

ETS API Client Application Design Guide

16.05.2022
Paris
V7.37

Table of Contents

1.	Introduction	5
1.1	Audience	5
1.2	Purpose	5
1.3	Changes History Table	5
2.	ETS API Package presentation	6
3.	Day-Ahead Market Basics	8
3.1	Website References	8
3.2	API Package documents References	8
3.2.1	Static Reference Data	8
3.2.2	Block Orders	8
4.	ETS API overview	9
5.	Zoom on Functional areas	12
5.1	API user Password management	12
5.1.1	Passwords life cycle overview	12

5.1.2	Reset Password details	13
5.1.3	Password policy	15
5.2	Trading Limits (TL)	15
5.2.1	Trading Limits fundamentals	15
5.2.2	Trading Limits life cycle	17
5.2.3	Trading Limits API retrieval policy	17
5.2.4	Trading Limits setup in ETS for ISVs in Test	17
5.2.5	Trading Limits requests	17
6.	Architecture	19
7.	ETS API schema and WSDL file	20
8.	An exclusively Synchronous System Messaging	22
8.1	Synchronous mode	22
8.2	Messages structure	23
9.	Open Access Login	24
10.	Security and certificates	27
11.	Warning and Error Management	28
11.1	Technical errors	28
11.2	Functional errors and warnings	29
12.	Timeouts and recovery procedure	31
13.	Typical structure of an ETS API application	32
	ETS API attention points	34
14.1	How to properly identify an auction in request messages?	34

14.2	Deprecated messages or tags	35
14.3	Order entry and ETS Portfolio set up	35
14.4	REMIT Trading on Behalf data: Optional Beneficiary and Trading Capacity (A,P) attributes:	36
15.	Java and Python Client Examples	37
16.	Client development guidelines	38
17.	API Conformance Test	39
17.1	API Conformance Test milestone	39
17.2	API Conformance Test operational steps	39
18.	Use Cases	42
18.1	Normal Day	42
18.2	DST 23 – Summer	43
18.3	DST 25 – Winter	43

1. Introduction

1.1 Audience

This Client API Application Design Guide is meant for:

- developers writing client applications to communicate with the ETS API ("Open Access" Server),
- any other profile like project managers who would like to understand the API fundamentals.

1.2 Purpose

The purpose of this document is to describe the principles of the ETS API, giving enough elements and guidelines:

- to understand the ETS API fundamentals
- to build an efficient client application that will pass the ETS API Conformance Test before going live.

Many references are made to the "Terms of Reference" document: we recommend to read this document as a priority:

- to understand the key policies and rules any API client application must comply with,
- to be aware of the points that will be checked during the ETS API conformance test.

1.3 Changes History Table

Date	Version	Change description
18.12.2020	V7.26	Document redesign to take 2019 customers' feedback into account.
18.05.2020	V7.31	Update for ETS 3.4
15.06.2020	V7.32	API schema 3.4.6, replacing the previous 3.4.2 version: <ul style="list-style-type: none"> - New 3.4 WSDL - Trading Limit update: new input and output parameters for the <i>RetrieveCentralCounterPartyAndSettlementMemberNames</i> method as announced in the initial 3.4 ETS API communication
18.01.2021	V7.33	<ul style="list-style-type: none"> - Precisions about the value to use for the Synchronous mode (values to use + <code><responseToken></code> tag not in use). - Precisions about the different possible response states (ACK, NAK, NOT FOUND, ERROR OCCURRED) and their interpretation.
03.02.2021	V7.34	- Security changes postponed from 3.5 to 3.6 (section §10)
22.03.2021	V7.35	- Precisions about the API conformance test operational steps (section §17)
01.02.2022	V1.36	- API Conformance Tests detailed procedure
21.03.2022	V1.37	- 3.6.1. schema introduction
06.10.2023	V1.38	<ul style="list-style-type: none"> - 3.7.2 API schema impacts: <ul style="list-style-type: none"> o 3.7.2 trading limit changes. Exposure broken down by area / portfolio / order type (linear, block). o Trading on behalf option (Capacity trading + Beneficiary fields)

2. ETS API Package presentation

In the ETS API package you will find the following content:

Folder / Document	Description
1-API specifications	
1. The Client API application Design Guide	(this document) Describes high level design fundamentals of the ETS API and how to design your own API client application.
2. ETS API Process for customers	This Excel sheet contains 2 tabs: <ul style="list-style-type: none"> - A diagram showing all the steps from the initial interest in the ETS API until go live, including API Conformance Tests - The details of each of these steps
3. ETS API Terms of Reference	Describes the expected behavior of the application using the API Public message interface of ETS, mainly: <ul style="list-style-type: none"> - Login/Logout policy - Market results retrieval policy
4. ETS API Certificates	<ul style="list-style-type: none"> - Technical information about certificates needed to connect to the ETS API - Certificate management process - Process to obtain a certificate.
5. Environments Detail Description	<ul style="list-style-type: none"> - Simulation and Production ETS Client and API environments details (specific to SEMOpx)
6. WSDL and its description	<ul style="list-style-type: none"> - The WSDL file which features all the supported API schemas: ETSOpenAccessWebService - Its documentation version: ETSOpenAccessWebServiceDoc
7. ETS Client and ETS API - Market Results Description	<ul style="list-style-type: none"> - This document describes the content of the market results report (at a portfolio/area aggregated level), featured in the response to the method RetrieveMarketResultsFor (CSV report embedded in the XML response). - As of ETS 3.3.2 the RetrieveTradesReportFor method enables to retrieve trades at a trade Id level.
8. ETS API Static Reference Data v1.0	List of Auction Name, Market Area, Area Set and Areas which are necessary to build API requests.
9. List of ETS API Deprecated tags	<ul style="list-style-type: none"> - Deprecated tags are old tags that SEMOpx still supports but only for backward compatibility reasons. - They will be decommissioned as of ETS API 3.4. - See the dedicated section below for more details.
10-Block Orders	<ul style="list-style-type: none"> - Functional description of block changes introduces with ETS 3.2

2-Sample Requests	
Sample API Requests and Responses	A list of common sample requests and responses for: <ul style="list-style-type: none"> - normal days - DST23 (Summer) and DST25 (Winter) days.
3-Sample Code	
Sample Java and Python code	A short sample code illustrating what are the key elements to know (login, session token, header data) for an efficient start with the ETS API.
4-Support	
1. ETS API FAQ	<ul style="list-style-type: none"> - Functional questions - Technical questions - Procedural questions: user credentials, access to doc, etc.
2. ETS API Issues - Support Checklist	<p>This is the file to be filled in and sent to SEMOpx when you experience an issue, in simulation or production.</p> <ul style="list-style-type: none"> - This is key to fasten the processing of your request.
3. SOAP UI Certificate installation guide	<p>How to import a certificate in SOAP UI.</p> <p>SOAP UI can be used to test requests and responses, but as well to reproduce potential issues you might encounter, using a “neutral” tool (removing the potential complexity of your application).</p>
5-Conformance Test	
1. ETS API Client App information	<p>This file enables us to collect key inputs about your API application.</p> <p>Please address this file to market operations marketops@ops.semopx.com) prior to any API conformance test.</p>
2. ETS API Acceptance Matrix	<p>This file gives you some guidance about the mandatory or optional requests to be tested during conformance tests.</p> <p>Please address it to market operations prior to any API conformance test.</p>

3. Day-Ahead Market Basics

Please find below a few links and documents references giving to the reader important basics about the Exchange markets concepts in play to understand ETS concepts in play with the ETS API.

3.1 Website References

You can have an overview of the Day-Ahead products and market in this Website section:

>TRADING & SERVICES > Trading Products>Day-Ahead Trading.

In the Website download section (once logged in the website as a member):

- *Download>Trading Products>Day-Ahead* : different documents listing the **auctions parameters**
- *Download>Rules, Fees & Processes> Day-Ahead processes: **Day-Ahead Timings & MM***: you can find along with the normal process all the MRC (Multi-Regional Coupling) delay and decoupling scenarios and the related ETS messages code references to take them into account in your API application.

3.2 API Package documents References

3.2.1 Static Reference Data

Please Refer to the API Package document **SEMOPX ETS API Static Reference Data**.

It describes for your exchange the list of Area Sets, Market Areas and Areas, and how they are related.

3.2.2 Block Orders

Please Refer to the API Package document **Block Orders.pdf**.

4. ETS API overview

Basically the ETS API is Web Service based, using SOAP 1.1 (Simple Object Access Protocol) over HTTPS.

- Requests and responses format is XML,
- Security wise it uses the TLSv1.2 protocol,
- It is a synchronous API, using requests/responses,
- There is no push/broadcast functionality: periodic inquiry requests must be used.
- Rules about inquiry requests periodicity are described in the “Terms of Reference” document.

Non-Market Participants (NMP) and Market Participants (including Independent Software Vendors “ISVs” in test environments) can access the below methods, for read-only and read-write applications.

Please consult each method comments in the WSDL file and use sample requests for further details.

Purpose	Web Method (*) : available for Non-Market Participants	Description
Login	EstablishConnection (*)	<p>Login / Logout via an ETS API server which creates a connection to the ETS trading back-end (like any ETS client connection).</p> <p>The login (“<i>EstablishConnection</i>”) response contains a session token (User Login Name + Session Key) that the API client must re-use for each called method/request, and so during the whole trading day until invalid.</p> <p><i>Please consult the Login/Logout policy in the Terms of Reference document</i></p>
Logout	Logout (*)	<p>Optional, for GUIs or if your company security policy requires it.</p> <p>Please see <i>Terms of Reference</i> for more details.</p>
Change or Reset your API user password	SetNewPassword (*) As of schema 3.4.6: <ul style="list-style-type: none"> • ResetPasswordRequestActivationCode (*) • ResetPassword (*) • UpdatePassword replaces SetNewPassword (*) 	<p>You have 7 days to change the initial password provided by Market ops.</p> <p>Changing a password automatically extends its validity to 90 days.</p> <p>You can use the API to request an API password reset or directly Update the password.</p> <p>Please refer to the “Zoom on functional areas” section for more details.</p>
Know Auctions characteristics	RetrieveAuctionInformation (*)	<p>For instance, retrieve the Auction theoretical publication time, to know when to start retrieving market results.</p>
Retrieve market results (for up to 14 days in the past)	RetrieveMarketResultsFor (*)	<p>Enables to retrieve market results at a portfolio aggregated level, for one or several areas.</p> <p>Results are in csv format embedded in XML.</p> <p><i>Please consult:</i></p> <ul style="list-style-type: none"> - the <i>Market results Retrieval policy in the Terms of Reference document</i> - the “<i>ETS Client and ETS API - Market Results Description</i>” document

	As of the API 3.3.2 schema RetrieveTradesReportFor (*)	Enables to retrieve pure XML market results. For market participants results are given at a trade level (with trade ID, trader ID, etc.). Please refer to the dedicated document describing the format of this report in the API package. Please note that market results are different for Market Participants and Non Market Participants (only public indexes).
Be aware of auction specific events	RetrieveMessagesUnreadOnly (*) SetMessagesAsRead (*)	You application is supposed to consume these message that give important information in case of maintenance info, decoupling, delay in market results publication, password expiry to come etc.
Know your areas and portfolio assignments and information	RetrieveViewableAreas (*) RetrieveAreaInformation RetrieveAreaPortfolioInformation RetrieveViewablePortfolios RetrieveViewableAreaPortfolioInformation RetrieveTradableAreas RetrieveTradableAreasets RetrieveTradablePortfolios RetrieveTradableAreaPortfolioInformation	If you want to collect dynamically the list of areas your API user is assigned to.
To ensure the API server can be reached	Keep Alive (*)	For testing purposes only (useful during initial connectivity tests to ensure the technical connection with the certificate to our API server is established). Does not require to be logged in (no valid session token required).
Retrieve your trading limit	As of the 3.4.6 API schema: RetrieveCentralCounterPartyAndSettlementMemberNames RetrieveTradingLimits RetrieveTradingLimitExtendedDetails	Your application can: - retrieve the initial trading limit for a given trading limit day - monitor the current consumption of the trade limit - retrieve all the setup limits Please refer to the "Zoom on functional areas" section for more details.

