: "^[0-9]*$", minLength: 1)
  le: Int @constraint(pattern: "^[0-9]*$", minLength: 1)
  gt: Int @constraint(pattern: "^[0-9]*$", minLength: 1)
  ge: Int @constraint(pattern: "^[0-9]*$", minLength: 1)
  between: [Int] @constraint(pattern: "^[0-9]*$", minLength: 1)
}

input ModelStringCountryInput {
  contains: String @constraint(pattern: "^[A-Za-z ]*$", maxLength: 2)
  notContains: String @constraint(pattern: "^[A-Za-z ]*$", maxLength: 2)
  eq: String @constraint(pattern: "^[A-Za-z ]*$", minLength: 2, maxLength: 2)
  ne: String @constraint(pattern: "^[A-Za-z ]*$", minLength: 2, maxLength: 2)
  beginsWith: String @constraint(pattern: "^[A-Za-z ]*$", maxLength: 2)
  endsWith: String @constraint(pattern: "^[A-Za-z ]*$", maxLength: 2)
```

```
}

input ModelStringAcctInputSet {
  in:[AcctValue]
}

input ModelStringBoolInputSet {
  in: [BoolValue]
}

input ModelStringCode4Input {
  contains: String @constraint(pattern: "^[0-9A-Za-z ]*$", maxLength: 4)
  notContains: String @constraint(pattern: "^[0-9A-Za-z ]*$", maxLength: 4)
  eq: String @constraint(pattern: "^[0-9A-Za-z ]*$", minLength:3, maxLength: 4)
  ne: String @constraint(pattern: "^[0-9A-Za-z ]*$", minLength:4, maxLength: 4)
  beginsWith: String @constraint(pattern: "^[0-9A-Za-z ]*$", maxLength: 4)
  endsWith: String @constraint(pattern: "^[0-9A-Za-z ]*$", maxLength: 4)
}

input ModelStringCode8Input {
  contains: String @constraint(pattern: "^[0-9A-Z ]*$", maxLength: 8)
  notContains: String @constraint(pattern: "^[0-9A-Z ]*$", maxLength: 8)
  eq: String @constraint(pattern: "^[0-9A-Z ]*$", minLength:1, maxLength: 8)
  ne: String @constraint(pattern: "^[0-9A-Z ]*$", minLength:1, maxLength: 8)
  beginsWith: String @constraint(pattern: "^[0-9A-Z ]*$", maxLength: 8)
  endsWith: String @constraint(pattern: "^[0-9A-Z ]*$", maxLength: 8)
}

input ModelStringCouponFreqInputSet {
  in: [String] @constraint(pattern: "^(1)|(2)|(3)|(4)|(5)|(6)|(255)|(8)$",
minLength: 1, maxLength: 3)
}

input ModelStringIdInput {
  contains: String @constraint(pattern: "^[0-9A-Za-z_\\-]*$", maxLength: 40)
  notContains: String @constraint(pattern: "^[0-9A-Za-z_\\-]*$", maxLength: 40)
  eq: String @constraint(pattern: "^[0-9A-Za-z_\\-]*$", minLength: 1, maxLength:
40)
  ne: String @constraint(pattern: "^[0-9A-Za-z_\\-]*$", minLength: 1, maxLength:
40)
  beginsWith: String @constraint(pattern: "^[0-9A-Za-z_\\-]*$", maxLength: 40)
  endsWith: String @constraint(pattern: "^[0-9A-Za-z_\\-]*$",maxLength: 40)
}

input ModelStringInput {
  contains: String
  notContains: String
```

```
  eq: String
  ne: String
  beginsWith: String
  endsWith: String
}

input ModelStringIsinInput {
  contains: String @constraint(pattern: "^[0-9A-Za-z ]*$", maxLength: 12)
  notContains: String @constraint(pattern: "^[0-9A-Za-z ]*$", maxLength: 12)
  eq: String @constraint(pattern: "^[0-9A-Za-z ]*$", minLength: 12, maxLength: 12)
  ne: String @constraint(pattern: "^[0-9A-Za-z ]*$", minLength: 12, maxLength: 12)
  beginsWith: String @constraint(pattern: "^[0-9A-Za-z ]*$", maxLength: 12)
  endsWith: String @constraint(pattern: "^[0-9A-Za-z ]*$", maxLength: 12)
}

