ESO

Restoration Services from Wind Farms
Calculation of Availability and Resilience of the Service

Table of Contents

A. Context

Wind Farms, like all other restoration service providers, will be expected to be available for the provision of the service at least 80% of the time and be able to deliver the contracted figures for a minimum of 10h.

Unlike other technologies, the ESO accepts the added challenge for this type of generation particularly around these requirements and is seeking to offer a way forward that will be able to be replicated across all potential Wind Farms seeking to provide a Restoration Service.

B. Operational Metering

1. Technical Parameters

All providers will be expected to provide the ESO with a "live feed" of the parameters listed below. These parameters are expected to be firmed up in the relevant contract/agreement.

a. Frequency (Hz).
b. Voltage (kV).
c. Availability of Contracted Units (for each contracted Unit, Available/Unavailable).
d. Power Output – MW.
e. Power Output – MVAr.

Specifically for Wind Providers and on top of the parameters above we are expecting to have the following information available in ESO Control Room:

f. Wind speed forecasts and observations (ms^{-1}).
g. Wind direction forecasts and observations (degrees).

2. Wind Speed/Direction

2.1 Location of the Measuring Equipment

As a minimum the ESO requires at least two meteorological masts or nacelle anemometers at the site (main and backup) and they should be located as near to the hub height of the wind turbine(s) as possible.

2.2 Resilience of the Measuring Equipment

All relevant meteorological mast(s) shall have a resilience aligned with the expected resilience for the restoration service. For the avoidance of doubt the ESO is expecting that if under a National Power Outage and in alignment with the declared availability declared for the Service (Example: if a Primary Service Provider ≥72h):

- The equipment will continue working under a National Power Outage.
- Data will continue to be made available up to the interface with the ESO system.

2.3 Minimum Delivery Requirements

2.3.1 Observations

This includes wind speed and direction data from all meteorological masts / nacelle anemometers, as well as the number of available turbines of each turbine type.

- Shall be delivered as close to real time as possible and with a maximum delay of 30 minutes.
- Represent the average wind conditions in continuously reporting blocks of 30 minutes or less.

2.3.2 Forecasts

- Shall be valid up to at least 10h after the delivery time.
- Shall be updated at least every hour.
- Shall contain the average wind conditions in continuously reporting blocks of 30 minutes.

2.4 Data Quality

Data quality will be assessed after passing the forecast wind data through the algorithm as described under section C. Methodology below to obtain a MW power forecast. The forecast accuracy will then be determined by comparing the MW power forecast with the metered power output of the wind farm.

3. Metering Interface with the ESO

3.1 Data Types

The ESO is expecting a live feed of the following data/parameters:

Data type	Requirements
Active Power (MW)	Accuracy: -/+ 1.0% of reading Frequency: 1 per Second
Reactive Power (MVAr)	Accuracy: -/+ 1.0% of reading Frequency: 1 per Second
Frequency (Hz).	Accuracy: -/+ 1.0% of reading Frequency: 1 per Second
User System Entry Point Voltage (kV)	Accuracy: -/+ 1.0% of reading Frequency: 1 per second
Tap Position (TPI)	Accuracy: Exact Reading Frequency: On Change
State of contracted unit(s)	Accuracy: Exact Reading Frequency: On Change

Availability of the Contracted Service	Accuracy: Exact Reading Frequency: On Change
Controlling Breaker (OPEN/CLOSED)	Accuracy: Exact Reading Frequency: On Change
Additional Requirements for wind farms only:	
Wind speed (0-50 m/s)	Accuracy: -/+ 5% of reading Frequency: 1 per Minute
Wind Direction (0 - 359 deg.) 0 deg denotes FROM due North	Accuracy: -/+ 15 deg. Frequency: 1 per Minute
Power Available (MW)	Accuracy: -/+ 1% of reading Frequency: 1 per Second
Forecasted Output (MW) Note: In alignment with section C. Methodology below	Accuracy: -/+ 1% of reading Frequency: 1 per Second