Read-Write applications (Market Participants and ISVs in test) can access the following main methods:

Purpose	Method	Description
Manage orders	EnterOrder, EnterBlockOrder, EnterBlockOrderBatch, EnterComplexOrder. Cancel related methods	Manage orders (enter, modify, cancel) – only for market participants <ul style="list-style-type: none"> - methods are specialized by order type (linear, blocks, complex) - several blocks can be entered at once Please note that order modification is done via the same “EnterOrder” methods as for order entry.
Retrieve Orders	RetrieveOrders, RetrieveBlockOrders, RetrieveComplexOrders, RetrieveExclusiveGroupWithId, RetrieveGroupForBlockId, RetrieveLoopBlocksWithGroupId, RetrieveSmartBlockOrders	<p>This enables to retrieve all orders of your portfolios:</p> <ul style="list-style-type: none"> - to check the latest version - when the server is busy (see Terms of Reference) to ensure of whether or not your request was processed. <p>Since there is no push functionality, this enables you in particular to build a consolidated vision of orders : retrieve all orders, potentially submitted by the ETS client and/or any other API application (for instance if you are trading on top/instead of the ETS client through an ISV application).</p>

5. Zoom on Functional areas

5.1 API user Password management

5.1.1 Passwords life cycle overview

Please find below a description of the different actions to take at each step of your API user password life cycle.

STEP 1	<p><u>Extend the initial password validity</u></p> <p>You just received your API user(s) and password(s) from Market Operations:</p> <ul style="list-style-type: none"> • You have 7 days to change the initial temporary password. • This will automatically extends its validity to 90 days. • You have to store the new validity date to be able to change again the password before it expires. <ul style="list-style-type: none"> ○ There is a safety net if you could not store it: please see the expiry notification 7 days in advance as described in STEP 2. ○ but please note there is no alternative way to retrieve an API user password validity date during its lifetime. <p>For API customers this operation <u>can only be done using API messages</u>: since in the very beginning you do not have any API application, please use a tool like SOAP UI and use the UpdatePassword (*) sample request to change your password as soon as you receive it from market operations.</p> <p>(*) as of the API 3.4 schema SetNewPassword becomes obsolete and must be replaced by UpdatePassword. The SetNewPassword method is still technically available in the 3.4.6 schema and be sent, but will not change any password when being processed. Instead an error message will be sent back:</p> <pre><SOAP-ENV:Body> <ns:SetNewPasswordResponse> <SetNewPasswordAcknowledgement> <ns:state>NAK</ns:state> <ns:error> <ns:errorId>OA 217</ns:errorId> <ns:errorMessage>Obsolete message. Use UpdatePassword.</ns:errorMessage> </ns:error> <ns:passwordInformation>Password not changed. See errors for further details</ns:passwordInformation> </SetNewPasswordAcknowledgement> </ns:SetNewPasswordResponse> </SOAP-ENV:Body></pre> <p><u>Update password</u> (as of the 3.4.x API schema only; for prior schemas please use SetNewPassword)</p> <p>A new UpdatePassword API method is introduced, requiring the old password as an input parameters:</p> <ul style="list-style-type: none"> ○ Inputs: sessionKey, userLoginName, old Password, new password ○ Output: status (success or failure with error description), passwordInformation (e.g. "Password changed. Valid until: 2020-06-22") <p>Though market ops can reset or extend a password this option should be kept only for emergencies: regular changes should be handled by the API application itself.</p>
STEP 2	<p><u>Anticipate passwords expiration by processing ETS API notifications (messages)</u></p> <p>ETS will generate a message announcing your password expiration 7 days in advance, retrievable via the RetrieveMessagesUnreadOnly method.</p> <p>Example: (see as well sample messages in the package)</p> <pre><SOAP-ENV:Body> <ns:RetrieveMessagesUnreadOnlyResponse> <RetrieveMessagesAcknowledgement> <ns:state>ACK</ns:state> <ns:messages> <ns:messageId>692482</ns:messageId> <ns:subject>Password expiration</ns:subject> <ns:time>2019-10-30T13:25:19+01:00</ns:time> <ns:text>Your password will expire in 7 days. Please change your password before November 6, 2019</ns:text> <ns:read>false</ns:read> </ns:messages> </RetrieveMessagesAcknowledgement> </ns:RetrieveMessagesUnreadOnlyResponse> </SOAP-ENV:Body></pre>

STEP 3	<p><u>Update the password before it expires</u></p> <p>Whether you stored the password validity date when you changed it, or whether you reacted to the notification described in STEP 3, you can update the password as explained at step 1. The new password will be valid for 90 days. Please store again the new validity date sent in the UpdatePassword response.</p>
STEP 4	<p><u>The password expired or is revoked : Password reset</u></p> <p>If the API user password expiry date is reached or if the password is revoked (after 3 attempts with an incorrect password) the only way to be able to log in again (EstablishConnection) is to reset the password.</p> <p>a) As of the 3.4.x schema:</p> <p>This should preferably be performed by the API application itself (keeping the Market Operations call possibility for emergencies), using the following methods, as described below:</p> <ul style="list-style-type: none"> • ResetPasswordRequestActivationCode • ResetPassword <p>The new password will be valid for 90 days. Please store again the new validity date.</p> <p>b) If you are using a lower API schema (3.x): the only solution to reset a password is to contact Market Operations.</p>

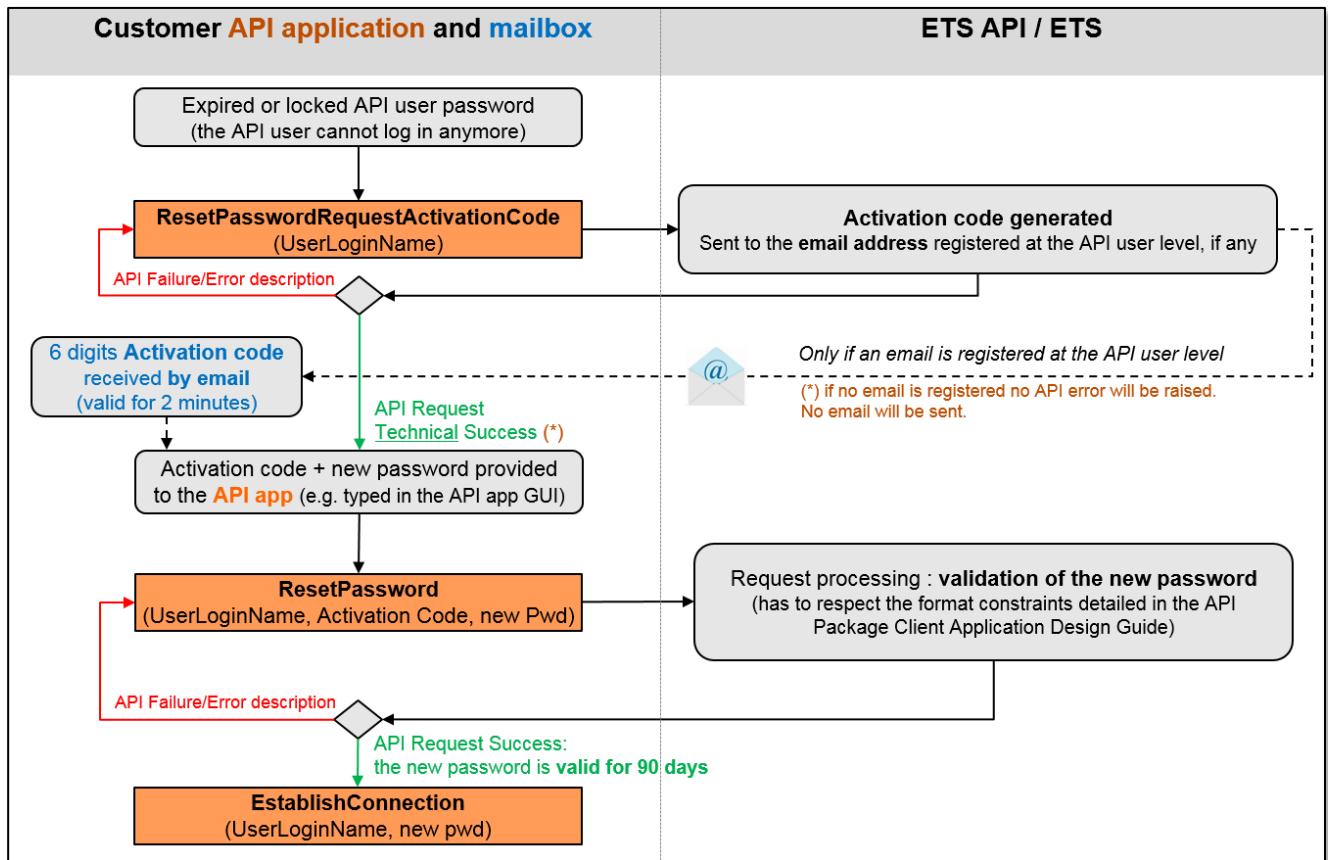
5.1.2 Reset Password details

As of [schema 3.4.6](#) the API offers new requests to ask an API user **Password Reset** directly :

- [ResetPasswordRequestActivationCode](#)
- [ResetPassword](#)

Until ETS API 3.3 the only way to get your API user password reset once expired or locked is to contact Market operations. The 3.4.6 schema along with ETS 3.4.6 enables you to implement a *Reset API user Password* functionality directly from your API application, without having to get in touch with Market operations.

Prerequisite: for the email mechanism described below to work it is required that SEMOPX registered an email address for your API user in ETS.