input ModelStringMbrInput {
  contains: String @constraint(pattern: "^[0-9]*$", minLength: 1, maxLength: 5)
  notContains: String @constraint(pattern: "^[0-9]*$", minLength: 1, maxLength: 5)
  eq: String @constraint(pattern: "^[0-9]*$", minLength: 4, maxLength: 5)
  ne: String @constraint(pattern: "^[0-9]*$", minLength: 4, maxLength: 5)
  beginsWith: String @constraint(pattern: "^[0-9]*$", minLength: 1, maxLength: 5)
  endsWith: String @constraint(pattern: "^[0-9]*$", minLength: 1, maxLength: 5)
}

input ModelStringPosTypeInputSet {
  in: [PosTypeValue]
}

input ModelStringPosStatusInputSet {
  in: [PosStatusValue]
}

input ModelStringQtyTypeInputSet {
  in: [QtyTypeValue]
}

input ModelStringReportStatusInputSet {
  in: [ReportStatusValue]
}

input ModelStringSignInputSet {
  in: [SignValue]
}

input ModelPositionSourceInputSet {
  in: [PositionSourceValue]
}
```

```
input ModelSettlementSourceInputSet {
  in: [SettlementSourceValue]
}

input ModelMainDepositoryInputSet{
  in: [MainDepositoryValue]
}

input ModelSettlementSystemInputSet {
  in: [SettlementSystemValue]
}

input ModelTradeCapacityInputSet {
  in: [TradeCapacityValue]
}

input ModelExecutionTypeInputSet {
  in: [ExecutionTypeValue]
}

input ModelGuaranteedFlagInputSet {
  in: [GuaranteedFlagValue]
}

input ModelInstrumentStatusInputSet {
  in: [InstrumentStatusValue]
}

input ModelStringInstrumentUnitInputSet {
  in: [InstrumentUnitValue]
}

input ModelStringAssetTypeInputSet {
  in: [AssetTypeValue]
}

input ModelSettlementStatusInputSet{
  in: [SettlementStatusValue]
}

input ModelPartialSettlementStatusInputSet{
  in: [PartialSettlementStatusValue]
}

input ModelStringCorporateEventInputSet {
  in: [CorporateEventValue]
```

```
}

input ModelStringExerciseStyleInputSet {
  in: [ExerciseStyleValue]
}

input ModelNetRuleInputSet{
  in: [NetRuleValue]
}

input PaginationModel {
  offset: Int! @constraint(pattern: "^[0-9]*$")
  limit: Int! @constraint(pattern: "^[0-9]*$")
}

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

input PositionFilterModel {
  accr_int: ModelFloatInput
  acct: ModelStringAcctInputSet
  amt: ModelFloatInput
  asset_type: ModelStringAssetTypeInputSet
  buy_in_dt: ModelIntInput
  buy_in_status: ModelStringBoolInputSet
  cash_settl_dt: ModelIntInput
  cash_settl_status: ModelStringBoolInputSet
  corporate_event: ModelStringCorporateEventInputSet
  corporate_event_indicator: ModelStringBoolInputSet
  created_at_tmsp: ModelStringInput
  end_valid_dt: ModelIntInput
  french_registered_flag: ModelStringBoolInputSet
  gcm: ModelStringMbrInput
  isin: ModelStringIsinInput
  main_depository: ModelMainDepositoryInputSet
  margin_acct_id: ModelStringIdInput
  mbr: ModelStringMbrInput
  miti: ModelStringInput
  modified_at_tmsp: ModelStringInput
  mtm_price: ModelFloatInput
  pos_acct_id: ModelStringIdInput
  position_id: ModelIntInput
  position_source: ModelPositionSourceInputSet
  position_status: ModelStringPosStatusInputSet
  position_type: ModelStringPosTypeInputSet
```

```
  previous_settle_ref: ModelStringInput
  qty_type: ModelStringQtyTypeInputSet
  qty: ModelFloatInput
  settl_curcy: ModelStringInput
  settl_dt: ModelIntInput
  settle_ref: ModelStringInput
  settle_system: ModelSettlementSystemInputSet
  side: ModelStringSignInputSet
  symbol_index: ModelFloatInput
  trade_dt: ModelIntInput
  unsettled_amt: ModelFloatInput
  unsettled_qty: ModelFloatInput
}

