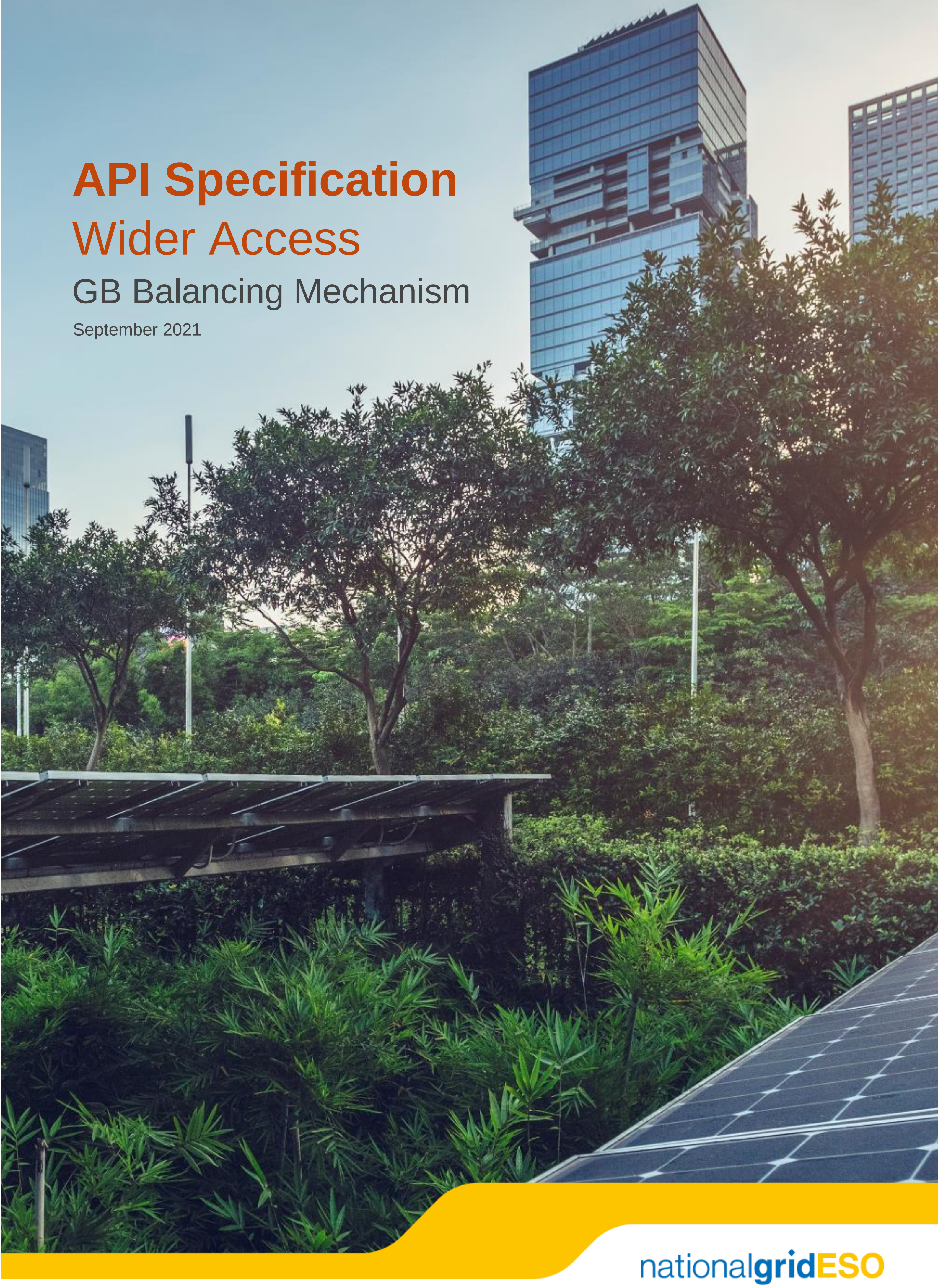
API Specification
Wider Access

GB Balancing Mechanism

September 2021



nationalgridESO

Contents

Introduction

Overview

This document aims to give a closer look at the APIs which the National Grid Electricity System Operator (NGESO) has in put place, to enable access to the GB Balancing Mechanism (BM) for small generating units (BMUs), via the Wider Access Application Programming Interface (WA API).

NGESO has embraced the Internet of Things ethos, making the WA API available to market participants. This enables faster, more flexible connections to the BM. This in turn offers a reduced cost to end-consumers as a consequence of establishing new connections. All new small BM participants can connect directly to the new API infrastructure. However, they may also opt to use an intermediate hosting service, provided by a preferred commercial vendor. The API solution is one of the deliverables for the overall Wider Access initiative.

NGESO now offers two options for dynamic exchange of data:

Connections of new private circuits using NGESO's telecommunications network provider via traditional, fixed-line technology

Connection to the WA API infrastructure using web services and internet-based connectivity.

This document aims to give an overview of the APIs that are published by the WA API infrastructure, to give smaller BM participants a detailed understanding of the requirements to join the BM via this route. Both WA API and Private Circuits provide functional equivalence in terms of Electronic Data Transfer (EDT) and Dispatch & Logging (EDL).

Further information

More information about Wider Access to the BM and connection via the API is available on the Balancing Mechanism Wider Access page of the NGESO website.

To discuss opportunities offered by Wider Access and the API, please contact NGESO via your account manager or email Commercial.Operation@nationalgrideso.com.

APIs

NGESO_Submission

Overview

The Submission API allows the Market Participants to submit Physical Notifications (PN), Quiescent Physical Notifications (QPN), Bids & Offers (BOD), & Maximum Export/Import Limits (MEL/MIL), before the Balancing Mechanism Gate is closed. It is the principal mechanism by which participants in the existing Pool submit their offer data to NGESO.

Day-ahead Dynamic Parameters have been removed from the Grid Code and are not used by National Grid. For the purposes of backwards compatibility, Trading Agents may still submit day ahead Dynamic Parameters by EDT and these will be accepted by National Grid without any validation or consistency checks.

Notification Time

The System Time of the Host shall be in GMT and shall be kept referenced to a recognised global time base. It is this time, which shall determine gate closure for submissions. Each invocation successfully transferred to the Host will be deemed to be a submission. The notification time of this file, and hence all data contained within it, shall be deemed to be the point in time that the submission was made.

Submission Acknowledgment

The acknowledgement message will contain the notification time of the submission processed by BM. The notification time will use the standard Time format defined in the Convention paragraph.

Submission Acceptance

Acceptance are produced once a submission request has been validated in its entirety. The acceptance response will contain a list of all BMUs for which all submitted data has passed formatting, consistency and validation tests.

Submission Rejection

Submission rejection are also produced once a submission has been validated in its entirety. Each record contained is checked for formatting, validity and consistency. Should formatting prove incorrect the request will not proceed to validation and will be rejected at that stage. Thus, a record that has invalid data and is also incorrectly formatted for the type of data will only have a message stating that it was rejected owing to a formatting error. The validity of the record will not be considered. Once a record has completed and passed formatting checks, it will be checked against each applicable validation and consistency rule. Any and all of these failures will be reported individually for each submitted record. Hence a single row that does not comply with multiple validation or consistency rules, will give rise to multiple error messages within a reject request.

Compression process

In the case of a large payload, compression can be utilised following the steps below.

1. Compress the payload by using gZip (more information at https://www.gzip.org/)
2. Signing the compressed payload from (1)
3. Including the additional **x-compress** field in the header part
4. Sending the following wrapped payload including the compressed payload from (1) in the **data** field

```
{
  “mpid”: “Market Participant ID”,
  “number”: “Sequence Number”,
  “data”: “Compressed Payload”
}
```

The **x-compress** field in the header can have the following states;

- **yes**, when the payload being sent is wrapped.
- **no**, when the payload being sent is the normal one (as stated below). If the **x-compress** field is not included in the header, the request will be considered as not compressed

Please note that when a wrapped payload is sent;

- it is not necessary to normalize the payload before compressing it
- the Acceptance/Rejection response from NGESO will be in the same format while the Acknowledgment will be not compressed. As a consequence, it is necessary to uncompress the received payload (in the **data** field) by using gZip.