3.2 Infrastructure

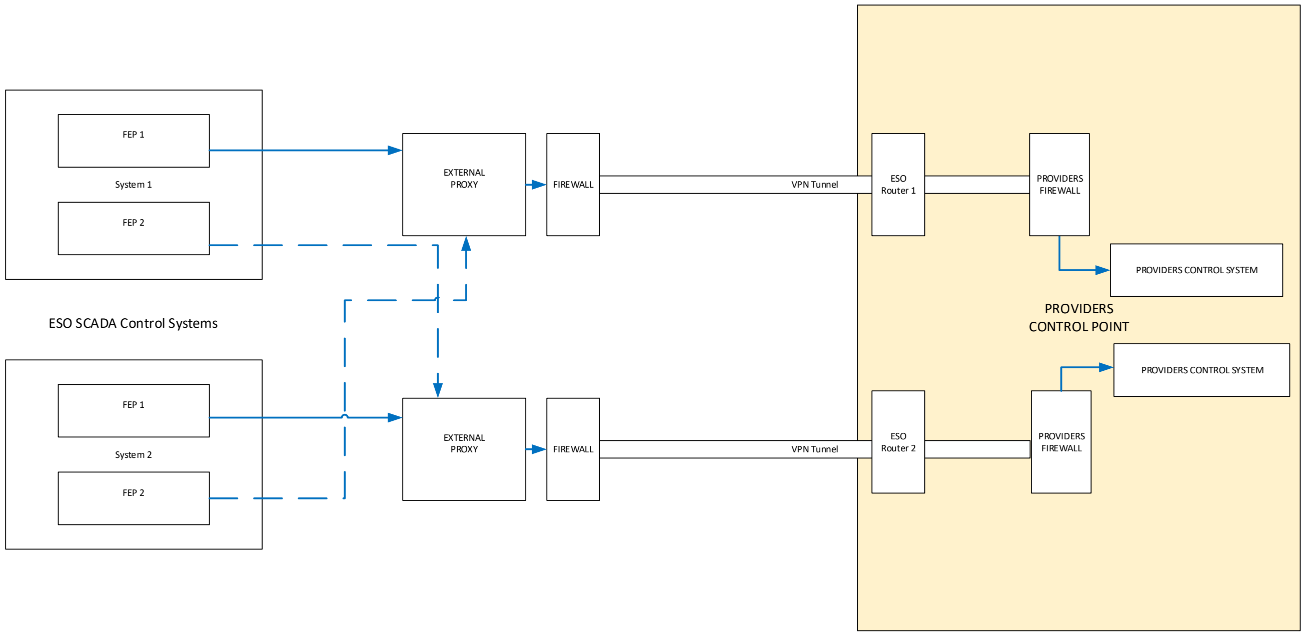
For the data listed above Providers are expected to comply with the following communication requirements:

3.3 Resilient comms

- The ESO requires resilient and mains power independent communication links to allow constant monitoring of the contracted units, before, during and after a restoration event.
- These communication links will be delivered between the ESO and the provider's control point. This is likely to be a combination of National Grid OPTEL and MPLS circuits.
- For existing units with communication links with the ESO these will be reviewed to make sure they meet the requirements of the service, mainly dual resilience, and mains power independence.

3.4 Supported Protocols

- The ESO supports an industry standard SCADA protocol for metering information. This protocol IEC 60870-5 104. The providers SCADA connection needs to support two connections from two independent SCADA systems the ESO operates.
- These systems operate in an ACTIVE / ACTIVE set-up and need to provide the same data to both systems independently. Each system supports an ACTIVE / STANDBY connection supported by the IEC 60870-5 104 protocol.



3.5 Latency

- ESO requires a time stamped value from the meter every 1 second, to arrive at the ESO boundary within 5 seconds of the reading being taken.
- For Aggregated units, the total BMU output should be recalculated every 1 second time stamped and sent onto NGESO.