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

input ReportFilterModel {
  mbr: ModelStringMbrInput
  gcm: ModelStringMbrInput
  agent: ModelStringMbrInput
  report_tmsp: ModelStringInput
  expiration_dt: ModelIntInput
  report_name: ModelStringInput
  report_code: ModelStringInput
  report_version: ModelIntInput
  report_format: ModelStringInput
  report_status: ModelStringReportStatusInputSet
  added_tmsp: ModelStringInput
}

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

input SettlementPositionFilterModel {
  agent: ModelStringMbrInput
  amt: ModelFloatInput
  buy_in_dt: ModelIntInput
  cash_settl_dt: ModelIntInput
  ccp_bic_code: ModelStringInput
  ccp_sec_acct: ModelStringInput
  corporate_event: ModelStringInput
  corporate_msg_ref: ModelStringInput
```

```
  created_at_tmsp: ModelStringInput
  csd_settle_acct: ModelStringInput
  delivery_acct_id: ModelStringIdInput
  effective_settl_dt: ModelIntInput
  end_valid_dt: ModelIntInput
  fail_acct: ModelStringInput
  french_registered_flag: ModelStringBoolInputSet
  gcm: ModelStringMbrInput
  hold_indicator: ModelStringBoolInputSet
  isin: ModelStringIsinInput
  main_depository: ModelMainDepositoryInputSet
  market_venue: ModelStringCode4Input
  miti: ModelStringInput
  modified_at_tmsp: ModelStringInput
  netting_rule: ModelNetRuleInputSet
  partial_settl_status: ModelPartialSettlementStatusInputSet
  previous_settle_ref: ModelStringInput
  qty_type: ModelStringQtyTypeInputSet
  qty: ModelFloatInput
  reason_code: ModelStringInput
  reason_descr: ModelStringInput
  settl_curcy: ModelStringInput
  settl_dt: ModelIntInput
  settle_ref: ModelStringInput
  settle_source: ModelSettlementSourceInputSet
  settle_status: ModelSettlementStatusInputSet
  settle_system: ModelSettlementSystemInputSet
  side: ModelStringSignInputSet
  symbol_index: ModelIntInput
  trade_dt: ModelIntInput
  unsettled_amt: ModelFloatInput
  unsettled_qty: ModelFloatInput
}
input TradeFilterModel {
  accr_int: ModelFloatInput
  acct: ModelStringAcctInputSet
  client_order_id: ModelIntInput
  counterparty_code: ModelStringCode8Input
  ctv: ModelFloatInput
  curr_exch_rate: ModelFloatInput
  exec_id: ModelStringInput
  execution_type: ModelExecutionTypeInputSet
  french_registered_flag: ModelStringBoolInputSet
  gcm: ModelStringMbrInput
  guaranteed_flag: ModelGuaranteedFlagInputSet
  isin: ModelStringIsinInput
  main_depository: ModelMainDepositoryInputSet
```

```
  market_price: ModelFloatInput
  mbr: ModelStringMbrInput
  mic: ModelStringCode4Input
  order_id: ModelIntInput
  pos_acct_id: ModelStringIdInput
  position_id: ModelIntInput
  qty_type: ModelStringQtyTypeInputSet
  qty: ModelFloatInput
  settl_curcy: ModelStringInput
  settl_dt: ModelIntInput
  settle_amt: ModelFloatInput
  settle_per: ModelFloatInput
  settle_ref: ModelStringInput
  settle_system: ModelSettlementSystemInputSet
  side: ModelStringSignInputSet
  symbol_index: ModelIntInput
  trade_curncy: ModelStringInput
  trade_dt: ModelIntInput
  trade_tm: ModelIntInput
}

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

type Instrument {
  accr_int_start_dt: Int
  adjustment_factor: Float
  adt: Float
  asset_type: String
  cfi_code: String
  closing_price: Float
  corporate_event: String
  country: String
  coupon_freq: String
  coupon_rate: Float
  crud: String
  end_valid_days: Int
  exercise_style: String
  fixed_coupon: String
  guaranteed_flag: String
  index_name: String
  instr_desc: String
  instr_group_code: String
  instr_status: String
  instr_subtype: String
```