Submission examples

Example 1 Submission of all data types for 1 BMU

```
{
    "sequence": "1099",
    "tradingAgent": "TR_AGT",
    "BMUSubmissionElements": [
      {
        "bmUnitName": "BM_UNIT_1",
        "pn": [
            {
              "timeFrom": "2018-10-31 18:30",
              "levelFrom": "10",
              "timeTo": "2018-10-31 19:00",
              "levelTo": "20"
          }
        ],
        "qpn": [
          {
              "timeFrom": "2018-10-31 18:30",
              "levelFrom": "-15",
              "timeTo": "2018-10-31 19:00",
              "levelTo": "0"
            }
        ],
        "bod": [
            {
              "timeFrom": "2018-10-31 18:30",
              "timeTo": "2018-10-31 19:00",
              "pairNumber": "1",
              "levelFrom": "100",
              "levelTo": "100",
              "offerPrice": "13.00",
              "bidPrice": "8.00"
            },
            {
              "timeFrom": "2018-10-31 18:30",
              "timeTo": "2018-10-31 19:00",
              "pairNumber": "-1",
              "levelFrom": "-100",
              "levelTo": "-100",
              "offerPrice": "13.00",
              "bidPrice": "8.00"
            }
        ],
        "mel": [
            {
              "timeFrom": "2018-10-31 18:30",
              "maximumExportLevelFrom": "0",
              "timeTo": "2018-10-31 19:00",
              "maximumExportLevelTo": "9999"
            }
        ],
        "mil": [
            {
              "timeFrom": "2018-10-31 18:30",
              "maximumImportLevelFrom": "-9999",
              "timeTo": "2018-10-31 19:00",
              "maximumImportLevelTo": "0"
            }
        ],
        "rure": [
            {
              "effectiveTime": "2018-10-31 19:00",
              "rate1": "15.0",
              "elBow2": "140",
              "rate2": "3.4",
              "elBow3": "145",
              "rate3": "12.7"
            }
        ],
        "ruri": [
            {
              "effectiveTime": "2018-10-31 19:00",
              "rate1": "010.0",
              "elBow2": "-0340",
              "rate2": "015.0",
              "elBow3": "-140",
              "rate3": "15.0"
            }
        ],
        "rdre": [
            {
              "effectiveTime": "2018-10-31 19:00",
              "rate1": "015.0",
              "elBow2": "+0140",
              "rate2": "015.0",
              "elBow3": "+0145",
              "rate3": "015.0"
            }
        ],
        "rdri": [
            {
"effectiveTime": "2018-10-31 19:00",
"rate1": "015.0",
"elBow2": "-0140",
"rate2": "015.0",
"elBow3": "-0140",
"rate3": "015.0"
            }
        ],
        "ndz": [
            {
"effectiveTime": "2018-10-31 19:00",
"timeValue": "30"
```

```
            }
        ],
        "nto": [
            {
"effectiveTime": "2018-10-31 19:00",
"timeValue": "59"
            }
        ],
        "ntb": [
            {
"effectiveTime": "2018-10-31 19:00",
"timeValue": "59"
            }
        ],
        "mzt": [
            {
"effectiveTime": "2018-10-31 19:00",
"timeValue": "999"
            }
        ],
        "mnzt": [
            {
"effectiveTime": "2018-10-31 19:00",
"timeValue": "999"
            }
        ],
        "sel": [
            {
"effectiveTime": "2018-10-31 19:00",
"MWlevel": "9999"
            }
        ],
        "sil": [
            {
"effectiveTime": "2018-10-31 19:00",
"MWlevel": "-9999"
            }
        ],
        "mdvp": [
            {
"effectiveTime": "2018-10-31 19:00",
"MDV": "99999",
"MDP": "239"
            }
        ],
        "rrb": [
            {
              "TimeFrom": "2018-10-31 18:30",
              "Direction": "UP",
              "MaxLevel": "1000",
              "MinLevel": "0",
              "Divisible": "TRUE",
              "Price": "13.00",
              "BidID": "ABCDEFGHI",
              "AssociatedBidType": "LINK",
              "AssociatedBidSet": "ABCDEFGHI"
            }
        ]
    }
    ]
  }
```

Example 2 Submission of multiple PN for 1 BMU

```
{
    "sequence": "1099",
    "tradingAgent": "TR_AGT",
    "BMUSubmissionElements": [
        {
            "bmUnitName": "BM_UNIT_1",
            "pn": [
                {
                  "timeFrom": "2018-10-31 18:30",
                  "levelFrom": "10",
                  "timeTo": "2018-10-31 19:00",
                  "levelTo": "20"
                },
                {
                  "timeFrom": "2018-10-31 19:30",
                  "levelFrom": "30",
                  "timeTo": "2018-10-31 20:00",
                  "levelTo": "40"
                }
            ]
        }
    ]
}
```

Example 3 Submission of multiple PN for multiple BMUs

```
{
    "sequence": "1099",
    "tradingAgent": "TR_AGT",
    "BMUSubmissionElements": [
        {
            "bmUnitName": "BM_UNIT_1",
            "pn": [
                {
                  "timeFrom": "2018-10-31 18:30",
                  "levelFrom": "10",
                  "timeTo": "2018-10-31 19:00",
                  "levelTo": "20"
                },
                {
                  "timeFrom": "2018-10-31 19:30",
                  "levelFrom": "30",
                  "timeTo": "2018-10-31 20:00",
                  "levelTo": "40"
                }
            ]
        },
        {
            "bmUnitName": "BM_UNIT_2",
            "pn": [
                {
                  "timeFrom": "2018-10-31 18:30",
                  "levelFrom": "10",
                  "timeTo": "2018-10-31 19:00",
                  "levelTo": "20"
                },
                {
                  "timeFrom": "2018-10-31 19:30",
                  "levelFrom": "30",
                  "timeTo": "2018-10-31 20:00",
                  "levelTo": "40"
                }
            ]
        }
    ]
}
```

POST **/submission** Add a new submission [Market Participant -> NGESO]

Request header and responses will be provided during the onboarding process.

Models

Submission Request data {

sequence*	**string** *pattern: ^\d{1,4}$* *example: 1099*
tradingAgent*	**string** *minLength: 1* *maxLength: 9* *example: TR_AGT*
BMUSubmissionElements*	**[. . .]**

}

BMU Submission Element {

bmUnitName*	**string** *title: Unit Name* *example: BM_UNIT_1* *minLength: 1* *maxLength: 9*
pn	**[. . .]**
qpn	**[. . .]**
bod	**[. . .]**
mil	**[. . .]**
rure	**[. . .]**
ruri	**[. . .]**
rdre	**[. . .]**
rdri	**[. . .]**
ndz	**[. . .]**
nto	**[. . .]**
ntb	**[. . .]**
mzt	**[. . .]**

mnzt	**[. . .]**
sel	**[. . .]**
sil	**[. . .]**
mdvp	**[. . .]**
rrb	**[. . .]**

}

Physical Notification {

timeFrom*	**string** *pattern: ^(2([0-9]{3}))[-](0[1-9]|10|11|12)[-](0[1-9]|[12]\d|3[01])[ ](([0-1][0-9])|(2[0-3]))[:]([0-5][0-9])$* *example: 2018-10-31 18:30*
levelFrom*	**string** *pattern: ^([+-]?\d{1,4})$* *example: 10*
timeTo*	**string** *pattern: ^(2([0-9]{3}))[-](0[1-9]|10|11|12)[-](0[1-9]|[12]\d|3[01])[ ](([0-1][0-9])|(2[0-3]))[:]([0-5][0-9])$* *example: 2018-10-31 19:00*
levelTo*	**string** *pattern: ^([+-]?\d{1,4})$* *example: 20*

}

Quiescent Physical Notification {

timeFrom*	**string** *pattern: ^(2([0-9]{3}))[-](0[1-9]|10|11|12)[-](0[1-9]|[12]\d|3[01])[ ](([0-1][0-9])|(2[0-3]))[:]([0-5][0-9])$* *example: 2018-10-31 18:30*
levelFrom*	**string** *pattern: ^(([-]\d{1,4})|[0])$* *example: -15*
timeTo*	**string** *pattern: ^(2([0-9]{3}))[-](0[1-9]|10|11|12)[-](0[1-9]|[12]\d|3[01])[ ](([0-1][0-9])|(2[0-3]))[:]([0-5][0-9])$* *example: 2018-10-31 19:00*
levelTo*	**string** *pattern: ^(([-]\d{1,4})|[0])$* *example: 0*

