API Manual ETPA

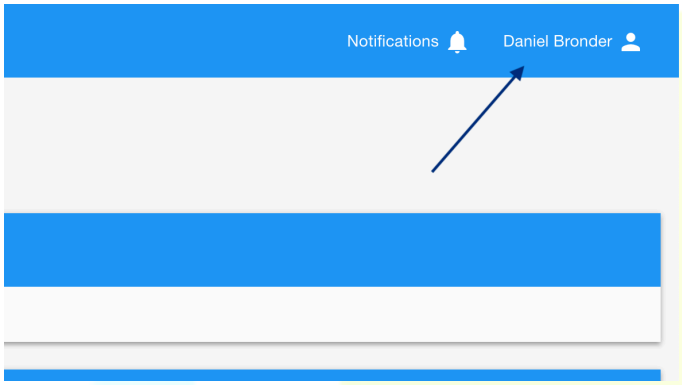
Table of Contents

1. Use of the API and WebSocket

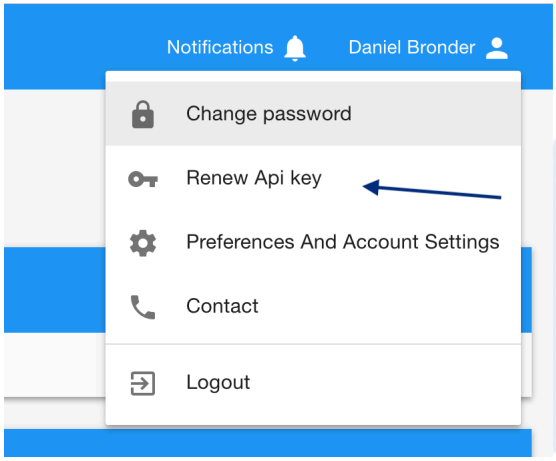
The API and WebSocket are two different ways to connect to the ETPA platform without using the Graphical User Interface (GUI) of the platform

1.2 Generate API Key

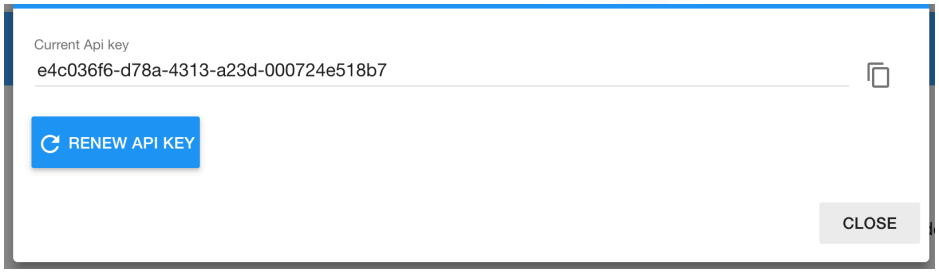
To use the API and WebSocket, you will need an API key. An API Key can be generated when you have an account. This can be done by login – click on name – renew API key.



Picture 1 - Generate API key 1/3



Picture 2 - Generate API key 2/3



Picture 3 - Generate API key 3/3

1.3 Acceptance Environment

Within ETPA there are two different environments that can be used. One of them is the acceptance (ACC) environment. The acceptance environment can be found here: https://acc-trading.etpa.nl. The acceptance environment is an environment where every participant and/ or broker can get familiar with the application, API and WebSocket. We highly recommend using the API and/or WebSocket of acceptance first before connecting your software to the production (PROD) environment. The URL for the API is: https://acc-trading.etpa.nl/public-api/1.0/electricity/[NODE] and for the WebSocket: wss://acc-trading.etpa.nl/public-api/1.0/electricity/websocket/[NODE]

On the acceptance environment you can get familiar with the API with Swagger interface: https://acc-trading.etpa.nl/swagger-ui.html#/ (this is only on ACC and not on PROD).

At ETPA we can provide a tradingbot for testing capabilities. To test your software in different situations and scenario's. Please get in contact with Jorrit Nijholt (+31 (0)6 28 50 28 06 or jorrit.nijholt@etpa.nl) or Daniël Bronder (+31 (0)6 10 86 42 86 or daniel.bronder@etpa.nl) to request the activation of the tradingbot and discuss the different options for testing.

1.4 Production Environment

When all the code of your software with our ACC environment is working as it should and as expected, the endpoint URL can be changed (declare it therefor at one place in your code) from the to the following PROD end-point:

API: https://trading.etpa.nl/public-api/1.0/electricity/[NODE]
WebSocket: wss://trading.etpa.nl/public-api/1.0/electricity/websocket/[NODE]

Make sure that for this environment you use the correct API key of a user that has a PROD account and the correct rights (user role). API key of ACC will not work in PROD.