```
  instr_trading_code: String
  instr_type: String
  instr_unit: String
  isin: String
  legal_form: String
  liquid_indicator: String
  listing_dt: Int
  main_depository: String
  maturity_dt: String
  mic: String
  minimum_settl_amt: Float
  msg_sequence: Int
  par_value: Float
  qty_type: String
  redemption_freq: Int
  settl_curcy: String
  settle_delay: String
  settle_system: String
  strike_curncy: String
  strike_price: Float
  symbol_index: Int
  trade_curncy: String
}

type Position {
  accr_int: Float
  acct: String
  adjustment_factor: Float
  amt: Float
  asset_type: String
  buy_in_dt: Int
  buy_in_status: String
  cash_settl_dt: Int
  cash_settl_status: String
  corporate_event_id: String
  corporate_event_indicator: String
  created_at_tmsp: String
  crud: String
  csd_bic_ccp: String
  csd_bic_cm: String
  end_valid_dt: Int
  french_registered_flag: String
  gcm: String
  haircut: Float
  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_settle_ref: String
  qty_type: String
  qty: Float
  settl_curcy: String
  settl_dt: Int
  settle_ref: String
  settle_system: String
  side: String
  symbol_index: Int
  trade_dt: Int
  unsettled_amt: Float
  unsettled_qty: Float
}

type Positions {
  member: String!
  input: [Position]
}

type PositionsHistoricalRequest {
  disposal_id: String
  mbr: String
  gcm: String
  user_id: String
  created_at_tmsp: String
  modified_at_tmsp: String
  status: String
  err_code: String
  err_desc: String
  crud: String
  date_from: Int
  date_to: Int
}

type PositionsHistoricalRequests {
```

```
  member: String!
  input: [PositionsHistoricalRequest]
}

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

type Reports {
  member: String!
  input: [Report]
}

type SettlementPosition {
  agent: String
  amt: Float
  bic_party_2: String
  bic_party_3: String
  buy_in_dt: Int
  cash_settl_dt: Int
  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: Int
```

```
  end_valid_dt: Int
  fail_acct: String
  french_registered_flag: String
  gcm: String
  hold_indicator: 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_settle_ref: String
  qty_type: String
  qty: Float
  reason_code: String
  reason_descr: String
  settl_curcy: String
  settl_dt: Int
  settle_ref: String
  settle_source: String
  settle_status: String
  settle_system: String
  side: String
  symbol_index: Int
  trade_dt: Int
  transformation_fraction: Float
  unsettled_amt: Float
  unsettled_qty: Float
}

type SettlementPositions {
  member: String!
  input: [SettlementPosition]
}

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: String
  french_registered_flag: String
  gcm: String
  guaranteed_flag: 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_type: String
  qty: Float
  settl_curcy: String
  settl_dt: Int
  settle_amt: Float
  settle_per: Float
  settle_ref: String
  settle_system: String
  side: String
  symbol_index: Int
  text: String
  trade_capacity: String
  trade_curncy: String
  trade_dt: Int
  trade_tm: Int
}

type Trades {
  member: String!
  input: [Trade]
}

type TradesHistoricalRequest {
  disposal_id: String
  mbr: String
  gcm: String
  user_id: String
  created_at_tmsp: String
  modified_at_tmsp: String
  status: String
  err_code: String
  err_desc: String
  crud: String
  date_from: Int
```

```
  date_to: Int
}

type TradesHistoricalRequests {
  member: String!
  input: [TradesHistoricalRequest]
}

type Query @rateLimit(limit: 1000, duration: 1) {
  exportInstruments(filterModel: InstrumentFilterModel, sortModel:
[InstrumentColSort], 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
  listInstruments(filterModel: InstrumentFilterModel, sortModel:
[InstrumentColSort], paginationModel: PaginationModel): [Instrument]
  listPositions(filterModel: PositionFilterModel, sortModel: [PositionColSort],
paginationModel: PaginationModel): [Position]
  listPositionsHistoricalRequests: [PositionsHistoricalRequest]
  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: [TradesHistoricalRequest]
}

type Mutation @rateLimit(limit: 1000, duration: 1) {
  downloadReports(reports: [DownloadReportInput]!): String
  submitPositionsHistoricalRequest(input: DateAsInput!): PositionsHistoricalRequest
  submitTradesHistoricalRequest(input: DateAsInput!): TradesHistoricalRequest