The API Password Reset process to be implemented is as follows:

- **Step 1 : [ResetPasswordRequestActivationCode](#)**: ask ETS to send you an Activation Code for a specific API user at the email address you chose to link to your API user (a dedicated communication will be sent to collect you choice). **Please note that this request should be used when not logged in.**

ResetPasswordRequestActivationCode	
Inputs	userLoginName
Outputs	Status (success or failure with error description)

- **Step 2:** if an email address is registered at the API user level you receive the [Activation Code \(valid for 2 minutes\)](#).
- **Step 3: [ResetPassword](#)** : you send the received activation code along with your chosen new password. Please note that this request does not require being already Logged in.

ResetPassword	
Inputs	<ul style="list-style-type: none"> • userLoginName • activationCode (the activation code received by email) • a password (the new password)
Outputs	<ul style="list-style-type: none"> • Status (success or failure with error description) : your request will be rejected, if your new password does not respect the password policy described below or the activation code has expired. • passwordInformation

- **Step 4:** you have to log in with your new password ;this new password is valid for 90 days.

5.1.3 Password policy

Here are the requirements an API user password has to comply with:

1. Must contain a minimum of 8 characters
2. Must include 3 out of 4 of symbol, number, uppercase letter and lowercase letter
 - a. Symbol examples: '\$', '&', '~', ']'
 - b. Number examples: '1', '7', '0'
 - c. Uppercase examples: 'E', 'T', 'S'
 - d. Lowercase examples: 'e', 't', 's'
3. May not include composed characters
 - a. Composed examples: 'é', 'â', 'ê'
4. May include a space, but this counts as a symbol
5. May only repeat each character once in sequence, i.e. each character can be doubled, but not tripled in a row

5.2 Trading Limits (TL)

5.2.1 Trading Limits fundamentals

Context: Trading limits have been implemented in ETS since the very beginning. The ETS API has always taken them into account when an API application would perform an order management action ((TL 001) Trading Limits Breached error), and did send an error message for instance when an order exceeding the current TL was submitted. What is new as of the 3.4.6 API schema is that API applications can now retrieve these TLs, either to monitor it directly or to just display it for the ones have a GUI.

Definition:

- Generally, a **trading - or trade - limit (TL)** is defined as a **restriction applied to the ordering/trading activity** of a market participant reflecting its **physical or financial limitations or based on risk management considerations** of the supporting Clearing Member.
- In particular, **financial - or monetary - trading limits for the auction system** consist of an **a priori control on incoming orders against a cash amount** which limits the **financial exposure** of a Trading Participant (TP).
- Financial trade limits apply to all products traded in ETS for a given exchange. In particular day ahead auctions and intraday auctions are covered by the same trade limits.

Objective: Trading limits enable to **control the financial exposure of members**.

Involved parties:

- **ECC** acts as a **central counterparty (CCP)** which places itself between the seller and the buyer. **ECC and its Clearing Members assume the trading participants' counterparty risk** and exploit the entire netting and cross-margining potential in the course of settlement.
- A **Clearing Member (e.g. Clearing Banks)** is a guarantor and payment agent for all trades concluded at the Spot Market. A CM "clears" transactions.
- **To each Trading Participant / Member corresponds one or several Settlement Members** (subdivisions of the Member trading activity e.g. MemberA_1_SM, MemberA_2_SM, e.g. for Green Power and Non-Green Power, or for Production and Supply).
- **Basic Process:**
 - **Settlement Members are customers of Clearing Members and negotiate one or several TL** with them, as well as the trading limit structure/granularity (optional scope for certain area/portfolio).

- **Clearing Members set trading limits** for their respective clients (settlement members) **in the ECC system** using an application provided by ECC.
- **ECC** forwards the list of active TL to SEMOPX.
- **SEMOPX integrates the list of TL received from ECC in ETS** which performs the trade limit checks on submitted orders.
- **Members** are aware of their Settlement Members TL and bid accordingly.
- **TL Versions:** in case of a TL change a new version of the TL is created and the new version integrated into ETS.

TL Granularity:

- Settlement Members can ask their clearing members to have a TL defined:
 - at the settlement member level (without any portfolio or area consideration)
 - OR at the area/portfolio level
- In other words:
 - Several TL can be defined for a given Settlement Member in order to cope with specific configuration situations.
 - Each TL covers a set of areas and portfolios.
- Note 1: a TL is never defined at the level of a trader or user.
- Note 2: though limits are actually defined at the Settlement Member level it will be common to talk about “Member trading limits”.

Scope:

- Orders in Trading Portfolios resulting from physical fulfilment of futures are excluded from limit checking.
- **Trading limits are distinct between SEMOPX and other exchanges.**
- **The Current value of the TL** is initialized with the TL initial value and **is defined for a given auction day starting at 16:00 CET**. It means **the trading limit amount should be sufficient to cover the exposure of the auctions run from 16:00 CET to 16:00 CET the next day.**
- **In case an auction is postponed after 16:00 CET, the initial trade limit is still considered for the auction.**

Currencies:

- The trade limits support **multi-currency** management. It means a trade limit can cover several delivery areas with different currencies.
- The possible currencies to define a trade limit are defined by ECC.

TL key concepts and calculation:

1. TL initial value = value agreed with your CM. It corresponds to the agreed Maximum Exposure.
2. Exposure = current cumulated Exposure = current cumulated exposure of submitted orders for all the auctions of the exchange (unless specific area/portfolio restrictions are in place)
3. **Current TL value = TL initial value – current cumulated exposure**
4. TL Min. / Max. Realistic Price: defines a price corridor.
 - Bids outside this corridor are allowed, but the resulting additional exposure is not covered by the limit.
5. TL Min. / Max price:
 - TL Pmin and Pmax are defined at TL level, and block the trader from:
 - Selling volumes when price < TL Pmin
 - buying volumes when prices > TL Pmax

TL recalculation impact on orders:

- if at any TL recalculation moment (e.g. TL upload, auction run, auction cancellation, FX rate update) the current TL becomes negative, then a subset of active open orders are deleted to bring the TL current value positive again (enough orders to restore a positive value).

5.2.2 Trading Limits life cycle

Several events types can lead to a change of Trading Limits for a given member in ETS:

1. **After an upload in ETS** (of the first TL version or a subsequent version): Member Trading Limits are potentially updated several times a day in ETS (*) which vary between business days and non business days.
2. **After an order management action done by any trader user (in the ETS client or via the API) of the member**
3. **After an auction Market results publication**
 - Before the auction is run a “worse case” execution scenario is taken into account in the TL calculation. Once the auction is being run and the executed quantities determined, the risk is lifted and the TL is recalculated.

5.2.3 Trading Limits API retrieval policy

As indicated in our *Terms of Reference* document, your API application can send the required set of TL requests to refresh its knowledge of all TL information at different moments:

- **periodically**: as mentioned in the Terms of Reference document, **every 3 seconds maximum** (meaning there should be at least 3 seconds between 2 sets of TL requests) ; the periodicity should be configurable (so that each API app can easily change it on request of SEMOpX if technically required to remain flexible).
- **AND after each order management action** (if the API app needs a refresh before the next periodic TL refresh)
- **AND after each Market Results publication** (if the API app needs a refresh before the next periodic TL refresh)
- **AND “on demand”** (not periodic), for instance:
 - For API apps with a GUI:
 - To populate a TL screen (initial data retrieval/display)
 - On demand of an end user (e.g. potential forced manual TL screen refresh)

5.2.4 Trading Limits setup in ETS for ISVs in Test

For ISVs willing to implement a Trading limits retrieval in their API app:

- please note that by default only members have a trading limit defined in ETS **Test** environments, ISVs having no trading limits setup, which functionally means an “infinite limit”.
- This means you need to contact Market operations to get an actual TL set up if interested.

5.2.5 Trading Limits requests

Three requests enable to retrieve the TL information you need:

- [RetrieveCentralCounterPartyAndSettlementMemberNames \(*\)](#):
 - This request enables to retrieve the CCP (=ECC) and the different Settlement Member (SM) names related to the connected API user member, to potentially reuse them in the [RetrieveTradingLimits](#) request/response content (as a input filter, or as a reference to interpret the SM names in the response).
 - Members having the possibility to access other member portfolios (specific configuration) can retrieve TL info/restrictions at the right level
 - New optional input parameters enable to filter the response by:
 - Market participant (member)
 - Area
 - Portfolio
 - members having TL restrictions by portfolio can get the details of these restrictions from the API
- [RetrieveTradingLimits](#):
 - enables to request the TL(s) for a given Trading Limit Day (mandatory input), with several additional optional filters (SM, area, portfolio, etc.)

- the API responds with the list of all TLs covering the related (area/Portfolio) combinations for which the user has Read or Read/Write permissions, with (among other) information regarding:
 - the TL ID and its version,
 - the exposure,
 - the initial and current TL value,
 - the TL min/max prices,
 - the realistic min/max prices.
- *RetrieveTradingLimitExtendedDetails:*
 - Enables to request the details of a specific TL identified by its ID and version
 - The API responds with the list all (area/ Portfolio) combinations linked to a specific TL ID/version, as well as the period for which the TL applies.

Please check the API package sample requests to illustrate this.

6. Architecture

Customer applications access the Open Access API via a web service that runs on the SEMOpx network. The following diagram shows one or more Open Access servers running on the SEMOpx network and the client applications using HTTPS web transport protocol to connect to the Open Access Server.

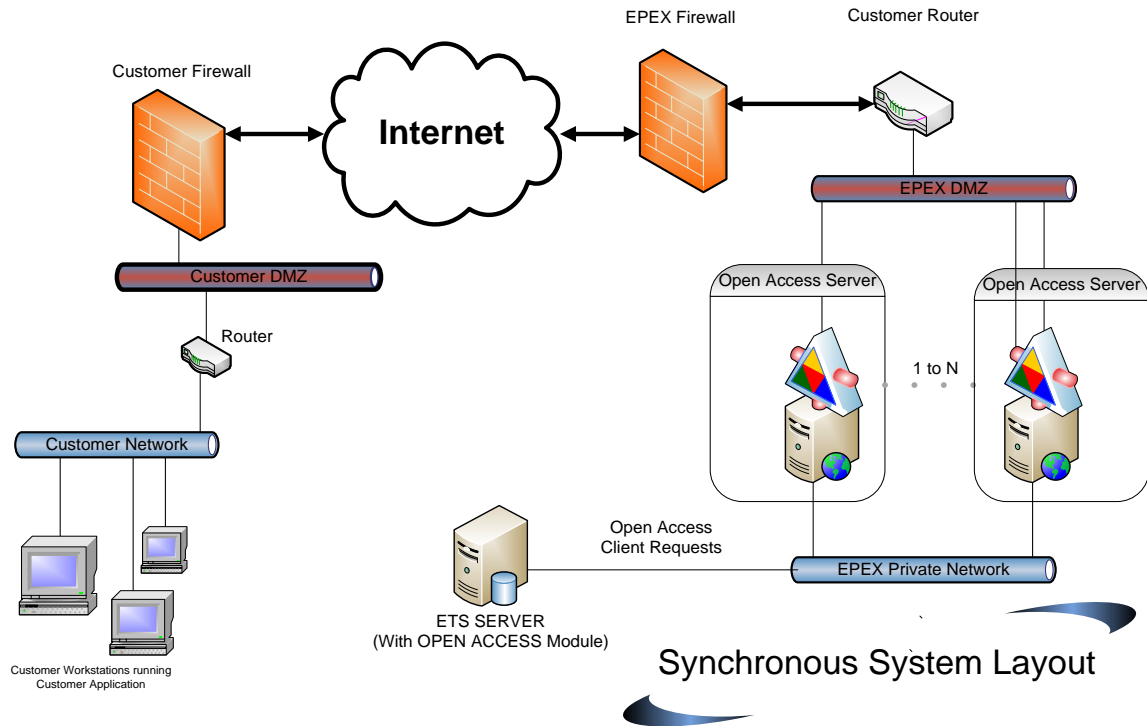


Figure 1: Synchronous System Layout

7. ETS API schema and WSDL file

In order to retrieve the ETS API schema (WSDL file) please refer to the “Environment Details” document in the ETS API package.

