EPEX SPOT
EUROPEAN POWER EXCHANGE

ETS API Terms of reference

Public message interface

Date: 20/05/2019

Place: Paris

Document release: v4

Content

1. Introduction 3
   a. Reference 3
   b. Audience 3
   c. Purpose 3
   d. Obligation to cooperate 3
2. Requirement 4
   a. Testing 4
   b. Request 4
      i. Login and logout 4
      ii. Retrieve information request 5
      Market results retrieval 5
      Assignments inquiry requests 8
      Order inquiry requests 8
      iii. Message Validation 9

1. Introduction

a. Reference

This document is an amendment to the "API Software Design Guide" document.

b. Audience

This document is addressed to developers developing applications for the Public API messaging interface of ETS.

c. Purpose

The purpose of this document is to describe the expected behavior of the application using the API Public message interface of ETS. It is a set of requirements, rules and regulations which need to be adhered and applied to using the ETS API. These rules will ensure the proper functioning between client, systems and application on exchange member side or non-market participant side (e.g. Data Vendors) and ETS on the side of EPEX SPOT.

d. Obligation to cooperate

The rules mentioned in this document should be respected by the application's developers. EPEX SPOT reserves the right to block access to the ETS API server and disconnect the customer's API user from the EPEX SPOT system (API server and ETS server) at any time if the application of the member endangers the stability of the Trading System.

2. Requirement

a. Testing

It is requested to the application provider to perform sufficient tests in the Simulation environment covering all functional features of its application before asking EPEX SPOT to release its application to production environment.

The execution of specific tests case in Simulation can be requested by EPEX SPOT to members for quality assurance reasons.

b. Request

An client API application cannot run polls that keep the server busy continuously.
The different policies below (e.g. login, market results retrieval) indicate you the rules your application needs to implement.

The client must use its own credentials (every customer should have its own certificates issued by or through EPEX SPOT) and no undocumented methods should be used.

Please note that only synchronous mode is supported.

i. Login and logout

The previous version of this document only indicated this statement:

> The client should not take up unnecessary resources. In particular login request should not be initiated for single requests.

Actually, considering the important growth of API applications in 2018, we had to go one step further.

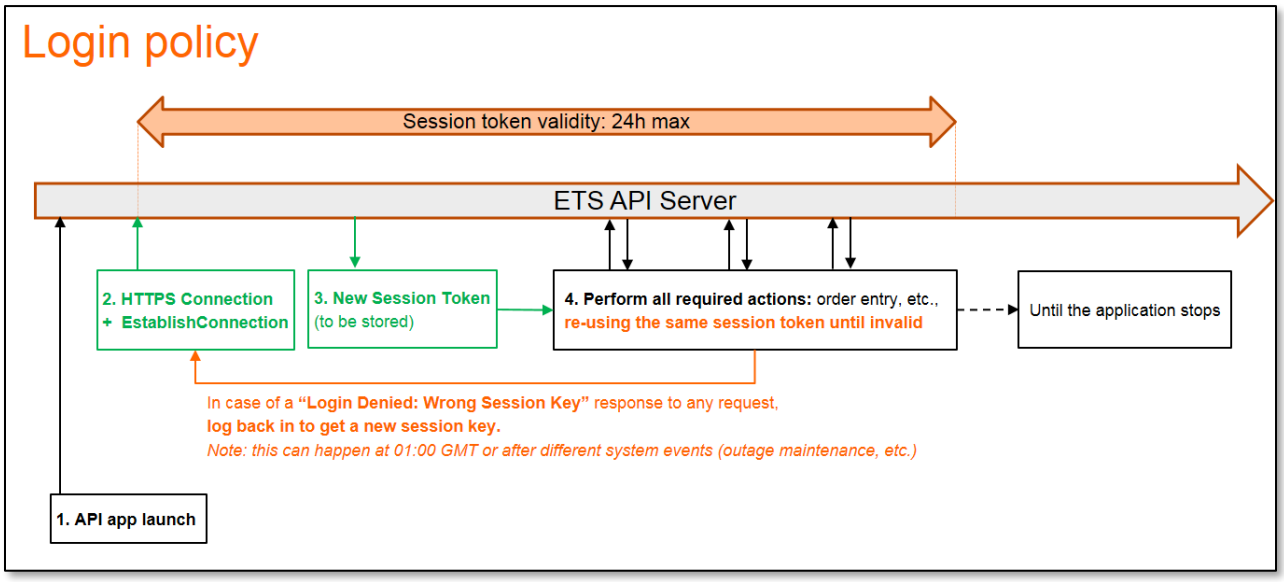
Indeed:

- EstablishConnection (Login) is the most demanding API method, its use should be minimized,
- the Session Token (User Login Name + Session Key) delivered in the Establish Connection response remains valid even if the HTTPS connection has been closed because of inactivity: it is not required to re-log in to get a new token.