}

type Subscription @rateLimit(limit: 1000, duration: 1) {
  onPositionsFeed: Positions
  onPositionsHistoricalRequestsFeed: PositionsHistoricalRequests
  onReportsFeed: Reports
  onSettlementPositionsFeed: SettlementPositions
  onTradesFeed: Trades
```

```
  onTradesHistoricalRequestsFeed: TradesHistoricalRequests
}

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 {
       accr_int_start_dt
       adjustment_factor
       adt
       asset_type
       cfi_code
       corporate_event
       country
       coupon_freq
       coupon_rate
       crud
       end_valid_days
       exercise_style
       fixed_coupon
       guaranteed_flag
       index_name
       instr_desc
       instr_group_code
       instr_status
       instr_subtype
       instr_trading_code
       instr_type
       instr_unit
       isin
       legal_form
       liquid_indicator
       listing_dt
       main_depository
       market_price
```

```
        maturity_dt
        mic
        minimum_settl_amt
        msg_sequence
        par_value
        qty_type
        redemption_freq
        settl_curcy
        settle_delay
        settle_system
        strike_curncy
        strike_price
        symbol_index
        trade_curncy
      }
}
```

5.1.2 Sample List Instruments Response


```
{
  "data": {
    "listInstruments": [
      {
          "accr_int_start_dt": ,
          "adjustment_factor": ,
          "adt": 1000,
          "asset_type": "E",
          "cfi_code": "ESVUFB",
          "corporate_event": "",
          "country": "BEL",
          "coupon_freq": "",
          "end_valid_days": 4,
          "exercise_style": ,
          "fixed_coupon": "",
          "guaranteed_flag": "1",
          "index_name": "",
          "instr_desc": "BQUE NAT. BELGIQUE",
          "instr_group_code": "A1",
          "instr_status": "1",
          "instr_subtype": "Share",
          "instr_trading_code": "BE0003008019"
          "instr_type": "Share",
          "instr_unit": "1",
          "isin": “BE0003008019”,
          "legal_form": "0",
          "liquid_indicator": "1",
          "listing_dt": 19800717,
```

```
          "main_depository": "00002",
          "market_price": 800.07,
          "maturity_dt": "",
          "mic": "XBRU",
          "minimum_settl_amt": ,
          "msg_sequence": 9152,
          "par_value": ,
          "qty_type": "U",
          "redemption_freq": ,
          "settl_curcy": "EUR",
          "settle_delay": "2",
          "settle_system": "60",
          "strike_curncy": "EUR",
          "strike_price": 0,
          "symbol_index": 1110028,
          "trade_curncy": "EUR",
          "crud": "I",
      }
    ]
}
}
```


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 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 descriptions.

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_type
        qty
        settl_curcy
```

```
        settl_dt
        settle_amt
        settle_per
        settle_ref
        settle_system
        side
        symbol_index
        text
        trade_capacity
        trade_curncy
        transact_tmsp
    }
}
```

5.3.2 Sample List Trades Response


```
{
  "data": {
    "listTrades": [
      {
          "accr_int": 0,
          "acct": "H",
          "client_order_id": 3622428957,
          "counterparty_code": "9",
          "crud": "I",
          "ctv": 572.88,
          "curr_exch_rate": 1.0,
          "exec_id": "B1TXM6Y4IH",
          "execution_type": "1",
          "french_registered_flag": "N",
          "gcm": "2000",
          "guaranteed_flag": "1",
          "isin": "BE0003008019",
          "main_depository": "00002",
          "market_price": 23.87,
          "mbr": "2000",
          "mic": "XBRU",
          "msg_sequence": 9152,
          "order_id": 619777,
          "pos_acct_id": "PA-000003000",
          "position_id": 221020000092,
          "qty_type": "U",
          "qty": 24,
          "settl_curcy": "EUR",
          "settl_dt": 20230717,
          "settle_amt": 572.88,
```

```
          "settle_per": 2,
          "settle_ref": "BX220924AAAA2",
          "settle_system": "60",
          "side": "B",
          "symbol_index": 1110028,
          "text": "RT0123URT",
          "trade_capacity": "1",
          "trade_curncy": "EUR",
          "trade_dt ":20221026,
          "trade_tm": 153321
      }
    ]
  }
}
```


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 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",
  "separator": ";"
}
```

5.4.2 Sample Export Trades Response


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

5.5 List Positions API

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

5.5.1 Sample List Positions Request