}

Bid-Offer Data {

timeFrom*	**string** *pattern: ^(2([0-9]{3}))[-](0[1-9]|10|11|12)[-](0[1-9]|[12]\d|3[01])[ ](([0-1][0-9])|(2[0-3]))[:]([0-5][0-9])$* *example: 2018-10-31 18:30*
timeTo*	**string** *pattern: ^(2([0-9]{3}))[-](0[1-9]|10|11|12)[-](0[1-9]|[12]\d|3[01])[ ](([0-1][0-9])|(2[0-3]))[:]([0-5][0-9])$* *example: 2018-10-31 19:00*
pairNumber*	**string** *pattern: ^[+-]?[1-5]{1}$* *example: 1*
levelFrom*	**string** *pattern: ^([+-]?\d{1,4})$* *example: 100*
levelTo*	**string** *pattern: ^([+-]?\d{1,4})$* *example: 100*
offerPrice*	**string** *pattern: (^[+-]?\d{1,5}[.]\d{2})$* *example: 13.00*
bidPrice*	**string** *pattern: (^[+-]?\d{1,5}[.]\d{2})$* *example: 8.00*

}

Maximum Export Limit {

timeFrom*	**string** *pattern: ^(2([0-9]{3}))[-](0[1-9]|10|11|12)[-](0[1-9]|[12]\d|3[01])[ ](([0-1][0-9])|(2[0-3]))[:]([0-5][0-9])$* *example: 2018-10-31 18:30*
maximumExportLevelFrom*	**string** *pattern: ^([+]?\d{1,4})$* *example: 0*
timeTo*	**string** *pattern: ^(2([0-9]{3}))[-](0[1-9]|10|11|12)[-](0[1-9]|[12]\d|3[01])[ ](([0-1][0-9])|(2[0-3]))[:]([0-5][0-9])$* *example: 2018-10-31 19:00*
maximumExportLevelTo*	**string** *pattern: ^([+]?\d{1,4})$* *example: 9999*

}

Maximum Import Limit {

timeFrom*	**string** *pattern: ^(2([0-9]{3}))[-](0[1-9]|10|11|12)[-](0[1-9]|[12]\d|3[01])[ ](([0-1][0-9])|(2[0-3]))[:]([0-5][0-9])$* *example: 2018-10-31 18:30*
maximumImportLevelFrom*	**string** *pattern: ^([-]\d{1,4}|[0])$* *example: -9999*
timeTo*	**string** *pattern: ^(2([0-9]{3}))[-](0[1-9]|10|11|12)[-](0[1-9]|[12]\d|3[01])[ ](([0-1][0-9])|(2[0-3]))[:]([0-5][0-9])$* *example: 2018-10-31 19:00*
maximumImportLevelTo*	**string** *pattern: ^([-]\d{1,4}|[0])$* *example: 0*

}

Run Up Rate Export {

effectiveTime*	**string** *pattern: ^(2([0-9]{3}))[-](0[1-9]|10|11|12)[-](0[1-9]|[12]\d|3[01])[ ](([0-1][0-9])|(2[0-3]))[:]([0-5][0-9])$* *example: 2018-10-31 19:00*
rate1*	**string** *pattern: ^([+]?\d{1,3}[.]\d{1})$* *example: 15.0*
elBow2	**string** *pattern: ^([+]?\d{1,4})$* *example: 140*
rate2	**string** *pattern: ^([+]?\d{1,3}[.]\d{1})$* *example: 3.4*
elBow3	**string** *pattern: ^([+]?\d{1,4})$* *example: 145*
rate3	**string** *pattern: ^([+]?\d{1,3}[.]\d{1})$* *example: 12.7*

}

Run Up Rate Import {

effectiveTime*	**string** *pattern: ^(2([0-9]{3}))[-](0[1-9]|10|11|12)[-](0[1-9]|[12]\d|3[01])[ ](([0-1][0-9])|(2[0-3]))[:]([0-5][0-9])$* *example: 2018-10-31 19:00*
rate1*	**string** *pattern: ^([+]?\d{1,3}[.]\d{1})$* *example: 010.0*
elBow2	**string** *pattern: ^([-]\d{1,4})$* *example: -0340*
rate2	**string** *pattern: ^([+]?\d{1,3}[.]\d{1})$* *example: 015.0*
elBow3	**string** *pattern: ^([-]\d{1,4})$* *example: -140*
rate3	**string** *pattern: ^([+]?\d{1,3}[.]\d{1})$* *example: 15.0*

}

Run Down Rate Export {

effectiveTime*	**string** *pattern: ^(2([0-9]{3}))[-](0[1-9]|10|11|12)[-](0[1-9]|[12]\d|3[01])[ ](([0-1][0-9])|(2[0-3]))[:]([0-5][0-9])$* *example: 2018-10-31 19:00*
rate1*	**string** *pattern: ^([+]?\d{1,3}[.]\d{1})$* *example: 015.0*
elBow2	**string** *pattern: ^([+]?\d{1,4})$* *example: +0140*
rate2	**string** *pattern: ^([+]?\d{1,3}[.]\d{1})$* *example: 015.0*
elBow3	**string** *pattern: ^([+]?\d{1,4})$* *example: +0145*
rate3	**string** *pattern: ^([+]?\d{1,3}[.]\d{1})$* *example: 015.0*

}

Run Down Rate Import {

effectiveTime*	**string** *pattern: ^(2([0-9]{3}))[-](0[1-9]|10|11|12)[-](0[1-9]|[12]\d|3[01])[ ](([0-1][0-9])|(2[0-3]))[:]([0-5][0-9])$* *example: 2018-10-31 19:00*
rate1*	**string** *pattern: ^([+]?\d{1,3}[.]\d{1})$* *example: 015.0*
elBow2	**string** *pattern: ^([-]\d{1,4})$* *example: -0140*
rate2	**string** *pattern: ^([+]?\d{1,3}[.]\d{1})$* *example: 015.0*
elBow3	**string** *pattern: ^([-]\d{1,4})$* *example: -0140*
rate3	**string** *pattern: ^([+]?\d{1,3}[.]\d{1})$* *example: 015.0*

}

Notice to Deviate From Zero {

effectiveTime*	**string** *pattern: ^(2([0-9]{3}))[-](0[1-9]|10|11|12)[-](0[1-9]|[12]\d|3[01])[ ](([0-1][0-9])|(2[0-3]))[:]([0-5][0-9])$* *example: 2018-10-31 19:00*
timeValue*	**string** *pattern: ^\d{1,3}$* *example: 30*

}

Notice to Deliver Offers {

effectiveTime*	**string** *pattern: ^(2([0-9]{3}))[-](0[1-9]|10|11|12)[-](0[1-9]|[12]\d|3[01])[ ](([0-1][0-9])|(2[0-3]))[:]([0-5][0-9])$* *example: 2018-10-31 19:00*
timeValue*	**string** *pattern: ^\d{1,2}$* *maximum: 59*

	example: 59

}

Notice to Deliver Bids {

effectiveTime*	**string** *pattern: ^(2([0-9]{3}))[-](0[1-9]|10|11|12)[-](0[1-9]|[12]\d|3[01])[ ](([0-1][0-9])|(2[0-3]))[:]([0-5][0-9])$* *example: 2018-10-31 19:00*
timeValue*	**string** *pattern: ^\d{1,2}$* *maximum: 59* *example: 59*

}

Minimum Zero Time {

effectiveTime*	**string** *pattern: ^(2([0-9]{3}))[-](0[1-9]|10|11|12)[-](0[1-9]|[12]\d|3[01])[ ](([0-1][0-9])|(2[0-3]))[:]([0-5][0-9])$* *example: 2018-10-31 19:00*
timeValue*	**string** *pattern: ^\d{1,3}$* *example: 999*

}

Minimum Non Zero Time {

effectiveTime*	**string** *pattern: ^(2([0-9]{3}))[-](0[1-9]|10|11|12)[-](0[1-9]|[12]\d|3[01])[ ](([0-1][0-9])|(2[0-3]))[:]([0-5][0-9])$* *example: 2018-10-31 19:00*
timeValue*	**string** *pattern: ^\d{1,3}$* *example: 999*