To each version of ETS and its API corresponds one single **WSDL file** version (Web Service Description Language). This file contains the **description of the different supported ETS API schemas**.

An **API schema** is compound of a set of web methods (e.g. *EstablishConnection*) with a specific set of parameters/attributes in the related requests and responses that can vary from a schema to another.

There are several ways to retrieve the last WSDL:

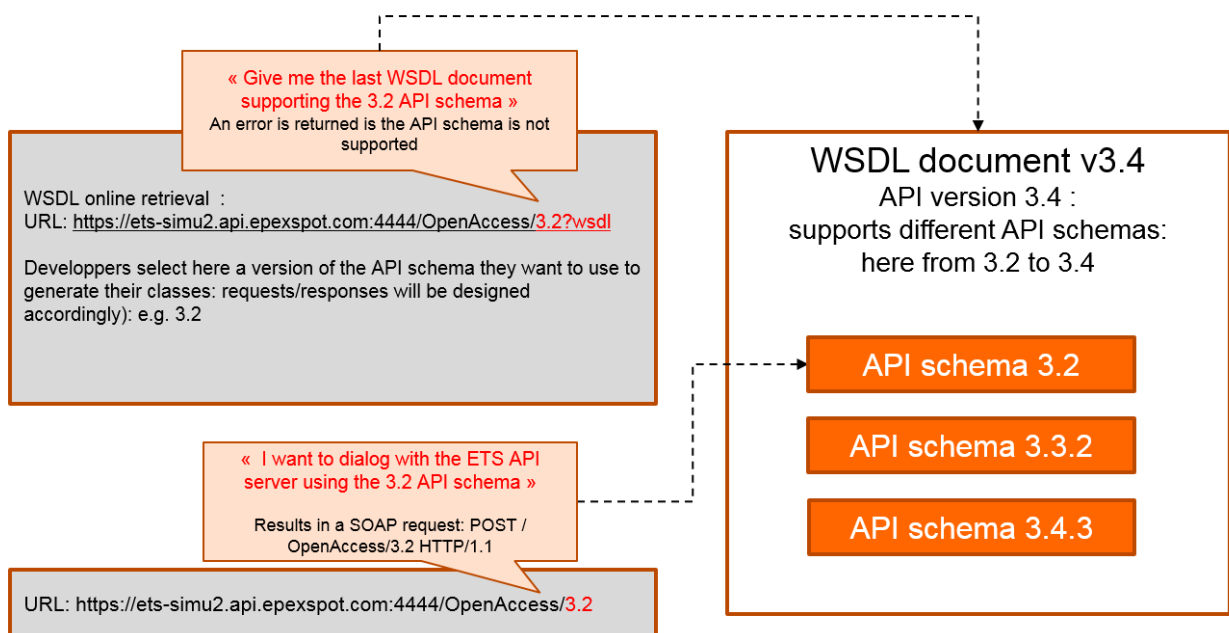
- **In the ETS API package.** Please contact our market operations team to get the latest API package.
- **In the SEMOpX info or DatanewZ communications** we send to announce new ETS API versions.
- **Online** from the relevant simulation environment:
 - o For instance for SEMOpX customers SIMU2 and 3.4.6 version with a request similar to:
 - <https://api1-ets.svpix.simu2.epexspot.com/OpenAccess/3.6> (if using the 3.6.4 schema (please favor Internet Explorer)

Please refer to the ETS API package “Environments Details” document to adapt the request above and retrieve your WSDL for one environment and API schema version.

Note : when a schema or a version has 3 digits, like 3.4.6, only use the two first one “3.4” to request the WSDL and in your API end point (to tell in which version you want to dialog).

What numbers refer to in URLs

Example with an app using the 3.2 API schema with ETS API 3.4



ETS API Schema version update:

We indicate in our communications whether or not an upgrade to a new API schema version is required or not. In general an upgrade is only mandatory to benefit from the new version functionalities. For instance we introduced with ETS API 3.3.1 a market result status that required to upgrade to the new schema, in order to get this status in the Market Result Retrieval response.

Technically upgrading to a new API schema means to re-generate your interfaces and classes from the new WSDL file so that methods changes are taken into account.

ETS API supported schema versions: current one + 2 previous ones

The old API schemas 2.10 and 3.0 will be decommissioned with ETS API 3.4 in Q1 2020, still supporting 3.2, 3.3 and 3.4 API schema versions.

Should you still use one of these schemas please start preparing the upgrade of your API application to a supported WSDL version, ideally 3.3.2 to benefit from the new Market Results Status (introduced with 3.3.1) and the Trade Report.

API Schema	Main introduced changes per API schema	ETS (and ETS API) version			
		ETS 3.3.2	ETS 3.4.6	ETS API 3.5.1 (ETS 3.5.x)	ETS API 3.6.1
API 2.10	(Initial schema)	Supported	Decommissioned	Not supported	N/A
API 3.0	-	Supported	Decommissioned	Not supported	N/A
API 3.2	<ul style="list-style-type: none"> EnterBlockORderBatch: <i>Minimum Acceptance Ratio</i> for curtailable blocks New methods for Loop Blocks 	Supported	Supported	Decommissioned (API schema 3.2 not supported in ETS 3.5)	N/A
API 3.3.2	<ul style="list-style-type: none"> Market Results status (<i>Unavailable</i>, etc.) Trade Report introduction (trade Id) Market results available to Non Market Participants 	Supported	Supported	Supported	Decommissioned (API schema 3.3.2 not supported with ETS 3.6.x)
API 3.4.3	<ul style="list-style-type: none"> New Methods to retrieve Trading Limits Reset Password Update Password Warnings distinguished from Errors Market results retrieval: RetrieveAuctionInformation enriched with Auction Max Cancellation time 	N/A	Supported	Supported	Supported
API 3.5.1	<ul style="list-style-type: none"> No security change: TLS and cipher suites changes postponed to 3.6 Spread blocks methods. Will be activated later. Security Update: decommission of TLS 1.0 and 1.1 and old cipher suites (please refer to SEMOpX communications for the exact date) 	N/A	N/A	Supported	Supported
API 3.6.1	<ul style="list-style-type: none"> Order IDs in order requests responses Last Update info in order retrieval API responses New Block comment in Trade report 	N/A	N/A	N/A	Supported

8. An exclusively Synchronous System Messaging

8.1 Synchronous mode

The API can only be used in synchronous mode:

For most messages, a SOAP header Complex Type: `AsynchronousResponseHeader` is required.

The 'asynchronousResponse' value is supposed to 'False' (value = '0'). But actually if you made the mistake to put a '1' it is ignored by the ETS API server which anyway forces the synchronous mode.

Please check the sample code for check the implementation level.

Principle: API client applications must wait for the ETS API answer before sending any new request (see precisions below in the Timeout section).

The below diagram shows the message flow in the synchronous mode of operation.

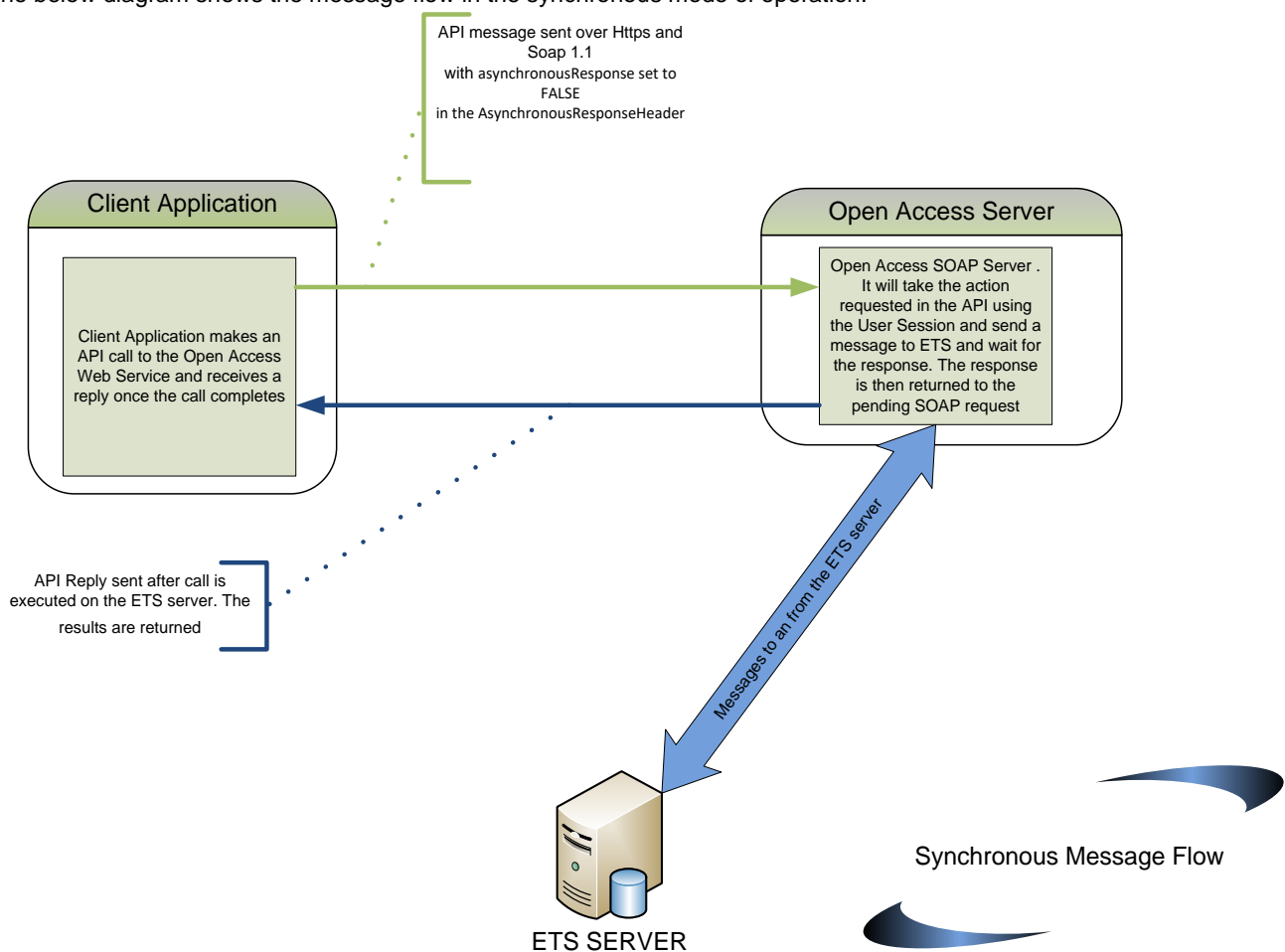


Figure 2: Synchronous Message Flow

Please note that the `<responseToken>` tag is not in use for ETS API. It is an optional tag that should not be used.