```
query ListPositions {
  listPositions {
    msg_sequence
    position_id
    pos_acct_id
    acct
    mbr
    gcm
    margin_acct_id
    french_registered_flag
    isin
    asset_type
    trade_dt
    settl_dt
    end_valid_dt
    side
    qty
    amt
    qty_type
    settl_curcy
    accr_int
    position_source
    settle_ref
    miti
    main_depository
    position_status
    crud
  }
}
```

5.5.2 Sample List Positions Response

```
{
  "data": {
    "listPositions": [
      {
        "msg_sequence": 9185,
```

```
        "position_id": 221020000094,
        "pos_acct_id": "PA-000003000",
        "acct": "C",
        "mbr": "1000",
        "gcm": "2000",
        "margin_acct_id": "1234-1234",
        "french_registered_flag": "N",
        "isin": "BE0003008019",
        "asset_type": "E",
        "trade_dt": 20230502,
        "settl_dt": 20230504,
        "end_valid_dt": 20230508,
        "side": "B",
        "qty": 1278657,
        "amt": 4526878.57,
        "qty_type": "U",
        "settl_curcy": "EUR",
        "accr_int": ,
        "position_source": "ST",
        "settle_ref": "BX220924AAAA2",
        "miti": "12456TTRR",
        "main_depository": "00002",
        "position_status": "LIVE",
        "crud": "I"
      }
    ]
  }
}
```


5.6 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 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.

5.6.1 Sample Export Positions Request

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

```
}
```

5.6.2 Sample Export Positions Response

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

5.7 List Settlement Positions API

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

5.7.1 Sample List Settlement Positions Request

```
query ListSettlementPositions {
  listSettlementPositions {
    msg_sequence
    gcm
    agent
    delivery_acct_id
    csd_settle_acct
    fail_acct
    isin
    symbol_index
    side
    trade_dt
    settle_system
    main_depository
    settl_curcy
    settl_dt
    end_valid_dt
    buy_in_dt
    qty
    qty_type
    amt
    unsettled_qty
    usettled_amt
    settle_status
    settle_source
    market_venue
    french_registered_flag
    hold_indicator
```

```
    settle_ref
    previous_settle_ref
    miti
    crud
  }
}
```

5.7.2 Sample List Settlement Positions Response

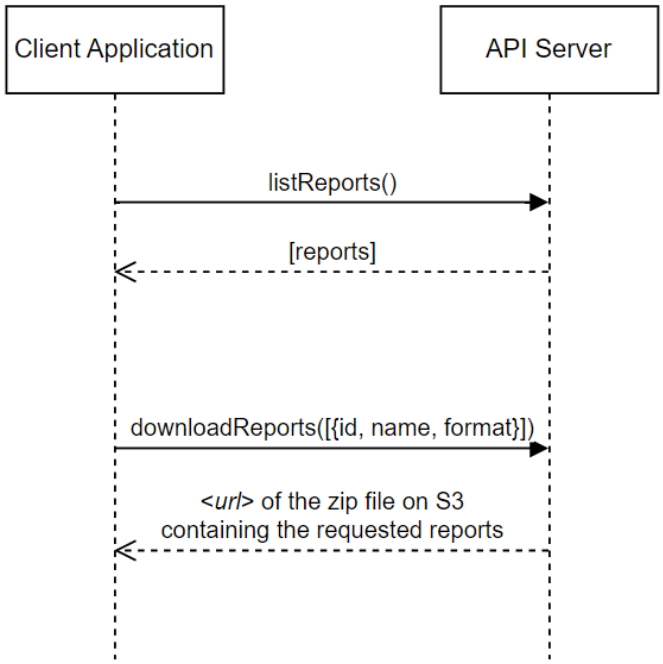