}

Stable Export Limit {

effectiveTime*	**string** pattern: ^(2([0-9]{3}))[-](0[1-9]|10|11|12)[-](0[1-9]|[12]\d|3[01])[ ](([0-1][0-9])|(2[0-3]))[:]([0-5][0-9])$ example: 2018-10-31 19:00
MWlevel*	**string** pattern: ^\d{1,4}$ example: 9999

}

Stable Import Limit {

effectiveTime*	**string** pattern: ^(2([0-9]{3}))[-](0[1-9]|10|11|12)[-](0[1-9]|[12]\d|3[01])[ ](([0-1][0-9])|(2[0-3]))[:]([0-5][0-9])$ example: 2018-10-31 19:00
MWlevel*	**string** pattern: ^([-]\d{1,4})$ example: -9999

}

Maximum Delivery Volume and Period {

effectiveTime*	**string** pattern: ^(2([0-9]{3}))[-](0[1-9]|10|11|12)[-](0[1-9]|[12]\d|3[01])[ ](([0-1][0-9])|(2[0-3]))[:]([0-5][0-9])$ example: 2018-10-31 19:00
MDV	**string** title: Max Delivery Volume (MW hours) pattern: ^([-+]?\d{1,5})$ example: 99999
MDP	**string** title: Max. Delivery Period (minutes) pattern: ^\d{1,3}$ example: 239

}

RR Bid {

TimeFrom*	**string** *pattern: ^(2([0-9]{3}))[-](0[1-9]|10|11|12)[-](0[1-9]|[12]\d|3[01])[ ](([0-1][0-9])|(2[0-3]))[:]([0-5][0-9])$* *example: 2018-10-31 18:30*
Direction*	**string** *example: UP* Enum: Array [2]
MaxLevel*	**string** *pattern: ^[+-]?\d{1,9}$* *example: 1000*
MinLevel	**string** *pattern: ^[+-]?\d{1,9}$* *example: 0*
Divisible*	**string** Enum: Array [2]
Price*	**string** *pattern: ^[+-]?\d{1,5}.\d{2}$* *example: 13.00*
BidID	**string** *pattern: ^\w{9}$* *example: ABCDEFGHI*
AssociatedBidType	**string** *example: LINK* Enum: Array [3]
AssociatedBidSet	**string** *pattern: ^\w{9}$* *example: ABCDEFGHI*

}

Successful Submission Response {

message*	**string** *example: Successful request*
version*	**string** *example: 1.0*

}

Error payload {

message*	**string** *example: Error message*

version*	**string** *example: 1.0*
code*	**string** *example: 400*
detail	**string** *example: Error information*

}

NGESO_Redeclaration

Overview

Redeclaration of availability and dynamic parameters to NGESO can be done by using this API. The redeclaration undergoes syntax and validation checking.

If the submission is valid, a successful technical acknowledgement will be returned to the Market Participant. If an error is encountered, a technical error will be sent.

The following data can be submitted;

- Maximum Export/Import Limit (MIL/MEL)
- Run Up/Down Rate Export (RURE/RDRE)
- Notice to Deviate From Zero (NDZ)
- Stable Export/Import Limit(SEL/SIL)
- Minimum Zero Time (MZT)
- Minimum Non Zero Time (MNZT)
- Run Up/Down Rate Import (RURI/RDRI)
- Notice to Deliver Offers/Bids(NTO/NTB)
- Maximum Delivery Volume (MDVP)

Please note that only one of the redeclaration data type above can be submitted. For example, a redeclaration payload for MEL will be as below.

```
{
    "controlPoint": "XX_YY",
    "bmUnitName": "XX-YYY45",
    "logTime": "18-OCT-2018 06:00",
    "refNumber": "10584466",
    "BMURedeclarationElements": [
        {
            "mel": {
                "timeFrom": "18-OCT-2018 06:00",
                "maximumExportLevelFrom": "0",
                "timeTo": "18-OCT-2018 06:30",
                "maximumExportLevelTo": "9999"
            }
        }
    ]
}
```

POST **/redeclaration** Add a new redeclaration [Market Participant -> NGESO]

Request header and responses will be provided during the onboarding process.

Models

Redeclaration Data	{
controlPoint*	**string** *minLength: 1* *maxLength: 9* *example: CP_EX*

bmUnitName*	**string** *minLength: 1* *maxLength: 9* *title: BM Unit Name* *example: EF-FLEO45*
logTime*	**string** *pattern: ^([1-9]|[012]\d|3[01])[-](JAN|FEB|MAR|APR|MAY|JUN|JUL|AUG|SEP|OCT|NOV|DEC)[-]((2[0-9]{3}))[ ]((\[0-1][0-9])|(2[0-3])):([0-5][0-9])$* *title: Redeclaration Reference Number* *example: 18-OCT-2018 06:00*
refNumber*	**string** *pattern: ^\d{1,10}$* *title: Redeclaration Reference Number* *example: 10584466*
BMURedeclarationElements*	**BMURedeclarationElements [. . .]**

}

BMURedeclarationElements {

minItems: 1
maxItems: 1

BMURedeclarationElements {

mel	**Maximum Export Limit { . . . }**
mil	**Maximum Import Limit { . . . }**
rure	**Run Up Rate Export { . . . }**
rdre	**Run Down Rate Export { . . . }**
ndz	**Notice to Deviate From Zero { . . . }**
sel	**Stable Export Limit { . . . }**
mzt	**Minimum Zero Time { . . . }**
mnzt	**Minimum Non Zero Time { . . . }**
ruri	**Run Up Rate Import { . . . }**
Rdri	**Run Down Rate Import { . . . }**
nto	**Notice to Deliver Offers { . . . }**
ntb	**Notice to Deliver Bids { . . . }**
sil	**Stable Import Limit { . . . }**
mdvp	**Maximum Delivery Volume { . . . }**

}]

Maximum Export Limit {

timeFrom*	**string** *pattern: ^([1-9]|[012]\d|3[01])[-](JAN|FEB|MAR|APR|MAY|JUN|JUL|AUG|SEP|OCT|NOV|DEC)[-]((2[0-9]{3}))[ ](([0-1][0-9])|(2[0-3])):([0-5][0-9])$* *example: 18-OCT-2018 06:00*
maximumExportLevelFrom*	**string** *pattern: ^([+]?\d{1,4})$* *example: 0*
timeTo*	**string** *pattern: ^([1-9]|[012]\d|3[01])[-](JAN|FEB|MAR|APR|MAY|JUN|JUL|AUG|SEP|OCT|NOV|DEC)[-]((2[0-9]{3}))[ ](([0-1][0-9])|(2[0-3])):([0-5][0-9])$* *example: 18-OCT-2018 06:30*
maximumExportLevelTo*	**string** *pattern: ^([+]?\d{1,4})$* *example: 9999*

}

Maximum Import Limit {

timeFrom*	**string** *pattern: ^([1-9]|[012]\d|3[01])[-](JAN|FEB|MAR|APR|MAY|JUN|JUL|AUG|SEP|OCT|NOV|DEC)[-]((2[0-9]{3}))[ ](([0-1][0-9])|(2[0-3])):([0-5][0-9])$* *example: 18-OCT-2018 06:00*
maximumImportLevelFrom*	**string** *pattern: ^([-]\d{1,4}|[0])$* *example: -9999*
timeTo*	**string** *pattern: ^([1-9]|[012]\d|3[01])[-](JAN|FEB|MAR|APR|MAY|JUN|JUL|AUG|SEP|OCT|NOV|DEC)[-]((2[0-9]{3}))[ ](([0-1][0-9])|(2[0-3])):([0-5][0-9])$* *example: 18-OCT-2018 06:30*
maximumImportLevelTo*	**string** *pattern: ^([-]\d{1,4}|[0])$* *example: 0*

}

Run Up Rate Export {

rate1*	**string** *pattern: ^([+]?\d{1,3}[.]\d{1})$*

	example: 15.0
elBow2	**string** *pattern: ^([+]?\d{1,4})$* *example: 140*
rate2	**string** *pattern: ^([+]?\d{1,3}[.]\d{1})$* *example: 3.4*
elBow3	**string** *pattern: ^([+]?\d{1,4})$* *example: 145*
rate3	**string** *pattern: ^([+]?\d{1,3}[.]\d{1})$* *example: 12.7*