8.2 Messages structure

A SOAP request is compound of 2 parts:

- a SOAP request header:

Field	Option. / Mand.	Description	Example
Content-Type	Mandatory		text/xml;charset=UTF-8
SOAPAction	Mandatory	The name of the used method.	EstablishConnection
Host	Mandatory	The end point URL:point	Please refer to your Exchange Environment Details document in the API package
Content-length	Mandatory	Size of the of XML payload in bytes (calculated by most SOAP libraries)	756

- an XML envelope:
 - o a SOAP header
 - o a SOAP Body

XML part example:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:openaccess">
  <soapenv:Header/>
  <soapenv:Body>
    <urn:EstablishConnection>
      <userLoginName>APIUSER2</userLoginName>
      <password>ApiUser2</password>
    </urn:EstablishConnection>
  </soapenv:Body>
</soapenv:Envelope>
```

9. Open Access Login

Login: session token principle and Terms of Reference Login policy

The first thing a client application must do to work with the Open Access API is to **login and retain the resulting Session Token**.

The *EstablishConnection* method implements this login functionality. The Client sends over ETS a userloginname and password as parameters to this call. The Establish Connection Response contains the Session Token (User Login Name + Session Key) to use as a required SOAP header in later API calls (more exactly the com).

Example (from the API package sample requests):

1. Send an Establish Connection request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:openaccess">
  <soapenv:Header/>
  <soapenv:Body>
    <urn:EstablishConnection>
      <userLoginName>APIUSER2</userLoginName>
      <password>ApiUser2</password>
    </urn:EstablishConnection>
  </soapenv:Body>
</soapenv:Envelope>
```

2. Get the Session Token from the response:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns="urn:openaccess">
  <SOAP-ENV:Header>
    <ns:SessionToken>
      <ns:userLoginName>APIUSER2</ns:userLoginName>
      <ns:sessionKey>09465235504287065206466360524122979280744126929067291382274042729462155209809711</ns:sessionKey>
    </ns:SessionToken>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <ns:EstablishConnectionResponse>
      <EstablishSessionResponse>
        <ns:state>ACK</ns:state>
        <ns:sessionToken>
          <ns:userLoginName>APIUSER2</ns:userLoginName>
          <ns:sessionKey>09465235504287065206466360524122979280744126929067291382274042729462155209809711</ns:sessionKey>
        </ns:sessionToken>
      </EstablishSessionResponse>
    </ns:EstablishConnectionResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3. Re-use this Session Token in any later request, for instance a market results retrieval:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:openaccess">
  <soapenv:Header>
    <urn:ResponseLimitationHeader>
      <urn:range>
        <urn:from>1</urn:from>
        <urn:to>100</urn:to>
      </urn:range>
      <urn:resultSize>100</urn:resultSize>
    </urn:ResponseLimitationHeader>
    <urn:SessionToken>
      <urn:userLoginName>APIUSER2</urn:userLoginName>
      <urn:sessionKey>09465235504287065206466360524122979280744126929067291382274042729462155209809711</urn:sessionKey>
    </urn:SessionToken>
    <urn:AsynchronousResponseHeader>
      <urn:asynchronousResponse>1</urn:asynchronousResponse>
      <urn:responseToken>1</urn:responseToken>
    </urn:AsynchronousResponseHeader>
  </soapenv:Header>
  <soapenv:Body>
    <urn:RetrieveMarketResultsFor>
      <MarketResultIdentifier>
        <urn:area>FR-RTE</urn:area>
        <urn:auctionIdentification>
          <urn:AuctionDate>2018-05-05</urn:AuctionDate>
        </urn:auctionIdentification>
      </MarketResultIdentifier>
    </urn:RetrieveMarketResultsFor>
  </soapenv:Body>
</soapenv:Envelope>
```


Your ETS API client application must respect the Login policy described in the “Terms of Reference” document, provided in the ETS API package.

This login policy indicates how your application should log in and log out, as well as the maximum session token validity period.

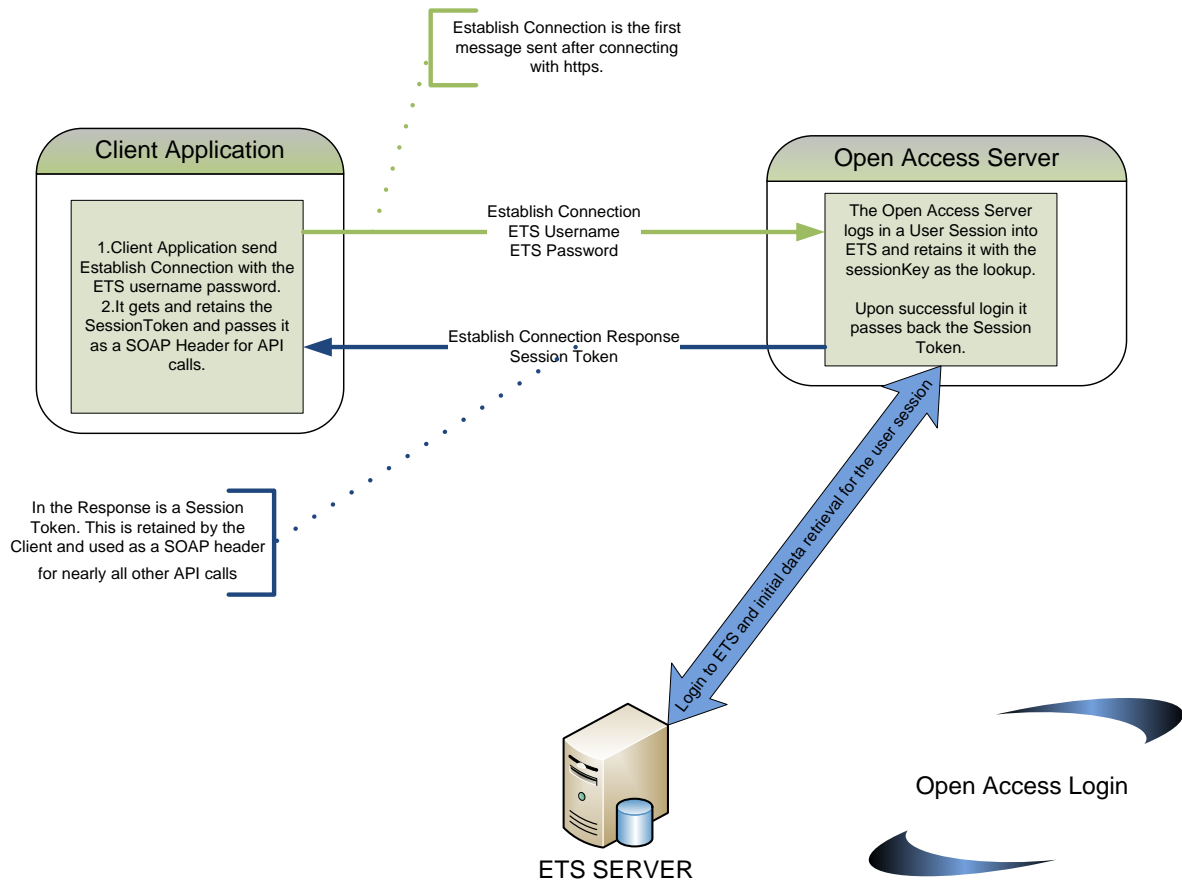


Figure 3: Open Access Login

How to recover from any technical session interruption

Retaining the User Session on the Open Access Server has the benefit of maintaining a session even if the HTTPS connection times out or gets disconnected.

The client application can re-connect the HTTPS connection and continue with the same Session Token as long as valid.

- This client does not even need to be run on the same machine but that client would need a valid client certificate in order to re-establish the https connection.
- This also avoids extra overhead when HTTPS connection time-outs occur.
- Since HTTPS connections are used there is no security issue with the session being held and run. Indeed a HTTPS re-connection requires a certificate and a valid username password.
- In case of a client crash: the ETS API session is still there and another client can log in with the aforementioned certificate and continue on from where it left things off.
- The server will not keep track of any message state: it will be up to the client to know and verify which API messages were sent and which responses have been received. Order states can be easily obtained using order inquiry requests (e.g. RetrieveOrders).

The ETS API requires a specific “API” user type

It is not possible connect to the ETS API using an ETS “fat” client user, and vice versa.

ETS API sessions are secure:

Using HTTPS, Client Applications running on customer workstations will send the *EstablishConnection* method and provide an user login name and password. **Since the HTTPS protocol is used along with client certificates, all login information is secure and encrypted.** For more information please see the “*Certificates*” section below.

Once a session is established a session token is returned that is used in the header of following API calls. This ensures session security and that orders are applied to the correct user.

10. Security and certificates

ETS API sessions are secure:

Because the communication between the Customer API Application and the ETS API Open Access Portal is over the internet, it must be secure:

- Both authentication and encryption are required for all communications between the Customer Application and Open Access Server.
- HTTPS connections are used for all the SOAP requests. This requires both client and server side certificates.
- Using HTTPS, Client Applications running on customer workstations will send the *EstablishConnection* method and provide an user login name and password. **Since the HTTPS protocol is used along with client certificates, all login information is secure and encrypted.**
- Once a session is established a session token is returned that is used in the header of following API calls. This ensures session security and that orders are applied to the correct user.
- During the development phase, the developer will use its own generated client and server certificates.
- *Please refer to the “ETS API Certificate” document in the API package for more information about the **Certificate process.***

Certificate Validity

Certificates are valid for 2 years.

SEMOpX Market operators monitor PRODUCTION certificates expiry dates and informs customers one month before the expiry date.

Customers will then be asked to renew the certificate process. Please consult the dedicated Certificate document in the ETS API package for more details.

Please consult the >1-API Specifications>04-ETS API Certificates dedicated document in order to further details:

- What is the process to get your signed certificate signed by SEMOpX
- How to build the KeyStore you need to connect your API application to the ETS API
- And much more

11.Warning and Error Management

The ETS API responds with 2 categories of errors: technical and functional.

11.1 Technical errors

Case 1 - Technical errors can correspond to an XML payload mistake in the request :

- an inappropriate parameter in the request, for instance empty or out of range
 - a) Example: requesting Auction Information for 0 nb of auction :

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:ets:api:1.0">
  <soapenv:Header>
    <urn:SessionToken>
      <urn:userLoginName>MARDATAAPI01</urn:userLoginName>
      <urn:sessionKey>113650631659770608713953664681160990188252531516917187299</urn:sessionKey>
    </urn:SessionToken>
  </soapenv:Header>
  <soapenv:Body>
    <urn:RetrieveAuctionInformation>
      <AuctionInformationQuery>
        <!--1 or more repetitions-->
        <urn:Area>GB-NG</urn:Area>
        <urn:NumberOfAuctions>0</urn:NumberOfAuctions>
      </AuctionInformationQuery>
    </urn:RetrieveAuctionInformation>
  </soapenv:Body>
</soapenv:Envelope>
```