3.6 Supported Data Types

The following subset of data types will be supported by the IEC104 client.

Data Type	Description	Direction
1–M_SP_NA_1	Single point information	To ESO Only
3–M_DP_NA_1	Double point information	To ESO Only
5–M_ST_NA_1	Step position information	To ESO Only
11–M_ME_NB_1	Measured value, scaled value	To ESO Only
30-M_SP_TB_1	Single point information with time tag CP56Time2a	To ESO Only
31-M_DB_TB_1	Double point information with time tag CP56Time2a	To ESO Only
32-M_ST_TB_1	Step position information with time tag CP56Time2a	To ESO Only
35-M_ME_TE_1	Measured value, scaled value with time tag CP56Time2a	To ESO Only
70-ME_EI_NA_1	End of initialisation	To ESO Only
100-C_IC_NA_1	(General-) Interrogation command	From ESO Only
107-C_TS_TA_1	Test command with time tag CP56Time2a	From ESO Only

Table 1 – IEC104 Data Type Table

Note on Measured Values: Scaled values to be received from the Market Participant need to be scaled to an NMV over 20,000 bits. For example:

Range	NMV	0 in 104	NMV in 104	Scale
-150 to 150	150	0	20,000	0.0075
-300 to 300	300	0	20,000	0.015
-450 to 450	450	0	20,000	0.0225

Table 2 – NMV Scaling Table

The **preferred** Measured Value Data Type is M_ME_TE_1

Note on Time Stamping: Time stamping of Data points is required from source. There is no mechanism to synchronise the time between systems. The Market Participant will be responsible for the correct time stamping at source. All times are to be provided in UTC.

3.7 Availability of Signals

The ESO's requires 99.95% availability of signals from a every unit in the service. The ESO will provide High Available infrastructure it is expected the providers will also implement a High Available infrastructure.

3.8 Communication Failures

- As with all metering equipment, it is prone to occasional failures. When a provider detects a failure in a meter or communication link, the affected data points should have the Not Topical or Invalid flag set to indicate to ESO that these values are no longer valid.
- When metering or communication failure affects an aggregated provider, the sub-unit that has experienced a metering or communication failure should set the affected data points to Suspect or Invalid but should use the last known good value for the aggregated BMU total output. The provider should consider a method of overwriting this value, to make sure the aggregated unit's output is not unduly affected by the metering or communication failure.
- ESO will automatically send an alert to the provider when the client detects a prolonged period of no communication.

3.9 Buffering

To stop links from becoming overloaded and delaying the reporting of real time data, Market Participants should limit the amount of data they are buffering while the server is not communicating to NGESO. NGESO would recommend only storing the last good value in the buffer for Measured Values while the IEC104 link is down.

4. PMU Data

ESO requires PMU data from the Grid Entry point of the Generating unit to be continually sent to ESO's Phasor Data Concentrators.

This data can be sent on the same data links as the Operational Metering.

It must conform to the following standards;

- IEEE C37.118.1a-2014
- IEEE C37.118.2-2011

It must sample at a rate of at least 50 Hz

C. Methodology

The ESO is seeking to standardise the approach as much as possible across providers but we do accept that differences might exist across parties, locations, etc.

With the above in mind to support the methodology outlined below the ESO is expecting to receive from Tender Participants the relevant OEM information (documentation) around:

- Power Curves for each relevant wind turbine type relevant to a potential service.
  - Power curves should be provided for at least 8 equally spaced wind directions.
  - Power Curves shall be supplied in a tabular format, giving the power output to be expected for wind speed values ranging from $0ms^{-1}$ to $50ms^{-1}$, with increments of $0.1ms^{-1}$.
  - Power Curves shall include any cut-in or cut-out effects at defined wind speeds, but for simplicity no hysteresis effects should be included.
- Other metadata for each turbine type, including as a minimum hub height and rotor diameter.