}

Run Up Rate Import {

rate1*	**string** *pattern: ^([+]?\d{1,3}[.]\d{1})$* *example: 010.0*
elBow2	**string** *pattern: ^([-]\d{1,4})$* *example: -0340*
rate2	**string** *pattern: ^([+]?\d{1,3}[.]\d{1})$* *example: 015.0*
elBow3	**string** *pattern: ^([-]\d{1,4})$* *example: -140*
rate3	**string** *pattern: ^([+]?\d{1,3}[.]\d{1})$* *example: 15.0*

}

Run Down Rate Export {

rate1*	**string** *pattern: ^([+]?\d{1,3}[.]\d{1})$* *example: 015.0*
elBow2	**string** *pattern: ^([+]?\d{1,4})$* *example: +0140*
rate2	**string** *pattern: ^([+]?\d{1,3}[.]\d{1})$* *example: 015.0*
elBow3	**string** *pattern: ^([+]?\d{1,4})$*

	example: +0145
rate3	**string** *pattern: ^([+]?\d{1,3}[.]\d{1})$* *example: 015.0*

}

Run Down Rate Import {

rate1*	**string** *pattern: ^([+]?\d{1,3}[.]\d{1})$* *example: 015.0*
elBow2	**string** *pattern: ^([-]\d{1,4})$* *example: -0140*
rate2	**string** *pattern: ^([+]?\d{1,3}[.]\d{1})$* *example: 015.0*
elBow3	**string** *pattern: ^([-]\d{1,4})$* *example: -0140*
rate3	**string** *pattern: ^([+]?\d{1,3}[.]\d{1})$* *example: 015.0*

}

Notice to Deviate From Zero {

timeValue*	**string** *pattern: ^\d{1,3}$* *example: 30*

}

Notice to Deliver Offers {

timeValue*	**string** *pattern: ^\d{1,2}$* *maximum: 59* *example: 59*

}

Notice to Deliver Bids {

timeValue*	**string** *pattern: ^\d{1,2}$* *maximum: 59* *example: 59*

}

Minimum Zero Time {

timeValue*	**string** *pattern: ^\d{1,3}$* *example: 999*

}

Minimum Non Zero Time {

timeValue*	**string** *pattern: ^\d{1,3}$* *example: 999*

}

Stable Export Limit {

MWlevel*	**string** *pattern: ^\d{1,4}$* *example: 9999*

}

Stable Import Limit {

MWlevel*	**string** *pattern: ^([-]\d{1,4})$* *example: -9999*

}

Maximum Delivery Volume {

MDV*	**string** *title: Max Delivery Volume (MW hours)* *pattern: ^([-+]?\d{1,5})$* *example: 99999*
MDP*	**string** *title: Max. Delivery Period (minutes)* *pattern: ^\d{1,3}$* *example: 239*

}

Error payload {

message*	**string** *maxLength: 200* *example: Error message*
version*	**string**

	example: 1.0
code*	**string**
	example: 400
detail	**string**
	example: Error information

}

Successful Redeclaration Response {

version*	**string**
	example: 1.0
message*	**string**
	maxLength: 200
	example: Successful Request

}

NGESO_Instruction

Overview

This API will be used by Market Participant to send Instruction response to NGESO. Responses can include;

- UserACK
- Acceptance
- Rejection
- Error

The type of response must be specified in the **instructionResp** field of the payload.

Acceptance is done in two steps using the same service. First the Market Participant will send status “UserACK” that is translated in the BM to “IU” as specified in [2], they will then use the same interface to send the “Acceptance” that is translated into “IA”. In the case where National Grid receives an “Acceptance” but not “UserACK”, the process will still be completed. An error can be received at any stage.

An UserACK payload to a received BOA instruction will be structured as follow;

```
{
    "controlPoint": "CP_EX",
    "refNumber": "0010584466",
    "instructionResp": "UserACK",
    "instructionType": "BOA",
    "bmUnitName": "XY-MNLX01",
    "logTime": "18-OCT-2018 00:00"
}
```

An error payload to a received BOA instruction will be structured as follow. **Please note** that in this particular case the **instructionResp** field must be equal to "Error" and the **detail** field must contain the error code as mentioned in reference [2] and [3].

```
{
    "controlPoint": "CP_EX",
    "refNumber": "0010584466",
    "instructionResp": "Error",
    "instructionType": "BOA",
    "bmUnitName": " XY-MNLX01",
    "logTime": "18-OCT-2018 00:00",
    "detail": "I001"
}
```

Market Participants can receive instruction for the following business entities.

- Pumped Storage
- Voltage / MVAR
- Reason Code
- Bid/Offer
- Status Change

The type of instruction must be specified in the **instructionType** field of the payload.

Request header and responses will be provided during the onboarding process.

POST **/instructionresp** MP Operational Response of an Instruction [Market Participant -> NGESO]

Models

Successful Instruction Response {

message*	**string**
	maxLength: 200
	example: Successful request
version*	**string**
	example: 1.0

}

Error payload	{
message*	**string**
	maxLength: 200
	example: Error message
version*	**string**
	example: 1.0
code*	**string**
	example: 400
detail	**string**
	example: Error information

}

Instruction response	{
controlPoint*	**string**
	minLength: 1
	maxLength: 9
	example: CP_EX
refNumber*	**string**
	pattern: ^\d{1,10}$
	example: 0010584466
instructionResp*	**string**
	example: Error
	Enum:
	[UserAck, Accepted, Rejected, Error]
instructionType*	**string**
	Enum:
	[BOAI, VoltageMVAR, ReasonCode, StatusChange, PumpedStorage]
bmUnitName*	**string**
	maxLength: 9
	title: BM Unit Name
	example: AG-FFLX01
logTime*	**string**
	pattern: ^([1-9]|[012]\d|3[01])[-](JAN|FEB|MAR|APR|MAY|JUN|JUL|AUG|SEP|OCT|NOV|DEC)[-]((2[0-9]{3}))[ ](([0-1][0-9])|(2[0-3])):([0-5][0-9])$
	example: 18-OCT-2018 00:03

detail	**string** *maxLength: 140* *example: I001*

}

NGESO_Health

Overview

In addition to the operational interfaces (EDT and EDL), the Health API will have the functionality of testing the connectivity between Market Participants and National Grid. In each side of the communication between National Grid and the Market Participants, there will be a service (heartbeat service) that will provide a response to indicate whether a connectivity exists.

The functionalities implemented with the NGESO_Health API are the following.

1. Checking whether the Wider Access APIs hosted on NGESO are reachable
2. Checking whether each individual API (submission & redeclaration) is up and running
3. The NGESO credentials are valid

GET **/health** Checks the connectivity health and credentials

Request header and responses will be provided during the onboarding process.

Models

Health of APIs {

version*	**string** *example: 1.0*
APIs*	**[. . .]**

}

List of APIs {

apiname*	**string** *example: WASubmissionEDT* Enum: [WASubmissionEDT, WARedeclarationEDL, WAInstructionEDL]
status*	**string** *example: Up* Enum: [Up, Down]
version*	**string** *example: 1.0*

}

Error payload	{
message*	**string** *example: Error message*
version*	**string** *example: 1.0*
code*	**string** *example: 400*
detail	**string** *example: Error information*

}

NGESO_Normalization

Normalization process

In order to standardise the Normalization process of a payload to be signed, the NGESO_Normalization API must be used.

Steps below describe the process to follow in order to obtain a correct signature to be included as part of the payload header;

1. Calling the NGESO_Normalization API;
2. Signing the resulted payload from point (1) (ensure that you are not adding any \r character in the normalized payload before signing it)

Please note that, the payload to be sent in point (1) must be the one to be signed. In the specification below, a generic example is included but it may be one of the following;

- EDT Submission payload;
- EDL Redeclaration payload;
- EDL Instruction Response payload.

POST **/normalization** Normalize the payload

Request header and responses will be provided during the onboarding process.

Models

Example of a payload sent {

field1	**string** *example: field 1*
field3	**string** *example: field 3*
field2	**string** *example: field 2*

}

Example of a payload normalized {

field1	**string** *example: field 1*
field2	**string** *example: field 2*
field3	**string** *example: field 3*

}

Error payload	{
message*	**string** *example: Error message*
version*	**string** *example: 1.0*
code*	**string** *example: 400*
detail	**string** *example: Error information*

}

Participant_Submission Responses (Acknowledgment/Acceptance/Rejection)

Overview

The Submission Response (Acknowledgment) API allows NGESO to send Acknowledgment of a received submission request.