2. The API

Within ETPA there are four different APIs that can be used. Certain account roles can only access certain APIs as shown in Table 1.

API/User	Trade	Reporting	Wallet	View only
Order	Yes	No	No	No
Trade	Yes	No	No	No
Reporting	Yes	Yes	Yes	No
User	Yes	Yes	Yes	No
Wallet	Yes	Yes	Yes	No

Table 1 - Access API

At some APIs some parameters are possible. These parameters are not required but could help with filtering the result.

Parameter	Type	Options	Description
My	Boolean	nothing, true or false	
Timeblock	String	nothing, BASELOAD, PEAK, OFFPEAK, INTRADAY or EXPOST	
Start	Integer		EPOCH time
End	Integer		EPOCH time
Filter (Reporting API)	String	nothing, CREATION or TRANSACTION	This can only be used if start and end are not empty
Filter (Trades API)	String	nothing, STARTED, END or EXECUTED	This can only be used if start and end are not empty

Table 2 - API parameters

Note: If the option is nothing the parameter should be empty.

2.2 Order API

The Order API can be used by an account who has trading rights. With the order API you are able to view all the orders, add order(s), get a specific order, edit an order and delete a specific order. The order API can be accessed with the following endpoint:

```
../orders
```

2.2.1 GET

With a get request you are able to get all the orders that are currently in the orderbook. With this request you are able to add two additional parameters. These are:

My and Timeblock (See Table 2)

2.2.2 POST

With the Post request you are able to send an order to the orderbook.

When you add and order with the API you can also add a call-back URL. This can be used to get the order ID of the order you submitted.

When you want to match an opposing order in the orderbook, you create a new order with the same price, time period and volume (or partial volume if needed) to match with it. The matching engine will match your order with the opposing order in the orderbook in the back-end.

An order will look like this:

```
{
  "allowedToBeUsedForIdcons": false,
  "ean": 871685920001768800,
  "end": 1465682400000,
  "metadata": {
    "key1": "value1",
    "key2": "value2"
  },
  "orderType": "BUY / SELL",
  "participantId": "292c3db8-3ee1-4999-bbed-d579225d1596",
  "price": 35,
  "quantity": 2,
  "start": 1465596000000,
  "timeblock": "INTRADAY / EXPOST / BASELOAD / PEAK / OFFPEAK"
}
```

NOTE: End/ Start times are in Epoch time (milliseconds). You can generate an epoch time from here: https://www.epochconverter.com/
NOTE: allowedToBeUsedForIdcons can only be used when the EAN is correct and the participant is able to create orders for IDCONS

Parameter	Description	Required
allowedToBeUsedForIdcons	If the order is allowed to be used for idcons purposes and only possibly when EAN is correct and the participant is allowed to create orders for IDCONS.	NO
ean	The ean code of the order	NO
end	The end time of the order in epoch time	YES
metadata	Additional information that can be used for internal administration	NO
ordertype	The type of the order: BUY or SELL	YES
participantId	The id the participant who	YES
price	The price of the order	YES
quantity	The quantity of the order	YES
start	The start time of the order in epoch time	YES
timeblock	The timeblock of the order: INTRADAY, EXPOST, BASELOAD, PEAK or OFFPEAK	YES

2.2.3 GET/{id}

With the Get/{id} you can get a specific order from the orderbook. The id is the id of the order.

2.2.4 PUT/{id}

With the PUT request you can edit a specific order. The id is the id of the order you would like to edit. You also have to add the order you have edited.

2.2.5 DELETE/{id}

This request will delete a specific order. To do this you will need to id of order you would like to delete.

2.3 Reporting API

The reporting API is used for reporting purposes. The reporting API can be used by users who have trade, wallet or reporting access. The endpoint for the reporting API is: **`../reporting.`**

2.3.1 Orders-trades

Within the Reporting API there are two options. The first option is to get all the orders and trades. The endpoint of this is: **`../orders-trades.`**

With this request you are able to add two additional parameters. These are:

Start, End and Filter (Reporting) *(see* Table 2*)*

2.3.2 Pv-party-participant

The second option within the reporting API is to retrieve all the trades from the participants from the pv-party. This enable pv-parties to get the trade information of their connected pv-party clients, no order information is given to the PV-party. The connection of participants to a PV-party is done by the ETPA admin in our system. The endpoint of this is: **`../pv-party-participants.`**

Start and End *(see* Table 2*)*

2.4 Trade API

The trade API can only be used by participants who have trading rights. The trading API has only a get request which will return trades. The endpoint of the trade API is: **`../trades.`**

With the trade API you could use four parameters to filter the trades. These parameters are:

Start, End, Filter (Trades) and My *(see* Table 2*)*

2.5 User API

The user API is the API which will give you all the information about an individual and all the information about the participants. The User API can be accessed by Trade, Report and Wallet users. The endpoint of the user API is as follows: **`../users.`**

2.5.1 Individual

This request will give you the information about you as an individual. To get the information about the individual add **`../individual`** to the endpoint of the API.

