nationalgridESO

EAC API guide

A guide to help you start using the EAC GraphQL API

Getting started

Authentication

All GraphQL API requests require a valid Bearer token.

First you need to acquire a token using a standard OAuth2 flow. More information on this step will be provided later.

In all requests, use the Authorization header to provide your Bearer token.

```
Authorization: "Bearer YOUR_TOKEN"
```

Postman as API client

Postman is an API platform for building and using APIs. Developers can use it as GraphQL client for testing.

NGESO will build for you a Postman Collection with some examples of queries. You will be notified once those examples will be published.

Development and testing

To help you build out your integration, the EAC will have a sandbox environment where you will be able to run tests.

In this sandbox environment authentication with be possible with a long-lived token.

More information on this sandbox environment will be provided later.

Terminology

Auctions

- An Auction Session is a single occurrence of an auction at a specific date and time consisting in the matching of sell and buy orders to return a selection of executed orders and market clearing prices.
- An Auction Session Log Item is an event related to an auction session.
- A Service Type is a categorization of the services, there are three types: Response, Quick Reserve, and Slow Reserve.
- A Service represent an Ancillary Service procured by the Buyer. For example, Dynamic Containment High and Dynamic Containment Low.
- A Service Window is a pre-defined delivery period of a service.

Baskets

- A Parent Order is a non-curtailable order whose acceptance is a pre-condition to the acceptance of one or more other (child) orders in the same Basket. A parent order may not have any linked child orders. A parent order has a single price and can offer a volume for each eligible service.

- A Child Order is an order that can only be accepted if another order to which it is linked, the “parent order” is also accepted (i.e., the parent order can be accepted alone or the parent and child orders can be accepted together, but the child order cannot be accepted alone). A child order has a single price and can offer a volume for each eligible service.
- A Substitutable Child Order is a normal child order with an additional rule. Substitutable child orders of the same basket are exclusive, the sum of their percentage of acceptance is limited to 100%. So, if one is fully accepted, the other must be fully rejected. But four substitutable child orders may also be accepted at 25% each.
- A Basket is a grouping of orders, it contains exactly one parent order and can in addition contains up to 10 child orders and 10 non-substitutable child orders (so up to 20 children in total).
- A Basket family is a grouping of non-concomitant baskets. Baskets belonging to the same family are looped together meaning that they will all be accepted or rejected together. Baskets with the same family name belong to the same family. A basket can belong to a single family.