The Submission Response (Acceptance/Rejection) API allows NGESO to send Acceptance and/or Rejection of a received submission request.

Notification Time

The System Time of the Host shall be in GMT and shall be kept referenced to a recognised global time base. It is this time, which shall determine gate closure for submissions. Each invocation successfully transferred to the Host will be deemed to be a submission. The notification time of this file, and hence all data contained within it, shall be deemed to be the point in time that the submission was made.

Submission Acknowledgment

The acknowledgement message will contain the notification time of the submission processed by BM. The notification time will use the standard Time format defined in the Convention paragraph.

Submission Acceptance

Acceptance are produced once a submission request has been validated in its entirety. The acceptance response will contain a list of all BMUs for which all submitted data has passed formatting, consistency and validation tests.

Submission Rejection

Submission rejection are also produced once a submission has been validated in its entirety. Each record contained is checked for formatting, validity and consistency. Should formatting prove incorrect the request will not proceed to validation and will be rejected at that stage. Thus, a record that has invalid data and is also incorrectly formatted for the type of data will only have a message stating that it was rejected owing to a formatting error. The validity of the record will not be considered. Once a record has completed and passed formatting checks, it will be checked against each applicable validation and consistency rule. Any and all these failures will be reported individually for each submitted record. Hence a single row that does not comply with multiple validation or consistency rules, will give rise to multiple error messages within a reject request.

Compression process

When a wrapped payload has been sent for submission, the Acceptance/Rejection response will also be in the same format, as for the example below.

```
{
    “mpid”: “Market Participant ID”,
    “number”: “Sequence Number”,
    “data”: “Compressed Payload (ACK, ACC/REJ)”
}
```

The compressed payload included in the data field must be uncompressed, by using gZip.

Please note that, in order to verify the signature, the compressed payload in the **data** field must be used.

Acknowledgment/Acceptance/Rejection examples

Example 1 Acknowledgment. The **notificationTime** is the same sent by BM

```
{
    "sequence": "1234",
    "notificationTime": "2018-10-11 01:03"
}
```

Example 2 Acceptance of multiple BMUs

```
{
    "sequence": "1234",
    "tradingAgent": "TR_AGT",
    "acceptance": [
        {
            "bmUnitName": "BM_UNIT_1"
        },
        {
            "bmUnitName": "BM_UNIT_2"
        }
    ]
}
```

Example 3 Rejection of a submission

```
{
    "sequence": "1234",
    "tradingAgent": "TR_AGT",
    "rejection": [
        {
            "code": "V_RURE_2",
            "message": "An invalid combination of NULL rates and breakpoints was
            encountered",
            "record": "RURE, TR_AGT, BM_UNIT_3, 2001-11-03 05:00, , , 12,"
        }
    ]
}
```

POST **/submissionack** BM Acknowledgment Response for a submission [NGESO -> Market Participant]

POST **/submissionresp** BM Acceptance/Rejection Response of a submission [NGESO -> Market Participant]

Request header and responses will be provided during the onboarding process.

Models

Market Participant Acknowledgement {

sequence*	**string** *title: Sequence* *pattern: ^\d{1,4}$* *example: 1234*
notificationTime*	**string** *pattern: ^(2([0-9]{3}))[-](0[1-9]|10|11|12)[-](0[1-9]|[12]\d|3[01])[ ](([0-1][0-9])|(2[0-3]))[:]([0-5][0-9])$* *example: 2018-10-11 01:03*

}

BM Acceptance Response {

bmUnitName*	**string** *title: Unit Name* *example: BM_UNIT_1* *minLength: 1* *maxLength: 9*

{

BM Rejection Response {

code*	**string** *example: V_RURE_2*
message*	**string** *maxLength: 200* *example: An invalid combination of NULL rates and breakpoints was encountered*
record*	**string** *example: RURE, TR_AGT, BMUNIT01, 2001-11-03 05:00, , , 12,*

}

Successful Submission Response {

message*	**string** *maxLength: 200* *example: Successful request*
version*	**string** *example: 1.0*

}

Error payload {

message*	**string** *maxLength: 200* *example: Error message*
version*	**string** *example: 1.0*
code*	**string** *example: 400*
detail	**string** *example: Error detailed information*

}

Acceptance Rejection Response {

sequence*	**string** *title: Sequence* *pattern: ^\d{1,4}$* *example: 1234*
tradingAgent*	**string** *minLength: 1* *maxLength: 9* *example: TR_AGT*
acceptance	**[BM Acceptance Response { . . . }]**
rejection	**[BM Rejection Response { . . . }]**

}

Participant_Redeclaration

Overview

This API is used by NGESO to send responses to a Market Participant redeclaration.

Type of response will be included in the **redeclarationResp** field of the payload being sent; possible responses are;

- Wait
- Accepted
- Rejected
- Expired

In case a Rejection payload is sent, the error code as specified in [2] will be provided as part of the **detail** field.

In case no responses are received by BM, a payload with **redeclarationResp** "Expired" will be sent and a new redeclaration should be sent.

Redeclaration, Acceptance and Rejection

Data validation is concerned with checking that data is in the correct format and within the correct limits, e.g. is it an integer, is it between given limits etc. Data consistency concerns itself with checking if a particular data record is consistent with other data records, and defaulting rules are applied in cases of missing data which should have been submitted. Failure to comply with the validation or consistency rules will result in rejection of the redeclaration for the BM Unit affected.

POST **/redeclarationresp** BM Operational Response of a redeclaration [NGESO -> Market Participant]

Request header and responses will be provided during the onboarding process.

Models

Redeclaration Acceptance/Rejection {

controlPoint*	**string** *minLength: 1* *maxLength: 9* *example: CP_EX*
refNumber*	**string** *pattern: ^\d{1,10}$* *example: 0010584466*
redeclarationResp*	**string** *example: Rejected* Enum: Array [4]
logTime*	**string**($string) *pattern: ^([1-9]\|[012]\d\|3[01])[-]* *(JAN\|FEB\|MAR\|APR\|MAY\|JUN\|JUL\|AUG\|SEP\|OCT\|NOV\|DEC)[-]((2[0-9]* *{3}))[ ](([0-1][0-9])\|(2[0-3])):([0-5][0-9])$* *example: 18-OCT-2018 06:00*

bmUnitName*	**string** *minLength: 1* *maxLength: 9* *title: BM Unit Name* *example: AG-FFLX01*
detail	**string** *maxLength: 140* *example: R999*

}

Successful Redeclaration Response {

version*	**string** *example: 1.0*
message*	**string** *maxLength: 200* *example: Successful Request*

}

Error payload {

message*	**string** *maxLength: 200* *example: Error message*
version*	**string** *example: 1.0*
code*	**string** *example: 400*
detail	**string** *example: Error information*

}

Participant_Instruction

Overview

The Instruction API will be used by NGESO to send instructions to the Market Participants. As explained in more detail in [2], the following instructions can be sent;

- Pumped Storage
- Voltage / MVAR
- Reason Code
- Bid/Offer
- Status Change

Please note that;

only one of the instruction types above can be sent at time.

The expected process after an instruction is;

- Market Participant sending Wait
- Market Participant sending User Acknowledgement
- Market Participant sending Acceptance or Rejection
- Error can be sent at any stage of the process

Please note that;

- the full instruction process must be completed within 2 minutes since the logTime. In case of time expiration, an expiration response will be sent with **"instructionType":"Expired"**

POST **/instruction** Add a new Instruction [NGESO -> Market Participant]

Request header and responses will be provided during the onboarding process.