Will trigger the following Error response:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>SOAP-ENV:Client</faultcode>
      <faultstring>Validation failed for: {AuctionInformationQuery}. Value: 0 must be >= <minInclusive value="1"></faultstring>
      <detail>
        <xsd:string>a WebServices.ValidationError</xsd:string>
      </detail>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

- Or for instance a request with a wrong XML format:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>SOAP-ENV:Client</faultcode>
      <faultstring>End of start tag expected, but not found</faultstring>
      <detail>
        <xsd:string>a XML.MalformedSignal</xsd:string>
      </detail>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Case 2 – Technical errors can correspond to:

- Any unexpected situation occurring at server level due to heavy network traffic, a server unavailability, server crash etc.:
 - a) Please check Terms of Reference *Order Inquiry Request* section explaining how to handle an “OA 012” “Server busy” error: **ErrorId = "OA 012"**, **Error text = "Trading System did not answer within: [x] Seconds"**
 - Basically this message is just there to indicate there is delay in the sending of the response from the ETS back-end server to ETS API server. It is not a "real" failure of the request. It means the request outcome is uncertain.
 - Your application must check whether or not the request was properly executed

- b) Heavy network traffic on API server, may lead to the error message “Transport error: 503 Error: Service Unavailable”

This 503-error means API server is busy and is unable to process the requests for the time being

In such a situation the client application must retry to send the API requests after few minutes (for ex: 5 minutes after receipt of the error).

11.2 Functional errors and warnings

The response contains different elements that enable the nature of the problem to be understood:

- A state, mainly **ACK** or **NAK**, (but potentially as well **NOT FOUND** or **ERROR OCCURED**) indicates there is a functional problem with the request, either at the API server or Back End level:
 - **NAK** is the response state for more general failures of execution of a request. It does not indicate an error in the system (i.e. ETS back end Server) but is usually returned after an invalid or inappropriate request has been sent (e.g. if mandatory information is missing from a request or a request contains an illegal value). This is detected at the API server level.
 - A typical example of an code that you may encounter is the expiry of the session token (given in the response to the Establish Connection request):

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns="urn:openaccess">
  <SOAP-ENV:Body>
    <ns:RetrieveAuctionInformationResponse>
      <RetrieveAuctionInformationAcknowledgement>
        <ns:state>NAK</ns:state>
        <ns:errors>
          <ns:errorId>OA 001</ns:errorId>
          <ns:errorText>Login Denied: Wrong session key</ns:errorText>
        </ns:errors>
      </RetrieveAuctionInformationAcknowledgement>
    </ns:RetrieveAuctionInformationResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