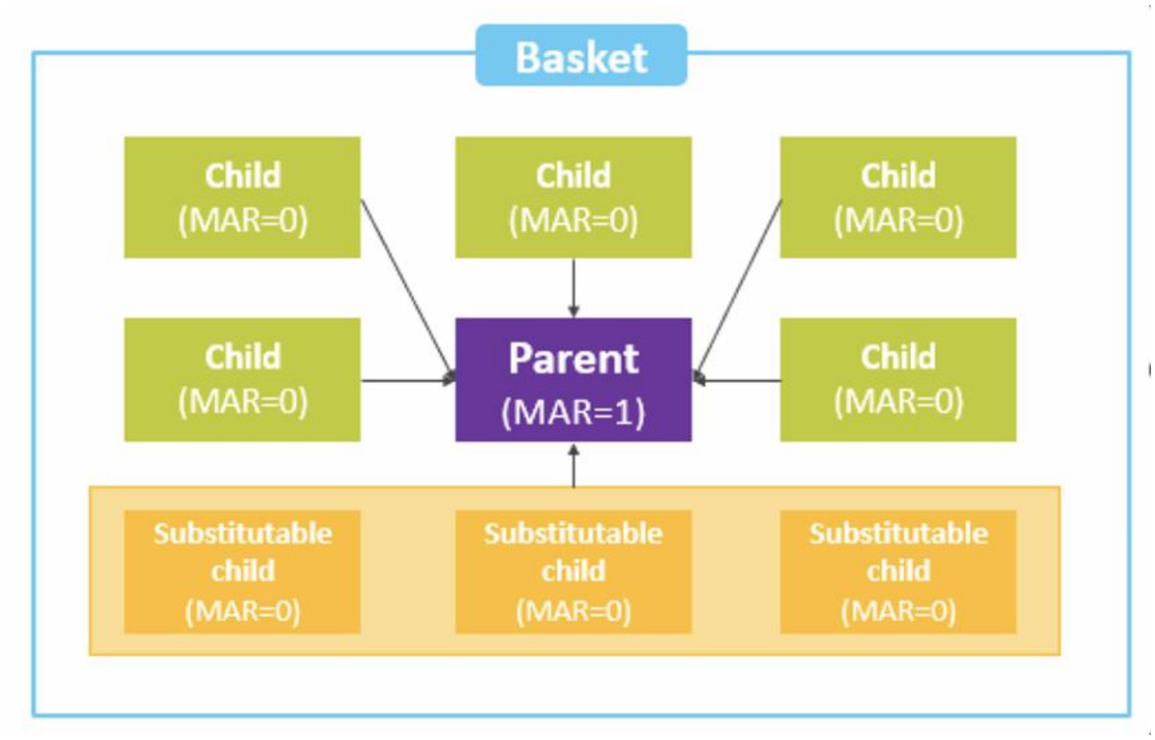
Units

- A Unit is a collection of one or several assets entitled to offer capacity.

Baskets

Basket structure

A basket is a grouping of sell orders of a unit, for one service type (Response, Quick Reserve, Slow Reserve) and is defined on exactly one service window. It must contain exactly one parent order and may contain up to 10 child orders and 10 substitutable child orders (so up to 20 children in total).



They are used to model mutual exclusivity between sets of orders. Any two baskets are mutually exclusive if they are defined on the same service window (or on service windows that overlap in any time period).

Baskets can be looped to other baskets that are defined on different service windows (baskets cannot be looped together if they are mutually exclusive). Baskets are looped together by assigning to them the same family name.

Create a basket

A Basket is linked to a given Unit, Auction Session, Service Type, and Service Window. You first need to retrieve those objects to create a Basket.

Fetch an auction session

First you need to retrieve the Auction Session for which you want to create a Basket. Use the auctionSessions query to retrieve the list of all Auction Sessions you have access to. The size of the list returned will be limited as it will contains only auctions of a rolling time horizon.

Use for example the following GraphQL query to retrieve the Auction Sessions.

```
query AuctionSessionsPerDate {
  auctionSessions    {
    ... on AuctionSession {
      identifier { id }
      status
      startDate
      startTime
      serviceWindows {
        serviceType {
          identifier { externalID }
        }
        identifier { id }
        name
      }
    }
    ... on TradeError {
      message
    }
  }
}
```

See below an example of the response you will get from the EAC with this query. A single Auction Session is described in this example for readability, in practice multiple ones will be returned.

```
{
  "data": {
    "auctionSessions": [
      {
        "identifier": {
          "id": 59
        },
        "status": "FINISHED",
        "startDate": "2023-05-09",
        "startTime": "13:30:00",
        "serviceWindows": [
          {
            "serviceType": {
              "identifier": {
                "externalID": "Response"
              }
            },
            "identifier": {
              "id": 1864

            },
            "name": "EFA 1"
          },
            {
              "serviceType": {
                "identifier": {
                  "externalID": "Response"
                }
              },
              "identifier": {
                "id": 1865
              },
              "name": "EFA 2"
            },
            {
              "serviceType": {
                "identifier": {
                  "externalID": "Response"
                }
              },
              "identifier": {
                "id": 1866
              },
              "name": "EFA 3"
            },
            {
              "serviceType": {
                "identifier": {
                  "externalID": "Response"
                }
              },
              "identifier": {
                "id": 1867
              },
              "name": "EFA 4"
            },
            {
              "serviceType": {
                "identifier": {
                  "externalID": "Response"
                }
              },
              "identifier": {
                "id": 1868
              },
              "name": "EFA 5"
            },
            {
              "serviceType": {
                "identifier": {
                  "externalID": "Response"
                }
              },
              "identifier": {
                "id": 1869
              },
              "name": "EFA 6"
            }
          ]
        }
      ]
    }
}
```

Fetch your units

You will also need to retrieve the list of your Units registered into the SMP (Single Market Platform).