Models

Instruction Request data {

controlPoint*	**string** *minLength: 1* *maxLength: 9* *example: CP_EX*
bmUnitName*	**string** *minLength: 1* *maxLength: 9* *title: BM Unit Name* *example: CLCPU-01*
refNumber*	**string** *pattern: ^\d{1,10}$* *title: Instruction Reference Number* *example: 0011513095*
logTime*	**string** *pattern: ^([1-9]|[012]\d|3[01])[-]*

	(JAN|FEB|MAR|APR|MAY|JUN|JUL|AUG|SEP|OCT|NOV|DEC)[-]((2[0-9]{3}))[ ](([0-1][0-9])|(2[0-3])):([0-5][0-9])$ *example: 20-FEB-2020 16:00*
instructionType*	**string** Enum: Array [6]
statusChange	**Message Data Part for Status Change Instruction Messages { . . . }**
boaMsg	**Message Data Part for BOA and Deemed Closed Instruction Messages { . . . }**
reasonCodeInstruction	**Message Data Part for Change of Reason Code Instruction Messages { . . . }**
mvarInstruction	**Message Data Part for Voltage/MVAR Instruction Messages { . . . }**
pumpedInstruction	**Message Data Part for Pumped Storage Unit Instruction Messages { . . . }**

}

Error payload {

message*	**string** *maxLength: 200* *example: Error message*
version*	**string** *example: 1.0*
code*	**string** *example: 400*
detail	**string** *example: Error information*

}

BM Element Instruction Response {

version*	**string** *example: 1.0*
timestamp*	**string** *example: 20-FEB-2020 16:00* *pattern: ^([1-9]|[012]\d|3[01])[-](JAN|FEB|MAR|APR|MAY|JUN|JUL|AUG|SEP|OCT|NOV|DEC)[-]((2[0-9]{3}))[ ](([0-1][0-9])|(2[0-3])):([0-5][0-9])$*
bmUnitName*	**string** *maxLength: 9* *title: BM Unit Name* *example: CLCPU-01*
refNumber*	**string** *pattern: ^\d{1,10}$* *title: Instruction Reference Number* *example: 0011513095*

instructionResp*	**string**
	example: Wait
	Enum:
	Array [1]

}

Message Data Part for Status Change Instruction Messages {

startInstructionCode*	**string**
	Enum:
	Array [3]
startTime*	**string**
	example: 20-FEB-2020 16:00
	pattern: ^([1-9]|[012]\d|3[01])[-]
	(JAN|FEB|MAR|APR|MAY|JUN|JUL|AUG|SEP|OCT|NOV|DEC)[-
	]((2[0-9]
	{3}))[ ](([0-1][0-9])|(2[0-3])):([0-5][0-9])$
ReasonCode*	**string**
	pattern: ^\w{1,3}$
	example: MFB
targetInstructionCode*	**string**
	Enum:
	Array [4]
TargetTime*	**string**
	example: 20-FEB-2020 16:00
	pattern: ^([1-9]|[012]\d|3[01])[-]
	(JAN|FEB|MAR|APR|MAY|JUN|JUL|AUG|SEP|OCT|NOV|DEC)[-
	]((2[0-9]
	{3}))[ ](([0-1][0-9])|(2[0-3])):([0-5][0-9])$

}

Message Data Part for BOA and Deemed Closed Instruction Messages {

type*	**string**
	example: DEEM
	Enum:
	Array [2]
boaNumber*	**string**
	pattern: ^\d{1,10}$
	example: 70382
numberDataPoints*	**string**
	pattern: ^(0[2-5]{1})$
	example: 04
mw1*	**string**
	pattern: ^[+-]?\d{1,4}$
	example: +2000
t1*	**string**
	example: 20-FEB-2020 16:00

	pattern: ^([1-9]|[012]\d|3[01])[-]
	(JAN|FEB|MAR|APR|MAY|JUN|JUL|AUG|SEP|OCT|NOV|DEC)[-]((2[0-9]
	{3}))[ ](([0-1][0-9])|(2[0-3])):([0-5][0-9])$
mw2*	**string**
	pattern: ^[+-]?\d{4}$
	example: +2000
t2*	**string**
	example: 20-FEB-2020 16:00
	pattern: ^([1-9]|[12]\d|3[01])[-]
	(JAN|FEB|MAR|APR|MAY|JUN|JUL|AUG|SEP|OCT|NOV|DEC)[-]((([0-9]
	{2}))[ ](([0-1][0-9])|(2[0-3])):([0-5][0-9])$
mw3	**string**
	pattern: ^[+-]?\d{4}$
	example: +2000
t3	**string**
	example: 20-FEB-2020 16:00
	pattern: ^([1-9]|[012]\d|3[01])[-]
	(JAN|FEB|MAR|APR|MAY|JUN|JUL|AUG|SEP|OCT|NOV|DEC)[-]((2[0-9]
	{3}))[ ](([0-1][0-9])|(2[0-3])):([0-5][0-9])$
mw4	**string**
	pattern: ^[+-]?\d{4}$
	example: +2000
t4	**string**
	example: 20-FEB-2020 16:00
	pattern: ^([1-9]|[012]\d|3[01])[-]
	(JAN|FEB|MAR|APR|MAY|JUN|JUL|AUG|SEP|OCT|NOV|DEC)[-]((2[0-9]
	{3}))[ ](([0-1][0-9])|(2[0-3])):([0-5][0-9])$
mw5	**string**
	pattern: ^[+-]?\d{4}$
	example: +2000
t5	**string**
	example: 20-FEB-2020 16:00
	pattern: ^([1-9]|[012]\d|3[01])[-]
	(JAN|FEB|MAR|APR|MAY|JUN|JUL|AUG|SEP|OCT|NOV|DEC)[-]((2[0-9]
	{3}))[ ](([0-1][0-9])|(2[0-3])):([0-5][0-9])$

}

Message Data Part for Change of Reason Code Instruction Messages {

type*	**string**
	example: REAS
	Enum:
	Array [1]
ReasonCode*	**string**

	pattern: ^\w{1,3}$
startTime*	**string** *pattern: ^([1-9]|[012]\d|3[01])[-](JAN|FEB|MAR|APR|MAY|JUN|JUL|AUG|SEP|OCT|NOV|DEC)[-]((2[0-9]{3}))[ ](([0-1][0-9])|(2[0-3])):([0-5][0-9])$* *example: 20-FEB-2020 16:00*

}

Message Data Part for Voltage/MVAR Instruction Messages {

type*	**string** Enum: Array [2]
value*	**string** *pattern: ^([+-]?\d{1,3})$* *example: +123*
targetTime*	**string** *example: 20-FEB-2020 16:00* *pattern: ^([1-9]|[012]\d|3[01])[-](JAN|FEB|MAR|APR|MAY|JUN|JUL|AUG|SEP|OCT|NOV|DEC)[-]((2[0-9]{3}))[ ](([0-1][0-9])|(2[0-3])):([0-5][0-9])$*

}

Message Data Part for Pumped Storage Unit Instruction Messages {

ReasonCode*	**string** *pattern: ^\w{1,4}$* Enum: Array [7]
startTime*	**string** *example: 20-FEB-2020 16:00* *pattern: ^([1-9]|[012]\d|3[01])[-](JAN|FEB|MAR|APR|MAY|JUN|JUL|AUG|SEP|OCT|NOV|DEC)[-]((2[0-9]{3}))[ ](([0-1][0-9])|(2[0-3])):([0-5][0-9])$*
target*	**string** *example: MW* *pattern: ^(MW|SH|SG|SP|[0-9]{1,2}.[0-9]{1,2})$*
targetTime*	**string** *example: 20-FEB-2020 16:00* *pattern: ^([1-9]|[012]\d|3[01])[-](JAN|FEB|MAR|APR|MAY|JUN|JUL|AUG|SEP|OCT|NOV|DEC)[-]((2[0-9]{3}))[ ](([0-1][0-9])|(2[0-3])):([0-5][0-9])$*

}

Participant_Health

Overview

In addition to the operational interfaces (EDT and EDL), the Health API will have the functionality of testing the connectivity between Market Participants and National Grid. In each side of the communication between National Grid and the Market Participants, there will be a service (heartbeat service) that will provide a response to indicate whether a connectivity exists.

The functionalities implemented with the NGESO_Health API are the following;

1. Checking the connectivity between NGESO and a Market Participant
2. Validate that the NGESO credentials are accepted by the Wider Access Participant

GET **/health** Checks the connectivity health and credentials

Request header and responses will be provided during the onboarding process.

Models

Successful Health Response {

message*	**string** *example: Successful request*
version*	**string** *example: 1.0*

}

Error payload {

message*	**string** *example: Error message*
version*	**string** *example: 1.0*
code*	**string** *example: 400*
detail	**string** *example: Error information*

}

API rules

Security

All the requests must be secured over HTTPS by using SSL/TLS 1.2 or above.