2.5.2 Participants

This request will give you the information of the participants that are representing you. To retrieve this information, you will need to add the **`../participants`** to the endpoint of the User API.

2.6 Wallet API

The Wallet API provides wallet data about a participant and it will give you the transactions. The Wallet can be used by participant who has reporting, trade or wallet access. The endpoint of the wallet API is: **`../wallets.`**

2.6.1 Balance/{id}

This get request will give you the current balance of a participant. The parameter is the id of the participant you would like to retrieve the balance from. You will need to add **`../balance/{id}`** to the endpoint of the wallet API.

2.6.2 Transactions/{id}

This get request will return all the transactions that have been made from a participant. The parameter is the id of the participant you would like to retrieve the transactions from. The endpoint for this request is: **`../transactions/{id}.`**

3. WebSocket

At ETPA we provide also WebSocket. This means you can automatically receive information about general information, orders and trades. The information about the WebSocket on acceptance can be found here as well: https://acc-trading.etpa.nl/#/socketInfo or for production: https://trading.etpa.nl/#/socketInfo.

To use the WebSocket on the acceptance environment you have to subscribe to the following URL: **`wss://acc-trading.etpa.nl/public-api/1.0/electricity/websocket`**

To use the WebSocket on the production environment you have to subscribe to the following URL: **`wss://trading.etpa.nl/public-api/1.0/electricity/websocket`**

For both the acceptance and production environment you will need to add your API key into the header of the request.

Implementing the normal GET orderbook calls is still recommended next to this WebSocket interface, due to possible internet connections errors (WebSockets message might therefore not be received at the client side). This will lead to a difference between the current know state of the orderbook, held by your system based on the WebSocket communication, and the real know state of the orderbook at the ETPA side. Some business logics based on these orderbook GET call will ensure that the systems stay in sync with each other at all times.

3.1 General

Within the general information there are two different channels. These are: **`/refresh`** and **`/queue/public`**

The **`/refresh`** channel lets the user know if the GUI needs to be refreshed.

The **`/queue/public`** channel lets the user know about all the information about orders and trades.

3.2 Orders

There are four channels which sends information about orders. These are: **`/state`**, **`/orderbook`**, **`/intradayorderbook`** and **`/expostorderbook`**.

The **`/state`** channel lets the user know if creating or modifying orders has been enabled or disabled. This channel will return either true or false.

The **`/orderbook`** channel lets the user know when a standard order has been created, cancelled or edited.

The **`/intradayorderbook`** channel lets the user know when an intraday order has been created, cancelled or edited.

The **`/expostorderbook`** channel lets the user know when an expost order has been created, cancelled or edited.

An order in the WebSocket will look like this:

```
{
 "id":"39d34519-6b3a-4a57-b3f8-ef42567d5593",
 "type":"INFO",
 "action":"ORDERS",
 "time":1480430752452,
 "message":null,
 "title":null,
 "order": {
     "id":"39d34519-6b3a-4a57-b3f8-ef42567d5593",
     "price":3,
     "quantity":1,
     "product":"ELECTRICITY",
     "timeblock":"BASELOAD",
     "type":"BUY",
     "start":1480460400000,
     "end":1480546800000,
     "participantId":"02866e9b-9359-4b68-b409-9473404b125e",
     "created":1480340400000,
     "priority":1480340400000
   }
 }
```

Code block 1 - Order from WebSocket

There are three different types for orders. These are: INFO, WARNING and REFRESH.

INFO: Order has been created
WARNING: Order has been cancelled
REFRESH: Order has been edited

3.3 Trades

There is only one channel which sends information about trades. This is the /trades channel. It will let the user know every time a trade is made. A trade in the WebSocket will look like this:

```
{
 "action":"TRADES",
 "id": "c456226f-5dba-4f40-bcde-7b3b9d027fbb",
 "message":null,
 "order":null,
 "time":1480430752452,
 "title":null,
 "trade": {
     "end":1480546800000,
     "excecuted":1480430752452
     "price":"3",
     "quantity":1,
     "start":1480460400000,
     "duration":24,
     "timeblock":"BASELOAD"
   },
 "type":"INFO",
 }
```

Code block 2 - Trade from WebSocket

4. FAQ (Frequently asked questions)

1. *Do you also need an API Key for the WebSocket?*

Yes, the API Key is used for authentication purposes.

2. Getting a 200 error while connecting to the correct WebSocket URL.

When you are using a different library, which doesn't support SockJS you'll need to add an extra /websocket to your URL. Now you will let the application know that you want to connect to WebSocket instead of the SockJS WebSocket connection.