As a result we ask you to follow the below connectivity cycle:

1. Your API application gets started
2. Log in: create an HTTP connection and send an EstablishConnection method (user + password)
3. Store the responded Session Token (user + session key)
4. Perform all required further actions by sending requests re-using the same token, until invalid
5. Once invalid, any request using it receives the response: "Login Denied: Wrong Session Key"
    - This will happen every early morning at 01:00 UTC when all connections to API servers are automatically kicked out for security purposes. At this moment all session keys become invalid.
    - This can also happen in case of an ETS maintenance or of an ETS outage.
6. When this happens go back to step 2 to get a new session key for the same user

Note: no explicit Logout is mandatory. The Logout method is only offered for customers needing it to comply with their company security policy.



HTTPS connections management:

- In case of an HTTP connection loss, just create a new one before sending a new request with the same session token.
- Please reserve the use of the *Keep Alive* method only for testing purposes, to check that your application can communicate the ETS API server (which does not require to be logged in), rather than for maintaining the HTTP connection alive.

ii. Retrieve information request

Market results retrieval

Market results must to be retrieved only when relevant:

- Starting at the theoretical auction publication time
- Stopping once the auction results are Final (and thus cannot be cancelled anymore)

The shortest time interval between 2 *RetrieveMarketResultsFor* requests for the same area **should be 30 seconds.**

Determining when to stop retrieving market results requires a good understanding of the different possible sequences of market results statuses and what is available via the API.

The sequence of market results statuses depends on the nature of the auction:

- Local and intraday auction:
  - *Unavailable* (Not yet published) -> *Final*.
  - There is no cancellation possibility.
- Day-ahead coupled auctions: *Unavailable -> Preliminary -> Final*
  - Thus this never happened in production it is process wise possible that preliminary market results get cancelled:
    - *Unavailable -> Preliminary -> Cancelled -> Preliminary -> Final*

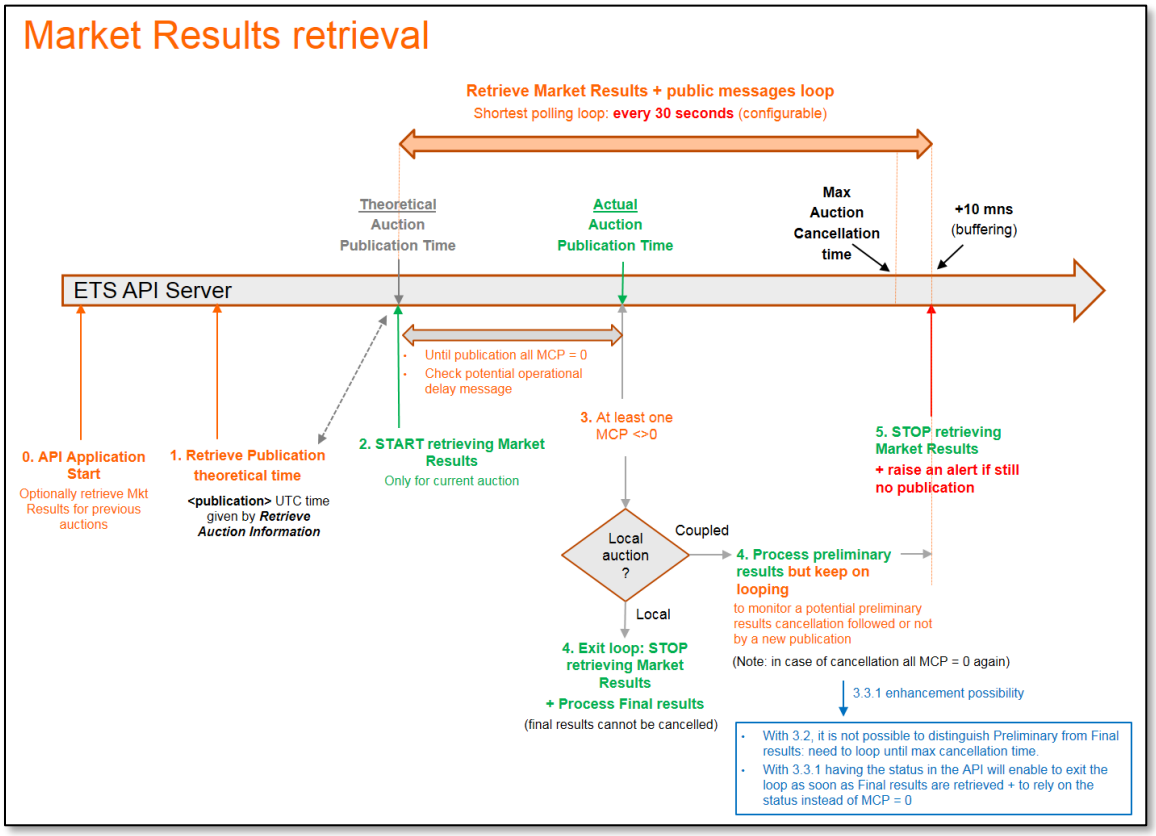
Please note that:

- though ETS displays market results in publication pop-ups, this information is not available via the ETS API 3.2 an earlier versions.
- **this status is introduced in the API response as of ETS API 3.3.1:** we highly recommend you to use this version of the API schema to simplify your application and make it more efficient
- the *RetrieveMarketResultsFor* API method always sends a response with zero values when results have not yet been published.

As a result:

- it is required in ETS 3.2 that your API application relies on a **convention to determine that market results have not yet been published**: *as long as all Market Clearing Prices equal 0 in the RetrieveMarketResultsFor, please consider that results have not yet been published.*
- In cases where preliminary results are published first, It is not possible until ETS 3.3.1 for your application to distinguish in the *RetrieveMarketResultsFor* response if results are still Preliminary or Final:
    - Your application should keep on retrieving results until a maximum cancellation time that we indicate in a table below, to ensure that results are still valid
    - As of 3.3.1, as soon as your application detects the switch to Final, it will not be necessary to keep on looping and retrieving again market results.
- At any stage of the publication process, your API application needs to read the content of public messages via the *RetrieveMessagesUnreadOnly* method: Delay information can be provided as public messages (e.g. for MRC: *[ExC_02]: Delay in Market Coupling Results publication* at 12:42 CET)
- Please use the *SetMessagesAsRead* method to change messages status Unread/Read.

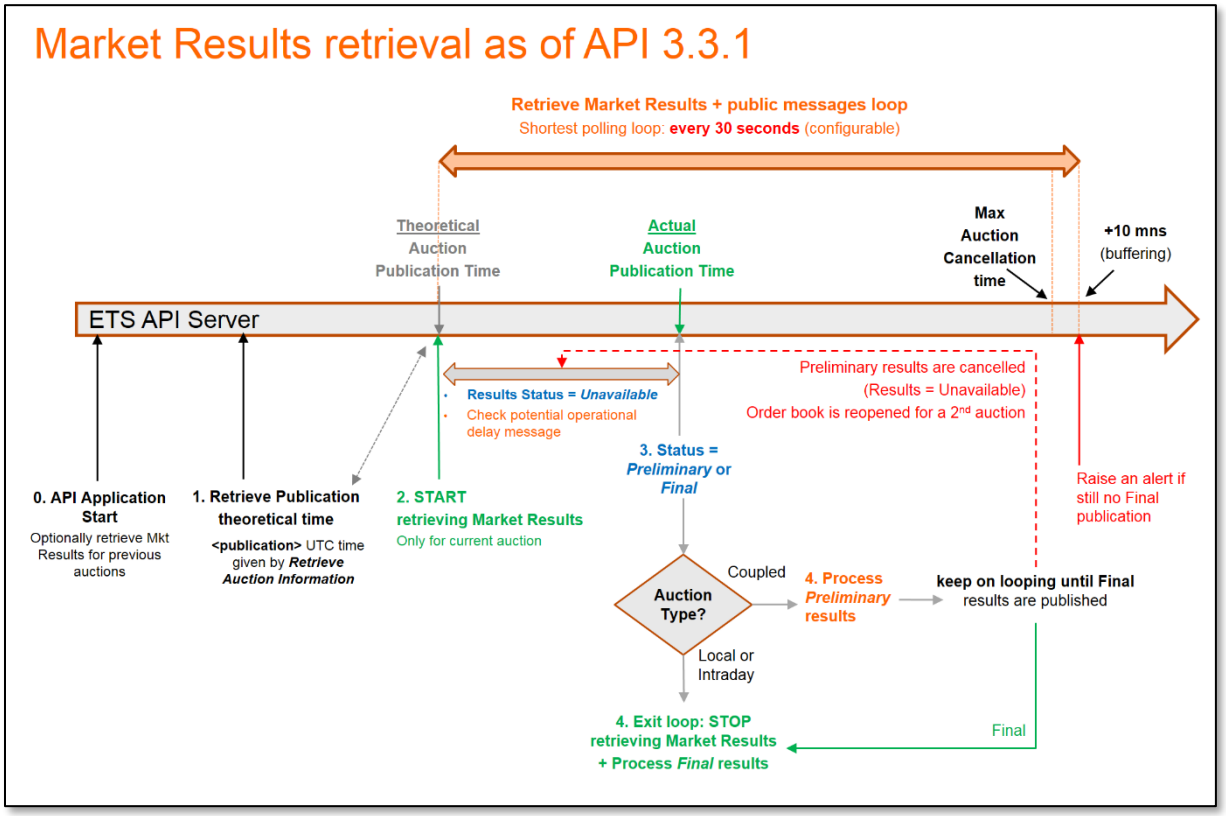
The diagram below summarizes the logic to be implemented before 3.3.1:



As of 3.3.1, the market results status enables you to enhance two steps in your ETS API applications:

- **Stop having to rely on a convention to determine whether or not results are published**
  - with 3.3.1 the status transition indicates when results are published:
    - *Unavailable -> Preliminary or Final*
- **as soon as your application detects the switch to *Final*, it must stop retrieving market results** (results cannot be cancelled anymore)**.**

As a result the implementation logic using the 3.3.1 Market Results status can be simplified:



Please find below useful auction characteristics to implement these logics:

Auction	Auction Technical Name (in API, FTP files)	Theoretical publication time (CET)	Status of the first results	Max Cancellation Time in operational procedures	Max Cancellation Time For API apps (+10 min margin)
MRC	PWR-MRC-D+1	12:42	Preliminary	15:30	15:40
15 Call DE	PWR-15-call-DE-D+1	15:15	Final	15:45	15:55
30 Call GB	PWR-30-call-GB-D+1	16:45	Final	17:30	17:40
CH	PWR-CH-D+1	11:10	Final	14:30	14:40
CH/IT-IDA1	PWR-CH/IT-D+1	16:45	Preliminary	17:45	17:55
CH/IT-IDA2	PWR-CH/IT-D	11:30	Preliminary	11:45	11:55
GB - IDA1	PWR-SEM—GB-D+1	19:00	Preliminary	19:50	20:00
GB - IDA2	PWR-SEM—GB-D	09:30	Preliminary	10:20	10:30

Assignments inquiry requests

As the assignments (e.g. portfolios, delivery areas) of each API user will not change all the time, "retrieve information" requests must only be sent:

- after the start of the client application in order to retrieve the initial assignments,
- and then at a frequency that that does keep the server continuously busy, to remain aware of potential new assignments.

1. **RetrieveTradableAreaPortfolioInformations**
2. **RetrieveTradableAreaSets**
3. **RetrieveTradableAreas**
4. **RetrieveTradablePortfolios**
5. **RetrieveViewableAreaPortfolioInformations**
6. **RetrieveViewableAreas**
7. **RetrieveViewablePortfolios**

Order inquiry requests

API Applications may need to retrieve orders in order to:

1) Build a consolidated view of all orders submitted for a given auction

- For the next coming auction: such requests should be sent at a frequency that that does keep the server continuously busy.
- For past and further delivery days: the request frequency should be lower as for the next coming auction.

2) Check whether or not an order management request has been taken into account by ETS:

- It may happen that when the ETS system is busy your application receives the following error after an order action:
  - ErrorId = "OA 012", Error text = "Trading System did not answer within: [x] Seconds"
- In such an case it is likely that your request was processed but it is customers responsibility to check whether this is the case or not : your API application must send a Retrieve order request to check if the entry, the update or cancellation has been taken into account.
- In this context the request can be sent immediately after having received the error needing to retrieve orders

If this "server busy" error occurs when your application processes a loop of several orders actions, two implementation strategies are possible, with equal efficiency:

a) You handle the error immediately before continuing the loop with the next order action
b) Or you temporarily ignore the error and finish the loop to process remaining order actions, and only check the error afterwards at the end of the loop.

iii. Message Validation

In general, ETS API will validate messages. Invalid messages will be rejected. If an application is erroneous and sends invalid messages in short frequency, this could have the effect of a denial of service attack and can result in a suspension of the related application.

Client applications ensure that messages sent are valid according to the specified XML schema (WSDL). Messages sent with an invalid XML schema that cannot be processed by API server will be rejected using a native error response message (utf-8 message).

All mandatory message properties have to be set when sending a request to API server. ETS API server will reject messages with insufficient message properties.