1. Availability of the Service (≥80%)

To be assessed from historical data and/or design documentation.

Providers expected to demonstrate that they will be able to comply with the declared availability.

1.1 Existing Assets

a. Relevant wind probability curves.
b. Half-hourly average figures for wind speed/direction (minimum: past ≥ 12 months, future ≥ 12 months).
c. Availability of Unit(s) considered for the provision of the service (minimum: past ≥ 12 months, future ≥ 12 months).
d. Contribution of Unit(s) considered for the provision of the service (minimum: past ≥ 12 months, future ≥ 12 months).
e. Probability of Power (applying algorithm below) vs Actual Output (minimum: last 12 months)

1.2 New Assets

a. Relevant wind probability curves.
b. Half-hourly average figures for wind speed/direction (minimum: future ≥ 24 months).
c. Availability of Unit(s) considered for the provision of the service (minimum: future ≥ 24 months).
d. Contribution of Unit(s) considered for the provision of the service (minimum: future ≥ 24 months).
e. Probability of Power (applying algorithm below) (minimum: future ≥ 24 months).

2. Resilience (delivery of contracted figures for at least 10h)

Providers expected to implement the algorithm bellow and shall consider it as an integral part of the Operational Metering requirements mentioned above.

2.1 Algorithm – Considerations

a. Power curves will have been supplied for each turbine type and wind direction.
b. Local variations in the observed wind speed and direction data may exist between the two (or more) meteorological masts / nacelle anemometers (e.g., due to wake effects). To account for this in a relatively simple and standardised manner, the maximum measured wind speed across the two (or more) meteorological masts / nacelle anemometers will be adopted as the ‘observed’

wind speed. For wind direction, an average across the two (or more) meteorological masts / nacelle anemometers will be taken as representative of the farm area.

c. For each timestamp, the total power output of the farm will be calculated as follows:
   - For each turbine type, the Power Curve associated with the measured/forecasted wind direction will be selected.
   - This Power Curve will be used to convert the wind speed to MW power output, interpolating linearly between the values in the power curve table.
   - The power output per turbine will be multiplied by the number of available turbines of that type.
   - The power output for all turbine types will be summed to determine the total power output of the wind farm.

For reference, examples are shown below for both the wind speed / direction table and power curve table format (using 8 equally-spaced directional bins).

Datetime **(UTC)**	**Wind speed (m/s)**	**Direction (degrees North)**
2021-01-01 00:00:00		
2021-01-01 00:30:00		
2021-01-01 01:00:00		
2021-01-01 01:30:00		
2021-01-01 02:00:00		
…		

Table 3 – Example wind speed table.

Wind speed (m/s)	**Wind direction (bin centre, degrees North)**							
	0.0	*45.0*	*90.0*	*135.0*	*180.0*	*225.0*	*270.0*	*315.0*
0.0								
0.1								
0.2								
0.3								
0.4								
…								

Table 4 – Example power curve table.

2.2 Algorithm

License

ESO

Python Software Requirements

```
# This file may be used to create an environment using:
# $ conda create --name <env> --file <this file>
# platform: win-64
blas=1.0=mkl
bottleneck=1.3.5=py39h080aedc_0
ca-certificates=2023.01.10=haa95532_0
certifi=2022.12.7=py39haa95532_0
intel-openmp=2021.4.0=haa95532_3556
mkl=2021.4.0=haa95532_640
mkl-service=2.4.0=py39h2bbff1b_0
mkl_fft=1.3.1=py39h277e83a_0
mkl_random=1.2.2=py39hf11a4ad_0
numexpr=2.8.4=py39h5b0cc5e_0
numpy=1.23.5=py39h3b20f71_0
numpy-base=1.23.5=py39h4da318b_0
openssl=1.1.1s=h2bbff1b_0
packaging=22.0=py39haa95532_0
pandas=1.5.2=py39hf11a4ad_0
pip=22.3.1=py39haa95532_0
python=3.9.16=h6244533_0
python-dateutil=2.8.2=pyhd3eb1b0_0
pytz=2022.7=py39haa95532_0
setuptools=65.6.3=py39haa95532_0
six=1.16.0=pyhd3eb1b0_1
sqlite=3.40.1=h2bbff1b_0
tzdata=2022g=h04d1e81_0
vc=14.2=h21ff451_1
vs2015_runtime=14.27.29016=h5e58377_2
wheel=0.37.1=pyhd3eb1b0_0
wincertstore=0.2=py39haa95532_2
```