- **ERROR OCCURRED** is the response state in case of unexpected behavior in the system (e.g. there is a problem with the connection to ETS, or the API server times out because ETS takes too long to respond to a request), or if an error is reported by ETS back end server (e.g. back end server could not process the order submission).
- **NOT FOUND** is the response state when an object (area, area set, area setting, auction, order, instrument, participant, portfolio or trading limit) pertaining to a request cannot be found, either because the object either does not exist in the system or because the object is not recognized by the system in its current state.:

```
</SOAP-ENV:Header>
<SOAP-ENV:Body>
  <ns:EstablishConnectionResponse>
    <EstablishSessionResponse>
      <ns:state>ERROR OCCURED</ns:state>
      <ns:errors>
        <ns:errorId>OA 018</ns:errorId>
        <ns:errorText>Cannot Login because of Error: Could not connect to trading system.
          See further message(s) for details.
        </ns:errorText>
      </ns:errors>
      <ns:errors>
        <ns:errorId>OS 008</ns:errorId>
        <ns:errorText>Peer communications error</ns:errorText>
      </ns:errors>
      <ns:sessionToken>
        <ns:userLoginName>TMCQWGV01API</ns:userLoginName>
        <ns:sessionKey>0000000000000000000000000000000000000000000000000000000000000000</ns:sessionKey>
      </ns:sessionToken>
    </EstablishSessionResponse>
  </ns:EstablishConnectionResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

- An Error ID and an Error Text give more details.

Note: the list of Errors can be extracted from the WSDL file.

Warnings: (as of schemas 3.4.x)

Until the ETS API schema 3.3.2 the ETS API replies with an error code even when a warning is raised (e.g. Deprecated tags), which can be confusing.

The 3.4.3 API schema introduces a new `<ns:warning>` tag.

There is no change in the `errorId` or `errorText`. The only difference is the change in the main xml tag `<ns:errors>`, which at the same time become `<ns:error>` (no final 's' since it always contains one single error)

Example :

ETS API schema versions prior to 3.4.x	As of schema 3.4.x
<code><ns:errors></code> <code><ns:errorId>OA 014</ns:errorId></code> <code><ns:errorText>Area: [Area] not found or no Permission</ns:errorText></code> <code></ns:errors></code>	<code><ns:warning></code> <code><ns:errorId>OA 014</ns:errorId></code> <code><ns:errorText>Area: [Area] not found or no Permission</ns:errorText></code> <code></ns:warning></code>
<code><ns:errors></code> <code><ns:errorId>OA 001</ns:errorId></code> <code><ns:errorText>Login Denied: No Credentials</ns:errorText></code> <code></ns:errors></code>	<code><ns:error></code> <code><ns:errorId>OA 001</ns:errorId></code> <code><ns:errorText>Login Denied: No Credentials</ns:errorText></code> <code></ns:error></code>

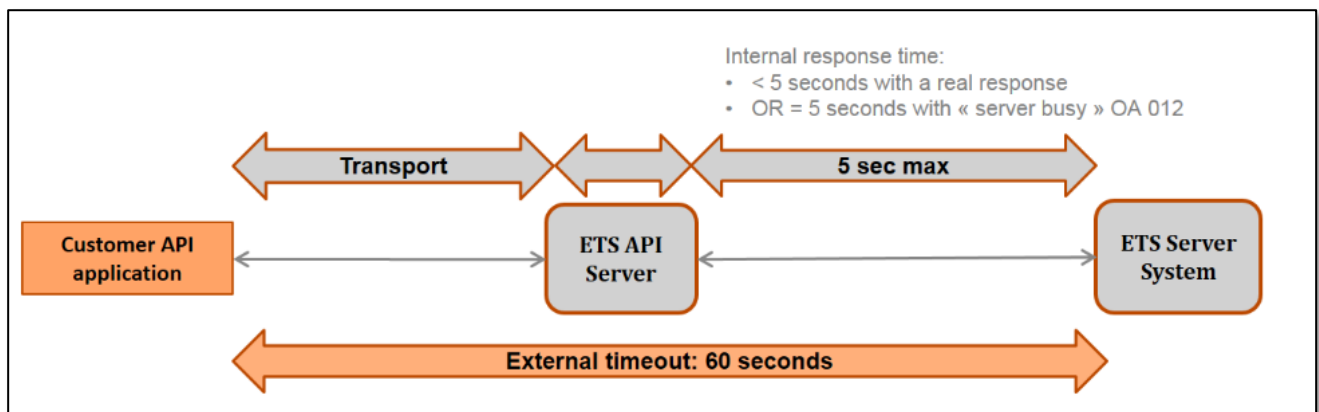
The following codes are concerned (errors that are warnings as of 3.4.3):

- (OA 016) Cannot resolve delivery Date ...
- (OA 022) Cannot resolve Area ...
- (OA 116) No Auction found for Delivery Date ...
- (OA 029) Missing Information to identify an Auction ...
- (OA 023) Cannot resolve Portfolio ...
- (OA 017) ... TimeInterval is required in this context
- (OA 116) Multiple Auctions for Delivery Date ...
- (OA 118) Multiple Auctions [for] ...
- (OA 027) Cannot resolve Multiple Durations
- (OA 028) Currency ... does not correspond to the (area/portfolio) combination
- (OA 014) Area: ... not found or no Permission
- (OA 002) Area Setting not found for: ...
- (OA 117) No Auction found [for] ...
- (OA 004) [No] Block order [with criteria] ... have been ... [not] found ...
- (OA 026) Complex Order [with Identification:] ... has not been found
- (OA 021) Daily order [with Identification:] ... has not been found
- (OA 005) Hourly order [with Identification:] ... has not been found
- (OA 013) Entered a wrong Period [starting at] ... [for] ... [because of]... [Auction Name] ... [Auction Time] ...
- OA 034) Member: ... not found
- (OA 015) Portfolio: ... not found or no Permission
- (OA 041) No trading limit

12. Timeouts and recovery procedure

The external client app timeout in 99% of situations, is the sum of:

1. **Transport time** between the client app and the ETS API server
2. **Processing time at the API server level:**
 - unmarshalling , marshalling (not linear): depends on message size.
 - Experience shows this processing time starts to increase significantly as of 2 Mb
3. **API/ETS/API server time** (mainly ETS server processing time), which depends on the internal timeout
the « internal » timeout is always = 5 seconds:
 1. If the ETS server processing time < 5 seconds, then the response is sent to the API server within these 5 seconds (as soon as available)
 2. If > 5 seconds then the timeout is reached and the sent response is OA 012 « server busy » message and the request processing continues
 - “server busy”: Error ID = “OA 012” ; Error text = “Trading System did not answer within: 5 seconds” : please check Terms Of Reference for more details on how to react.



We recommend to setup a 60 seconds timeout at your application level.

Messages size should not exceed 2 Mb, to guarantee a response within this timeout value (in most cases the response time should be much shorter).

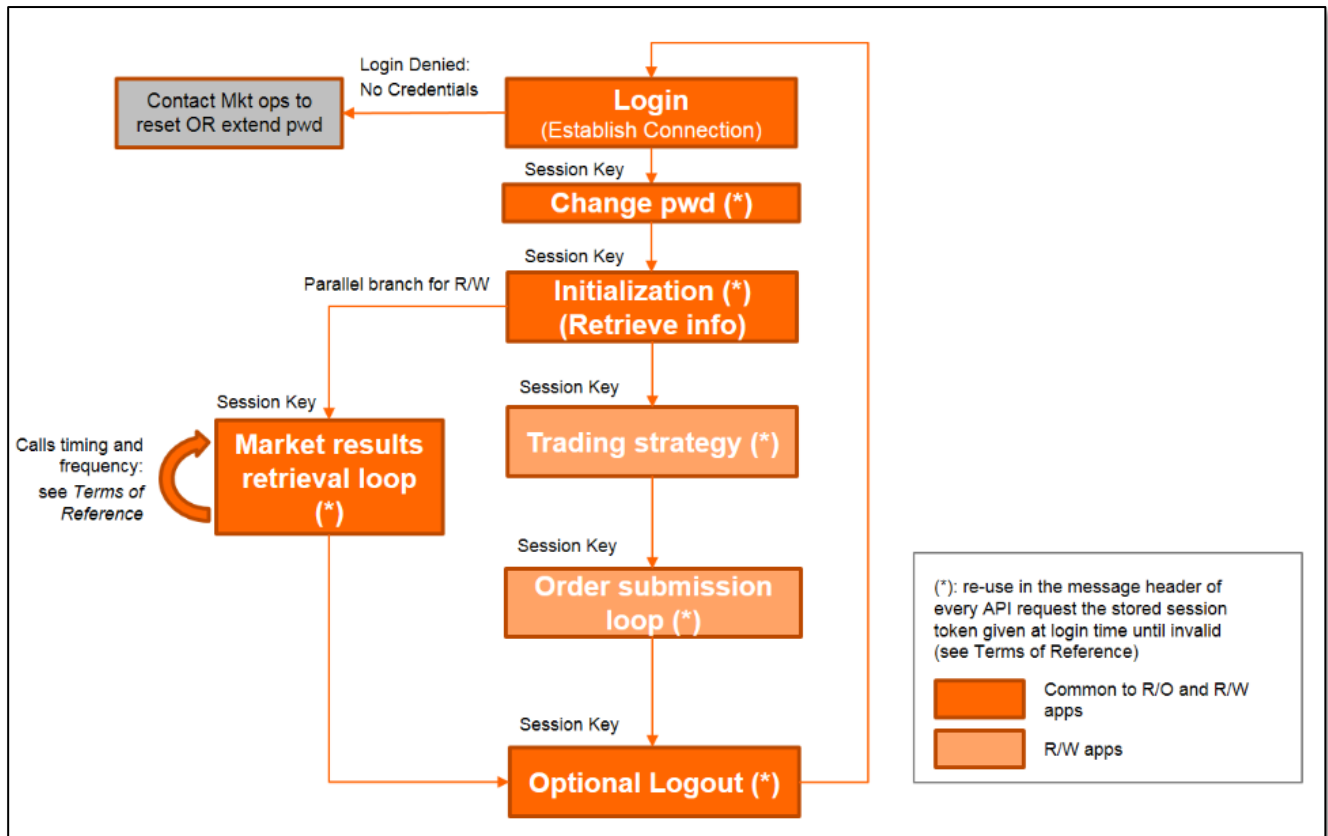
Timeout Recovery procedure recommendation:

Your API application should get an answer in 99% of situations, but in case your API app sends a request and does not receive any response after having waited for 60 seconds, we recommend the following:

1. Raise an internal warning and inform SEMOpx so that we can investigate what happened
2. Check the state of your technical HTTPS connection and recreate one if necessary
3. Your application is responsible for checking whether or not the request that timed out has been processed:
 - If an inquiry request timed out:
 - it is of course enough to just resend it (there was no other expected result than a response)
 - If an order management request timed out:
 - you must first use a retrieve order request to check if your request was processed or not before trying to send it again
4. If your new request times out again:
 - Logout and log back in (EstablishConnection resulting in a new session token)
 - Repeat the same steps:
 - Either the issue is solved
 - Or the same issue remains
 - In any case please inform SEMOpx so that we can investigate what happened

13. Typical structure of an ETS API application

A typical Read Only or Read Write application can be designed as follows:



The table below further explains the different steps, indicating which one are appropriate for:

- Read Only (RO):
 - o Most of the time RO apps retrieve Market Results
 - o They can also gather orders submitted from different apps (ETS client, different API apps)
- or Read-Write applications managing orders (R/W), also called “Trader applications”.

M: Mandatory / R: Recommended / O: Optional

Step	RO	R/W	Step Description
Login	M	M	<ul style="list-style-type: none"> - Send the Establish connection request - Wait for the response and store the session Key in the received session Token - Contact the Market Operations team if the API sends back a “Login Denied: No Credentials” error
Update/ change Password	R	R	<ul style="list-style-type: none"> - You have 7 days to change the initial password. Market operations can only extend this initial duration to 10 days. - Changing/updating a password automatically via the API extends its validity to 90 days. - Though market ops can reset or extend a password this option should be kept only to emergency, regular changes should be handled by the API application itself
Initialization	R	R	<ul style="list-style-type: none"> - Initialize your API client with all the necessary “retrieve information” requests (e.g. to know the price tick or max price of an area and implement a check against order prices, or retrieve the auction characteristics to know Market Results <i>Theoretical Publication Time</i>). - See <i>Terms of Reference</i> for the usage of Inquiry requests
Trading strategy	O (*)	O	<ul style="list-style-type: none"> - Trading strategy : defined here as the algorithm that calculates which orders the API app must submit - this algorithm/logic can be directly in the ETS API application or located in another system that directly feeds the API order submission loop below - (*) For read only apps : It is also possible to your API app needs to retrieve orders to feed another system that takes care of the order calculation: in that case, your API app must retrieve orders
Order submission loop	N/A	M	<ul style="list-style-type: none"> - For all orders that need to be submitted or managed in a row, please repeat the following sub-steps: <ol style="list-style-type: none"> 1) Send the required “order entry” request (select the method related to the order type: EnterOrder, EnterBlockOrderBatch, etc.) 2) Wait for the API response 3) Wait for x seconds (should ideally be configurable: e.g. 1 second) before sending the next order - Please see the <i>Timeouts</i> section above and <i>Terms of Reference</i> for any further guidance on dealing with API (lack of) response.
Market Results Retrieval loop	O	O	<ul style="list-style-type: none"> - Optional since it depends on the purpose of the application.
Logout	O	O	<ul style="list-style-type: none"> - See <i>Terms of Reference</i> for Logout usage

Note: The ETS API works in synchronous mode. This means that an API application should always wait to receive a response to the sent request before trying to send the subsequent request.

ETS API attention points

14.1 How to properly identify an auction in request messages?

There are several possibilities to identify an auction in an API request:

Possibility	Comment and Example
<deliveryDay>	<p>Works as long as it enables to uniquely identify an auction:</p> <pre> <urn:EnterOrder> <Order> <urn:area>your order delivery area</urn:area> <urn:portfolio>TEST-T01</urn:portfolio> <urn:deliveryDay>2018-05-06</urn:deliveryDay> </pre>
<deliveryDays>	<p>Dates range : this possibility is proposed by the ETS API but is not applicable so far for your Exchange, since several auctions would correspond to the date range</p> <ol style="list-style-type: none"> <from> Starting date <to> Ending date <p>Request:</p> <pre> <urn:EnterOrder> <Order> <urn:area>delivery area</urn:area> <urn:portfolio>TEST-T01</urn:portfolio> <urn:deliveryDays> <urn:from>2018-09-22</urn:from> <urn:to>2018-09-23</urn:to> </urn:deliveryDays> </pre> <p>Response:</p> <pre> <ns:EnterOrderResponse> <EnterOrderAcknowledgement> <ns:state>ACK</ns:state> <ns:OrderId>102365</ns:OrderId> <ns:OrderId>102366</ns:OrderId> </EnterOrderAcknowledgement> </ns:EnterOrderResponse> </pre>
<AuctionIdentification> (recommended)	<p>This is mandatory to use where there are several auctions per day (e.g. GB-IDA1 and GB-IDA2 Irish auctions).</p> <ul style="list-style-type: none"> Possibility 1: <UTCDateTime>: UTC Timestamp of the Auction execution <pre> <urn:EnterBlockOrderBatch> <blockOrderBatch> <urn:portfolioName>TEST-T01</urn:portfolioName> <urn:auctionIdentification> <urn:UTCDateTime>2018-05-05T12:00:00+02:00</urn:UTCDateTime> </urn:auctionIdentification> <urn:areaName>your delivery area</urn:areaName> </pre> Possibility 2: <AuctionDate> + <name> <ol style="list-style-type: none"> AuctionDate = Date of the Auction Example Name = PWR-SEM-GB-D+1 or PWR-SEM-GB-D Optional <duration>: '15 min', '30 min', '60 min' can be used along with possibilities 1 and 2:

	<ul style="list-style-type: none"> i. When using it as an Input for an <u>order entry/order modify</u> action: <ul style="list-style-type: none"> • the tag <duration> should contain the value as 15 min, 30 min, or 60 min. • for an order entry/modification input, the tag <duration> can have only one Auction Duration value • the specified duration is used to affect only this duration. • If no auction corresponds to this duration then the action will fail. • if the tag <duration> has an empty value or the tag <duration> itself is removed in the request file, then the API Request will FAIL ii. When using it as an Input for an <u>order retrieval/order cancellation</u> action: <ul style="list-style-type: none"> • if there is a value for <duration> tag (for example 15 min) then the auction with the specified duration will be affected. <p>For instance here, 15 min auction will be affected for a given Auction Date and Market Area</p> <ul style="list-style-type: none"> • if <duration> tag is omitted then all the available Auction Durations are affected for a specific Auction Date and Market Area
--	---

14.2 Deprecated messages or tags

1) Decommission of deprecated tags to come in 3.4.3:

Deprecated tags are old tags that SEMOpx still supports but only for backward compatibility reasons.

In general we recommend you to stop using these deprecated tags and use the supported ones instead, to upgrade more easily to the latest API schema.

These deprecated tags will be decommissioned from the 3.4.3 API schema with the release of ETS API 3.4 in Q1 2020.

Applications using API schemas 3.1, 3.2 and 3.3.x will be still able to use these deprecated tags. When using these tags, please be aware the API server will insert in the response a warning message : ***Used deprecated Choice : <deprecated tag name>***.

It is not related to any error but just to warn you that your application is using an old tag.

You will find in the API Package ***ETS API_Deprecated Tags Decommission List.xlsx*** document thate list of the deprecated tags that will be decommissioned with 3.4. Should this list be further extended to additional tags we will inform you in advance.