Use for example the following query.

```
query Units {
  units {
    ... on Unit {
      identifier {
        externalID
      }
      name
    }
    ... on TradeError {
      message
    }
  }
}
```

You will receive a response looking like this.

```
{
  "data": {
    "units": [
      {
        "identifier": {
          "externalID": "414a8de4-852c-45ba-b73f-643d88d2bec1"
        },
        "name": "AG-GEDF02"
      }
    ]
  }
}
```

Fetch available services

With the Auction Session retrieved previously, you have access to the list of Service Windows of the auction. Each Service Window belongs to a specific Service Type. To create a Basket, you will also need the list of Services belonging to each Service Type.

To retrieve all Services, you can use the following query.

```
query Services {
  services {
    ... on Service {
      identifier {
        externalID
      }
      name
      serviceType {
        identifier {
          externalID
        }
      }
    }
  }
}
```

The list of the 10 products will be returned, including 6 response products, 2 quick reserve products and 2 slow reserve products. Here is an example with only 2 products.

```
{
  "data": {
    "services": [
      {
        "identifier": {
          "externalID": "Dynamic Containment HF"
        },
        "name": "Dynamic Containment HF",
        "serviceType": {
          "identifier": {
            "externalID":"Response"
          }
        }
      },
      {
        "identifier": {
          "externalID": "Dynamic Containment LF"
        },
        "name": "Dynamic Containment LF",
        "serviceType": {
          "identifier": {
            "externalID":"Response"
          }
        }
      }
    ]
  }
}
```

Submit your basket

Use the enterBaskets mutation to create one or multiple baskets at once. For each basket, some information will be required, let us go through it step by step.

a. General information

First you need to provide some general information. You can define a name for you Basket and you need to select a Unit and a Service Window. Note that a Service Window is linked to a given Service Type. All orders of Basket will need to use only Services of this Service Type that are available for this Unit.

```
name: "Basket EFA 1",
unit: {
  externalID: "414a8de4-852c-45ba-b73f-643d88d2bec1"
},
serviceWindow: {
  id: 1864
},
```

b. Parent order

Then you need to provide a parent order. Give it a price and positive volumes for the services of the Basket's service type for which you want to bid. The list of volumes may be empty, but you can have at most one item per service.

```
parent: {
  price: 6,
  serviceVolumes: [
    {
      service: { externalID: "Dynamic Containment LF" },
      volume: 5
    }
  ]
}
```

c. Child orders

Optionally, you can also provide some child orders. As for the parent order, give them a price and positive volumes for the services of the Basket's service type for which you want to bid. Each child order must have at least one non-zero volume and at most one item per service. Up to 10 substitutable and up to 10 non-substitutable child orders are allowed.

```
children: [
  {
    price: 13,
    serviceVolumes: [
      {
        service: { externalID: "Dynamic Containment HF" },
        volume: 1
      },
      {
        service: { externalID: "Dynamic Containment LF" },
        volume: 1
      }
    ],
    substitutability: false
  },
  {
    price: 13,
    serviceVolumes: [
      {
        service: { externalID: "Dynamic Containment LF" }
        volume: 2
      }
    ]
    substitutability: true
  }
]
```

d. Complete example

All together it gives us the following query.

```
mutation enterBaskets {
  enterBaskets( baskets: [
    {
      name: "Basket EFA 1",
      unit: {
        externalID: "414a8de4-852c-45ba-b73f-643d88d2bec1"
      },
      serviceWindow: {
        id: 1864
      },
      parent: {
        price: 6,
          serviceVolumes: [
            {
              service: { externalID: "Dynamic Containment LF" },
              volume: 5
            }
          ]
        }
        children: [
          {
            price: 13,
            serviceVolumes: [
              {
                service: { externalID: "Dynamic Containment HF" },
                volume: 1
              },
              {
                service: { externalID: "Dynamic Containment LF" },
                volume: 1
              }
            ],
            substitutability: false
          },
          {
            price: 13,
            serviceVolumes: [
              {
                service: { externalID: "Dynamic Containment LF" }
                volume: 2
              }
            ]
            substitutability: true
          }
        ]
      }
    ])  {
      ... on BasketsEntryConfirmation {
        baskets {
          identifier { id }
        }
      }
      ... on TradeError {
        message
      }
    }
}
```

In response, you will receive a unique identifier for the created Basket. This identifier can later be used to retrieve, update or delete the Basket.

```
{
  "data": {
    "enterBaskets": {
      "baskets": [
        {
          "identifier": {
            "id": 16
          }
        }
      ]
    }
  }
}
```