Example Python Code

```
#!/usr/bin/env python3

from pathlib import Path

import numpy as np
import pandas as pd
from datetime import datetime
from random import random

#%% run_demo function
def run_demo(wind, pcurve):
    """
    This demo function takes a wind speed and direction data, and converts to power output, using
    directionally-varying power curves.
```

```
    The intention of this code is to provide the user with a simple demonstration of the method
    outlined in the accompanying documentation.
    ESO accepts no liability as to the accuracy of this code. Tender participants are responsible for
    the implementation of any code used as part of the trial or for any other purpose.

    Args:
        wind (pd.Dataframe): Wind data as a function of time (index), including "speed" and "direction" columns
        pcurve (pd.Dataframe): Power conversion curves as a function of wind speed (index) and direction
(column)
    """
    # loop through directional bins and group wind speed data
    dbins = pcurve.columns.values.astype('float')
    ddelta = 0.5*(dbins[1] - dbins[0])
    power = []
    for xd, di in enumerate(dbins):
        # first, calculate the bin edges for this bin
        dstart = di - ddelta
        dend = di + ddelta
        if dstart < 0:
            # bin includes the 0/360 boundary
            dstart += 360.0
            wind_di = wind[(wind['direction'] >= dstart) | (wind['direction'] < dend)]
        else:
            # bin does not include the 0/360 boundary
            wind_di = wind[(wind['direction'] >= dstart) & (wind['direction'] < dend)]

        # apply directional power curve to the selected wind speed data
        pcurve_di = pcurve.iloc[:,xd]
        # use linear interpolation between steps of power curve
        power_di = np.interp(x=wind_di.loc[:,'speed'].values,
                             xp=pcurve_di.index.values,
                             fp=pcurve_di.values)
        power.append(pd.Series(data=power_di, index=wind_di.index))

    # combine and sort power data
    power = pd.concat(power, axis=0).to_frame('power').sort_index()
    return power


#%% main
if __name__ == '__main__':
    # mock up some test wind speed and direction data (for demo purposes only)
    df_wind_test = pd.DataFrame({'speed': [4.5, 6.73, 8.8, 15.0, 1.5],
                                 'direction': [359.1, 1.5, 45.6, 176.5, 265.0]})
    df_wind_test.index = pd.date_range(start=datetime(2021,1,1),
                                       end=datetime(2021,1,1,2),
                                       freq='30T')

    # mock up a set of dummy power curves (for demo purposes only)
    udata = np.arange(0,50.1,0.1)
    u00 = 2.0
    pdata = ((udata-u00).clip(min=0))**3
    irated = np.argwhere(udata>=15).min()
    pdata[irated:] = pdata[irated]
    icutout = np.argwhere(udata>=25).min()
```

```
pdata[icutout:] = 0
pdata *= 100.0/max(pdata)      # e.g. for a 100 MW farm

all_directions = np.linspace(0,315,8)
all_directions = np.asarray([f"{d:.2f}" for d in all_directions])   # make string to avoid floating point errors
df_pcurve = pd.DataFrame(index=udata, columns=all_directions, dtype='float')
for direction in all_directions:
    df_pcurve.loc[:,direction] = pdata*0.5*(1+random())

# run demo
df_power = run_demo(wind=df_wind_test, pcurve=df_pcurve)
```