```
{
  "data": {
    "listSettlementPositions": [
      {
        "msg_sequence": 2854,
        "gcm": "2000",
        "agent": "5000",
        "delivery_acct_id": "DA-159875",
        "csd_settle_acct": "SECURITYACCOUNTEXAEUROCLEAR",
        "fail_acct": "",
        "isin": "BE0003008019",
        "symbol_index": 1110028,
        "side": "B",
        "trade_dt": 20230711,
        "settle_system": "60",
        "main_depository": "00002",
        "settl_curcy": "EUR",
        "settl_dt": 20230713,
        "end_valid_dt": 20230719,
        "buy_in_dt": 20230724,
        "qty": 3489779,
        "qty_type": "U",
        "amt": 2090997,
        "unsettled_qty":  0,
        "unsettled_amt":  0,
        "settle_status": "",
        "settle_source": "T",
        "market_venue": "XBRU",
        "french_registered_flag": "N",
        "hold_indicator": "N",
        "settle_ref": "BX220924AAAA2",
        "previous_settle_ref": "",
        "miti": "12456TTRR",
        "crud": "I"
      }
    ]
  }
}
```

5.8 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:



5.8.1 Sample List Reports Request

```
query ListReports {
  listReports {
    msg_sequence
    report_id
    mbr
    gcm
    agent
    report_tmsp
    expiration_dt
    report_name
    report_code
    report_version
    report_format
```

```
    report_status
    added_tmsp
    crud
    restore_request_tmsp
  }
}
```

5.8.2 Sample List Reports Response


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

6. MUTATIONS

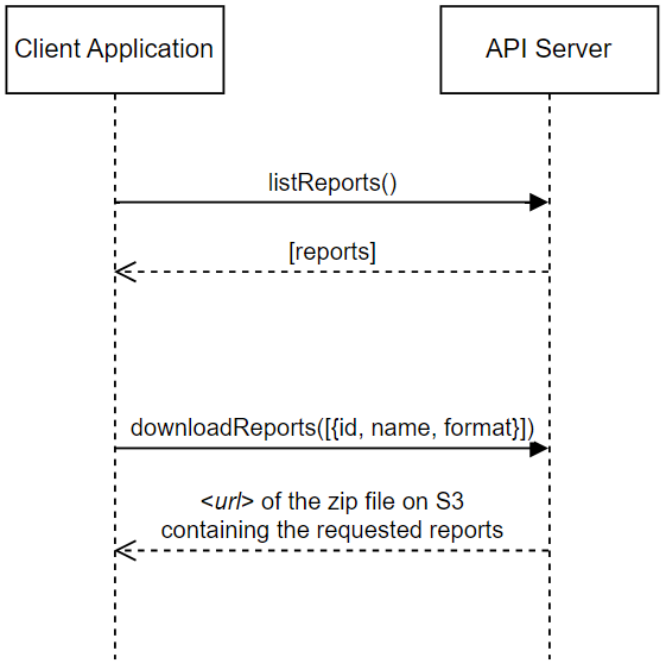
This section focuses on the mutations that can be executed via API. Mutations are used to perform operational requests (e.g. posting amendments).

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 5.8.

The diagram below shows the flow that the client applications must implement to download 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 5.5.

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": 4526878.57,
          "asset_type": "E",
          "end_valid_dt": 20230508,
          "gcm": "2000",
          "isin": "BE0003008019",
          "margin_acct_id": "1234-1234",
          "mbr": "1000",
          "miti": "12456TTRR",
          "pos_acct_id": "PA-000003000",
          "position_id": 221020000094,
          "qty_type": "U",
          "qty": 1278657,
          "settl_curcy": "EUR",
          "settl_dt": 20230504,
          "settle_ref": "BX220924AAAA2",
          "side": "B",
          "trade_dt": 20230502
        }
      ]
    }
  }
}
```