All the APIs are configured as protected OAuth2 resources and therefore will reject anonymous requests. In order to be able to consume an API exposed by NGESO and make a submission correctly, a **token** (also called JSON Web Token - JWT) must be provided as part of the payload header.

A **token** can be obtained using the 'Client ID' and 'Client Secret' provided to the Market Participant as part of the onboarding process and making a request to the Identity Provider. A **token** does expire and must be included in each request.

Additionally, when making a request for a token, a **Scope** and an **Audience** must be specified, based on the API being consumed.

The **Audience** will be always the same, i.e. **NGESO**. Details of the **Scope** used to protect the different resources will be provided during the registration process.

Subscription Key

When sending a submission or instruction payload to NGESO, an **OSCGAppKeyHeader** must be included which will be used to provide usage analytics of the APIs for the Market Participants.

OSCGAppKeyHeader is not needed for requests done by NGESO to APIs exposed by Market Participant.

Signature

To prevent disputes in the authorship of a payload, the requesting party (NGESO or Market Participant) will include a **signature** as part of the payload header.

More details on how to generate signatures and the verification of these will be provided during the registration process.

Normalization

As specified in the security section, each payload being sent must be signed. A **signature** must be applied to the normalized payload being sent.

In order to standardise the process of normalization of a payload, the Normalization endpoint must be called. For all the details about the process, please refer to the **NGESO_Normalization** API.

Please note that;

- **Normalised payload** is only used for signature purposes, but a normal payload must be sent during a submission.

Reply To

In order to identify the Market Participant for Acknowledgment, Acceptance and Rejection responses, the **Fully Qualified Domain Name (FQDN)** of the API exposed by Market Participant must be shared with NGESO via email.

Sequence number (Submission API)

For the Submission API, each submission payload will include a sequence number **sequence** as a mandatory field. As specified in [4] in more detail, the sequence number should be incremented by one after each submission. Should a submission be out of sequence, it will be rejected in its entirety. The last sequence number sent will be included in the rejection payload as part of the Market Participant's API for Acceptance/Rejection.

Please note that, sequence number should not be increased by one after receiving a technical error (e.g. 400, 401, 500) in response to a submission.

For example;

- a submission is made with a sequence number 0001;
- a technical error 400 is received;
- next submission should have sequence number 0001.

Please note that, as the technical response is synchronous with the request which has been made, Market Participant cannot submit a new request until the technical response is received.

If a submission response (ACK, ACC and/or REJ) is not received within **5 minutes**, a new submission must be sent to NGESO. The sequence number of the new submission request will depend on the success or not of the previous request.

Reference number (Instruction and Redeclaration API)

For the Instruction API, each submission payload will include a reference number **refNumber** as a mandatory field. The reference number must be the same of the one associated to the instruction received.

For the Redeclaration API, each submission payload will include a reference number **refNumber** as a mandatory field.

Reference number should be incremented by one after each redeclaration.

Please note that;

- reference number should not be increased by one after receiving a technical error (e.g. 400, 401, 500) in response to a redeclaration;
- as the technical response is synchronous with the request which has been made, Market Participants cannot submit a new request until the technical response is received.

If a redeclaration response is not received within **2 minutes**, a new redeclaration must be sent to NGESO. The reference number of the new redeclaration request will depend on the success or not of the previous request.

List of errors

When errors are encountered in an API request, a technical error payload will be sent. The table below includes the list of error codes, descriptions and details which can be received.

Error code	Message	Details	HTTPS Error Code
WABE0001	Verify Token Error (parse token/token invalid)	Token is invalid	401
WABE0002	Verify Signature Error	The signature does not match	401
WABE0003	Verify Signature Error (Wrong key)		401
WATE0001	SecurityOperations Initialization	Identity Provider ID was not set up	500
WATE0002	SecurityOperations Initialization	Service to retrieve Public Key is not set up	500
WATE0003	SecurityOperations Initialization	Exception while initializating Token Parser	500
WATE0004	SecurityOperations Initialization	Exception while initializating Non Repudiation	500
WATE0005	Controller Initialization	Identity Certificate (jwt) is not setup	500
WATE0006	Resource not Found		500
WATE0007	Unexpected Error		500
WATE0008	Mandatory Arguments not informed		500
WATE0009	Create Signature Error		500
WABE1001	Market Participant Id mismatch	Market Participant Id in the payload does not match with the one received after signature verification	401
WABE1002	Error verifying the signature	Signature Verification process ended in error	500
WABE1004	Request in error		400

WABE2001	Internal server error	A technical error when processing the request	500
WABE2001	Internal server error	Internal server error	
WABE3001	Internal server error	Internal server error	500
WABE3004	Wrong number of elements	Only one element can be re-submitted	400
WABE3005	Duplicated Message	There is a running instance for same reference number	500
WABE3007	Input data invalid	Payload sent is incorrect	400
WABE3008	Wrapper data invalid		400
WABE2004	Request Duplicated	Another Submission with the same sequence number is already in process	400
WABE2002	Request not valid	Invalid Payload received	400
WABE2006	Internal server error	Request Timeout	500
WABE5002	Error in retrieving token	WATokenManagement service ended in error	401
WABE5003	Error calling Market Participant	Error received while calling Market Participant	500
WABE5004	Request in error		500
WABE5005	Wrong Scope	Wrong Scope passed to the service	401
WABE6001	Instruction Response not valid	Error verifying the signature and token for incoming Instruction Response	500
WABE6002	Instruction Response not sent	Technical error received while sending Instruction Response	500
WABE6009	Error in verifying Token	Instruction could not be sent to BM	500
WABE6008	Request not valid	Invalid Payload received	500
WABE6013	Error received from BM	Error received from BM	500
WABE9001	Invalid request		400
WATE3001	Internal server error	Internal server error	500

WATE9004	Requested Control Point not found	The Control Point was not among the ones configured in the DB	500
WATE9005	Cannot write	The channel is offline	500
WATE9006	Cannot open the Selector		500
WATE9007	The Channel is Closed		500
WATE9008	The Channel is Not Connected		500
WATE9011	The Socket is not ready	A message cannot be sent before the Channel with BM is ready	500
WABE9001	Invalid request		400
-	UnAuthorized to access the resource.		401
-	Failed to authenticate application		401
-	content size [-] exceed MaxMessageSize [1024000]		413
-	Error : Application has reached its limit for this minute.		429
-	API Rate Limit has been reached		429

OAuth2 tokens

In order to consume the Wider Access APIs, the consumers will have to be authenticated with a JSON Web Token (JWT). The process to issue such a **token** will follow the *OAuth2* protocol, and specifically the *client_credentials* flow.

NGESO will implement an Identity Provider where all the Market Participant identities, including those of NGESO, will be stored. The identities are created as part of the enrolment process and will consist of:

A set of user and password to access the Development Portal - not required to consume APIs

A 'Client ID' and 'Client Secret', needed to obtain a token to consume an API

More information on the generation and validation of a **token** will be provided during the registration process.

OAuth Roles

OAuth defines three roles in the process of issuing and validating token:

- **Authentication Server**. In charge of issuing new tokens. This role will be fulfilled by NGESO
- **Resource Provider**. The party that provides the APIs to be consumed. It is responsible for validating that the token associated to the API request is a valid token.
    - NGESO will play the role of Resource Provider for the EDT Submission, EDL Redeclaration, EDL Instruction User ACK, EDL Instruction ACC/REJ
    - The Market Participants will play the role of Resource Provider for the EDT User ACK, EDT ACC/REJ, EDL redeclaration ACC/REJ and EDL Instruction
- **Client**. Party that wants to consume a protected Resource. It can be represented by both NGESO and Market Participants

Conventions

- **timeFrom** = DD-MON-YYYY hh:mm
- **timeTo** = DD-MON-YYYY hh:mm
- **levelFrom** = MW
- **levelTo** = MW
- **offerPrice** = £ / MWh
- **bidPrice** = £ / MWh
- **rate** = MW / minute

Date/Time Format:

- **YYYY** = year (numeric)
- **MON** = month from the set {JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC}
- **DD** = day (numeric)

a single space separator between date and time

- **hh** = hours
- **mm** = minutes

Reference

[1] National Grid ESO

[2] EDL Message Interface Specification

[3] Data Validation and Consistency Checking

[4] EDT Message Interface Specification