14.3 Order entry and ETS Portfolio set up

In order to submit any type of orders via the ETS API, an API user must have:

- at least one active portfolio with:
 - The general read/write permission
 - The permission to submit an order on the corresponding area
 - If relevant for the traded area set (*):
 - the optional/on demand permission to submit smart blocks (C02, C88, C04)
 - the permission to submit big blocks

(*) the corresponding area set must be configured to allow the user to submit a specific order type (ex: Classic Block Order (C01), Linked Family (C02), Loop family (C88), Exclusive Group (C04)).

Please contact our Market operation team to double check your API user and portfolio settings if required.

14.4 REMIT Trading on Behalf data: **Optional** Beneficiary and Trading Capacity (A,P) attributes:

The ETS API enables as well market participants to enter information related to their trading on behalf, order per order:

- a **trading capacity** = (**optional**)
 - A for Agent, when trading on behalf of another company,
 - P (for proprietary trading),
- a **beneficiary**, when trading capacity = A: (**optional**)
 - Optional, Alphanumeric (12 characters).
 - The field is meant to be populated with a unique ACER code.

The following 3.7 methods are enriched with these new **optional attributes**:

- **<beneficiary>** - optional Type *string* <http://www.w3.org/2001/XMLSchema> restriction { maxLength="12", minLength="1" }
 Text box to populate the ACER Code.
 Optional, Alphanumeric (up to 12 characters), it can be populated only if Trading Capacity field = "A".
 Example: A00010247.SE
 only usable in API version >= 3.7
- **<tradingCapacity>** - optional Type *string* <http://www.w3.org/2001/XMLSchema> restriction enumeration { 'P', 'A' }
 Trading Capacity field: Optional. This field can contain either "P" or "A" characters.
 Per default, it should always be "P".
 "P" stands for Proprietary.
 "A" stands for Agent.

Impact on the following APIs:

- **Enter orders methods:**
 - EnterScalableComplexOrder
 - EnterOrder
- **Retrieve orders methods:**
 - RetrieveScalableComplexOrders
 - RetrieveOrders
- **The Trade report content :**
 - RetrieveTradeReportFor

15. Java and Python Client Examples

The ETS API Package features basic sample code in Java and Python illustrating how to log in ETS via the API:

- First log in, using the EstablishConnection method
- The response provided by the API server contains the sessionToken needed for any further API calls (until it gets invalid, see Terms of Reference for further details)
- Please note that in the SOAP header:
 - o 'asynchronousResponse' variable set to 'false' indicates the API synchronous mode
 - o the 'SessionToken' header will carry the session token obtained from the login sequence, for any further API call, until invalid

The API is straight forward to use in any language especially if there is a class builder that reads the WSDL and generates the interfaces and classes for you.

Java and Python

Please check the commented sample Java code in the ETS API package.

16. Client development guidelines

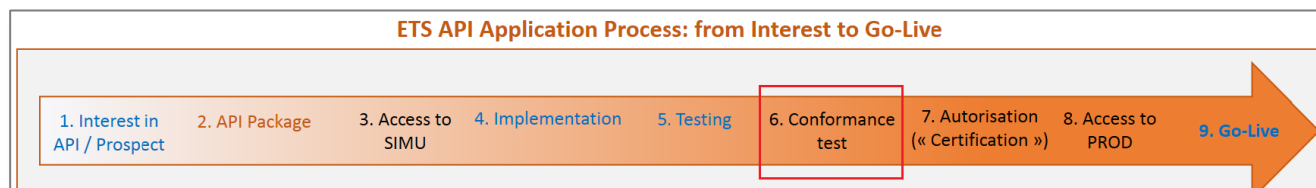
The development of a client should follow the following rules:

- respect all the policies and rules described in the “Terms of Reference” document,
- No undocumented method should be used,
- The client must not try asynchronous mode in the client, as this is not supported,
- The client must use its own credentials (every customer should have its own certificates issued by or through SEMOpX, please refer to the “ETS API Certificate” document in the API package for further explanations).

17. API Conformance Test

17.1 API Conformance Test milestone

The **01-ETS API Process for customers** document positions the **Conformance Test phase** in the API application life cycle:



An API Conformance Test is mandatory for any new API application.

For existing applications, your Exchange explicitly communicates when a Conformance Test is required or not (in case of an ETS API version upgrade, DST).

Objective of Conformance Tests:

- enable our customers to have the **best API experience once in Production**
 - **by detecting before production any incorrect implementation:**
breaching our Terms of Reference, that could lead to service interruptions in production:
 - a) e.g. no respect of our Login Policy
 - b) e.g. no respect of our Market Results / Trade Report retrieval policy
 - c) e.g. no respect of our Trading limits retrieval policy
- that would degrade the API customer experience:**
- Improper order entry request (please use orders similar to your production data)
 - Unable to retrieve market results / trade report
 - Improper management of DST situations (tested a few weeks before DST days)

17.2 API Conformance Test operational steps

Conformance test Pre-requisites: your application must be already completely tested on your own before you apply for a conformance test. Please contact Market Operations to perform any required action (e.g. market orders entry, running the auction) needed to complete your internal test phase.

When ready please contact Market operations to register for a conformance test. You will receive a communication giving you instructions like below:

Up to this point you have gained access to the simulation environment with your application and you have implemented and tested all the features you need to operate your application in the production environment.

Before this access can be granted your API application has to go through a conformance test (as described in our ETS API Specification Package) in order to ensure that your implementation respects the ETS API Terms of Reference.

Please find below the steps that need to be followed for the conformance test.

Scope: Depending on the functional scope of your API application there may be steps that need to be skipped since irrelevant.

Please arrange with Market Operations team the exact date and timing for the test as well as the market area(s) that will be tested. In particular if you follow/trade several auctions, please focus on one single auction for the conformance test.

Conformance Test Steps	Read only applications	Read Write Applications (able to place orders)
1. Preparation	<p>At the agreed date for the agreed auction(s):</p> <ul style="list-style-type: none"> Login when the Orderbook of the agreed auction(s) for the test is open (e.g. in the morning of the test day) Ensure the API user used during the test is not used by any other of your ETS API application at the same time. Choose the test environments (SIMU1: iso-prod or SIMU2: iso-prod or next version) on which the same ETS API version as the one your target in production is deployed. <p>Note: the point is not necessarily to place orders, but to log in soon enough to have the time to perform step #2 and so that we can check your API app does not start retrieving results too early (see Terms Of Reference).</p>	
	<p>In case your app retrieves market results or own orders :</p> <ul style="list-style-type: none"> Place orders for the agreed auction(s) with another user: <ul style="list-style-type: none"> via the ETS Client or via another Read Write API app (custom/in-house or ISV software) 	<p>In case your app retrieves market results or own orders :</p> <ul style="list-style-type: none"> Place orders for the agreed auction(s): <ul style="list-style-type: none"> a) with your API app b) and any other app which is part of your production setup (ETS Client, custom/in-house or ISV software)
	<ul style="list-style-type: none"> The Market operations team will also place orders in the market in order to create trades for you (i.e. to ensure that the orders you placed are executed when running the auction). 	
2. Login Policy	<ul style="list-style-type: none"> Log out (after your first login) before the publication of the auction and then immediately re-log in: <ul style="list-style-type: none"> your app sends a new <i>EstablishConnection</i> request as indicated in our <i>ETS API Terms Of Reference</i> (ToR) your API app must use the same session token (API user name + Session Key) across the entire test (as long as valid) as we want to ensure that your API app does not create unnecessary established connections. Note: potential Login issues due to an expired user or an invalid password will not be counted as breaching ToR 	
3. Market results / Trade report retrieval policy	<ul style="list-style-type: none"> Let your API app retrieve market results automatically when they are available (as defined in our API <i>ToR</i>). Ensure your API app can as well retrieve market results "on demand" (as per <i>ToR</i>) 	
4. Conformance Test analysis & results	<ul style="list-style-type: none"> On the day after the test Market Operations team will check the logs of your API to assess if all the criteria have been met (the correct xml requests have been sent), that your API app respects the <i>Terms of References</i> Once your requests are analyzed the Market Operations team will contact you to provide you with the conformance test results: <ul style="list-style-type: none"> a) Passed : <ul style="list-style-type: none"> i) your app is authorized to be operated in production, b) Passed with a caveat : <ul style="list-style-type: none"> i) you may be authorized to operate your application in production, 	

	<ul style="list-style-type: none"> ii) but you have a certain period to implement some adjustments and go through a new conformance test. c) Failed : <ul style="list-style-type: none"> i) your app is not authorized to be operated in production, ii) explanations are given on what needs to be adjusted before attempting a new conformance test.
5. Next Steps	<ul style="list-style-type: none"> • In case you passed the conformance test please arrange the documentation work with your Key Account Manager (for members) / the Market Data team (for ISV and non-members/Data Vendors) if anything is pending • and then provide the Market operations team with the .csr (Certificate Signing Request) for the Production environment.

When does your app need to go through a new conformance test?

- **your app does not need to go through a new conformance test until:**
 - a) SEMOpx asks you to do it (e.g. new ETS API version, issues observed in production)
 - b) You perform an update of the code that handles the logic explained in our ToR : please contact market operations before releasing your new version in production so we can assess the need of a potential new conformance test

18. Use Cases

The 3 use cases below use a subset of sample requests/responses files available within the API package.
Please check the package content to see all available sample files.

The provided sample Market results are the result of:

- orders featured in the package sample files
- other counter orders that have been submitted using other portfolios or test market participants (not shown in sample messages), to ensure some orders are executed.

Please note that market result responses are different:

- before 3.1.1 API schema: there is no status
- as of 3.1.1 API schema : the status (Unavailable, Preliminary, Final, Cancelled) is featured in the response
- for Non Market Participants only the indices part is featured (no private execution information)

18.1 Normal Day

Step #	API request step	Description																		
1	Establish Connection	The test customer application logs in via its Trader user APIUSER2 using the <i>EstablishConnection</i> method. Then it waits for the response: the successful response contains a session token to be used for all other API calls, until this token becomes invalid.																		
2	Keep Alive	In test environments the application sends a “Keep Alive” request to make sure the HTTPS connection is active. This is a useful check to be done when starting the implementation to check that the app is well technically connected to the ETS API server before getting any further.																		
3	Update Password	It changes the password using the <i>UpdatePassword</i> method. <i>Note: in the sample the same password is used for the old and new values so that the set of requests can be replayed.</i> <i>NB: SetNewPassword must be replaced by UpdatePassword as of the 3.4.x schemas</i>																		
4	Enter Order	It submits 1 linear order on the FR-RTE area and TEST-T01 portfolio. <table><tr><td>Prices</td><td>-500</td><td>0</td><td>500</td><td>1000</td><td>1500</td><td>2000</td><td>2500</td><td>3000</td></tr><tr><td>Quantity</td><td>700</td><td>600</td><td>500</td><td>400</td><td>300</td><td>200</td><td>100</td><td>0</td></tr></table>	Prices	-500	0	500	1000	1500	2000	2500	3000	Quantity	700	600	500	400	300	200	100	0
Prices	-500	0	500	1000	1500	2000	2500	3000												
Quantity	700	600	500	400	300	200	100	0												
5	Enter Block Order Batch	It submits different block orders on the FR-RTE area and TEST-T01 portfolio.																		
-	N/A	The order book closes and the auction is run: purchase and sell curves are calculated. For period 1:																		

		<p style="text-align: center;">P/S Curves</p> <p>The auction price is determined and results are published.</p>	
6	Retrieve market results	The application retrieves market results.	
7	Logout	The application logs out (optional, recommended for applications with a GUI).	

18.2 DST 23 – Summer

Only 23 periods are valid. Period #24 is invalid.
Please consult the Sample files provided in the API package.

18.3 DST 25 – Winter

25 periods are possible for this long day:

- period 4 corresponds to 02X-03X,
- period 25 corresponds to 23-00

Please consult the Sample files provided in the API package.