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"
        }
      ]
    }
  }
}
```

8. LIST OF ATTRIBUTES

Field name	Description
accr_int	Accrued Interest.
accr_int_start_dt	Accrued Interest Start Date.
acct	Account Type.
added_tmsp	Added Timestamp.
adjustment_factor	Adjustment Factor.
adt	Average Daily Turnover.
agent	Settlement Agent.
amt	Amount.
asset_type	Asset Type.
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.
cash_settl_dt	Cash Settlement Date. This date is available only on failed positions and represents the eventual date.
cash_settl_status	Cash Settlement Status. Indicates whether the position is flagged for a cash settlement.
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.
client_order_id	Client Order ID.
closing_price	Last adjusted closing price of the instrument.
corporate_action_fraction	Fractional quantity remaining as a rest after a Corporate Action Transformation.
corporate_event	Corporate Event Type.
corporate_event_indicator	Corporate event indicator. Indicates whether the position is under corporate event.
corporate_msg_ref	Corporate message reference. Reference to the transformation/claim instruction.
counterparty_code	Counterparty Code.
country	Country.
coupon_freq	Coupon Frequency.
coupon_rate	Coupon Rate.

created_at_tmsp	Timestamp of the request/position/settlement position creation.
crud	CRUD (Create Read Update Delete), field used to track the operation executed on the record.
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.
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 Clearing.
delivery_position_id	Delivery Position ID, identifier of one specific settlement position.
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.
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.
exercise_style	Exercise Style .
expiration_dt	Expiration Date, last day of the availability of a report.
fail_acct	Fail Account.
fixed_coupon	Fixed Coupon, indicates if the rate of the coupon is fixed or variable
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

gcm	Clearing Member Code, internal to Euronext Clearing
guaranteed_flag	Guaranteed Indicator, specifies if the trade or the instrument is guaranteed or not
haircut	Haircut applied to the position (in percentage terms) as a consequence of the exchange rate conversion. It is zero in case Settlement Currency and Clearing currency are equal.
hold_indicator	Hold Indicator, indicates the status of the instruction.
index_name	Index Name.
instr_desc	Instrument Description.
instr_group_code	Instrument Group Code.
instr_status	Instrument Status.
instr_subtype	Instrument Subtype.
instr_trading_code	Instrument Trading Code.
instr_type	Instrument Type.
instr_unit	Instrument Unit.
isin	ISIN.
legal_form	Legal Form.
liquid_indicator	Liquidity Indicator, indicates whether the instrument is liquid or not, as defined per MiFID II.
listing_dt	Listing Date, first day of trading for the instrument.
main_depository	Main Depository.
margin_acct_id	Margin Account ID, internal to Euronext Clearing.
market_price	Price.
market_venue	Market Venue.
matching_status	Matching Status. Indicates whether the SI is already matched or to be matched.
maturity_dt	Maturity Date of the Instrument, when relevant.
mbr	Member Code, internal to Euronext Clearing.
mic	MIC.
minimum_settl_amt	Minimum Settlement Amount.
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.
netting_rule	Indicates the netting rule applied.
order_id	Order ID.
par_value	Par Value.
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.
pos_acct_id	Position Account ID, internal to Euronext Clearing.
position_id	Position ID.
position_source	Position Source.
position_status	Position Status.
position_type	Position Type.
previous_settl_ref	Previous Settlement Reference.
qty	Quantity.
qty_type	Quantity type.
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).
redemption_freq	Redemption Frequency.
report_code	Report Description.
report_description	Report Description.
report_format	Report Format.
report_id	Report ID.
report_name	Report Name.
report_status	Report Status.
report_tmsp	Report Timestamp.
report_version	Report Version.
restore_request_tmsp	Restore Request Timestamp.
settle_amt	Settlement Amount.
settl_curcy	Settlement Currency.
settle_delay	Settlement Delay.
settl_dt	Settlement Date.
settle_per	Settlement Period.
settle_ref	Settlement Reference.
settle_status	Settlement Status.

settle_source	Settlement Source.
settle_system	Settlement System.
side	Side.
status	Status of the operational request.
strike_curncy	Strike Currency.
strike_price	Strike Price.
symbol_index	Symbol Index.
text	Text, sent by the Client when inserting the order.
trade_capacity	Trading Capacity
trade_curncy	Trading Currency
trade_dt	Trade Date
trade_tm	Trade Time
transformation_fraction	
unsettled_amt	Unsettled Amount
unsettled_qty	Unsettled Quantity
user_id	User that has requested the operation

9. STATIC VALUES

Field	Enum Values
acct	'C': 'Client' 'H': 'House' 'L': 'Liquidity Provider'
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'
partial_settle_status	'PARC: 'Partially Confirmed' 'PAIN': 'Partial Settlement '
position_source	'ST': 'Trading' 'CA': 'Corporate Action' 'BP': 'Buyer Protection'
position_status	'LIVE': 'Open' 'CANS': 'Cancelled by CSD'
position_type	'S': 'Active' 'F': 'Failed'
qty_type	'F': 'Face Value' 'U': 'Unit'
report_status	'A': 'Available' 'R': 'Restoring' 'E': 'Error' 'N': 'Not Available'
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' '01': 'Euroclear Bank' '1': ' T2S' '2': '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).