**A logo with a map and blue text

Description automatically generated**

**Wind and Solar Power Forecast Web Service:   
Instructions for Getting Started & Using “Sandbox WPF WebServices – WPLP” and “WPF WebServices – WPLP”**

**Objective:**

This guide aims to help Lead Participants set up a Python environment to interact with the Wind Plant Lead Participant (WPLP) Web Service API or the Solar Plant Lead Participant (SPLP) Web Service API. By the end of this guide, Lead Participants will understand how to install necessary tools, set up a development environment, and execute Wind and Solar Power Forecast Web Service calls.

**WPFA/SPFA:**

Submitting Wind Plant Forecast Availability (WPFA) and Solar Plant Forecast Availability (SPFA) data is the only required API interaction. This guide demonstrates how to submit sample WPFA/SPFA data and aims to minimize the steps for doing so. However, automating and scheduling this API interaction and tailoring it to a specific Lead Participant's WPFA/SPFA data is outside of the scope of this guide.

**Disclaimer**:

This guide is specifically designed for Microsoft Windows operating systems. macOS and Linux users may refer to it generally, but will need to make adjustments for their respective operating systems. Additionally, this guide utilizes an expired Development certificate, so some steps may differ from those using a current Production certificate, which is necessary to submit WPFA/SPFA data.

**Table of Contents**

[**0. Unzip the Folder** 4](#_Toc175917171)

[0.1. Extract Files from the Zipped Project Folder 4](#_Toc175917172)

[**1. Configuring Certificates** 5](#_Toc175917173)

[1.1. Importing Client Certificate 5](#_Toc175917174)

[1.2. Exporting CA Certificate 9](#_Toc175917175)

[**2. Setting up Python Integrated Development Environment (IDE) with PyCharm** 14](#_Toc175917176)

[2.1. Install PyCharm 14](#_Toc175917177)

[2.2. Setting up a New Project 14](#_Toc175917178)

[**3. Configuring Properties** 16](#_Toc175917179)

[3.1. Navigate to the Properties.py Tab 16](#_Toc175917180)

[3.2. ENDPOINT\_URL 16](#_Toc175917181)

[3.3. CA\_CERT\_PATH 16](#_Toc175917182)

[3.4. CLIENT\_CERT\_PATH 17](#_Toc175917183)

[3.5. CERT\_PASSWORD 17](#_Toc175917184)

[**4. Converting and Combining Certificates** 18](#_Toc175917185)

[4.1. Navigate to the Unzipped Project Folder 18](#_Toc175917186)

[**5. Running the GUI and interacting with the API** 19](#_Toc175917187)

[5.1. Run the GUI 19](#_Toc175917188)

[5.2. Fetch Categories 20](#_Toc175917189)

[5.3. Select Category 21](#_Toc175917190)

[5.4. Fetch Entities 22](#_Toc175917191)

[5.5. Select Entity 23](#_Toc175917192)

[5.6. Fetch Schedules 24](#_Toc175917193)

[5.7. Select Schedule 25](#_Toc175917194)

[5.8. Fetch Forecasts 27](#_Toc175917195)

[5.9. Create and view CSV 28](#_Toc175917196)

[5.10. Submit Forecast 30](#_Toc175917197)

[**6. Running Functions** 31](#_Toc175917198)

[6.1. Navigate to Calling\_API.py tab 31](#_Toc175917199)

[6.2. Calling make\_request\_categories() 31](#_Toc175917200)

[6.3. Calling make\_request\_entities() 35](#_Toc175917201)

[6.4. Calling make\_request\_schedules() 38](#_Toc175917202)

[6.5. Calling make\_request\_forecasts() 44](#_Toc175917203)

[6.6. Calling submit\_forecast() 45](#_Toc175917204)

[**7. Getting Forecast Values** 49](#_Toc175917205)

[7.1. Save as a csv 49](#_Toc175917206)

[7.2. Read the csv file in Python and convert to a list: 49](#_Toc175917207)

[7.3. Use the list as input to submit\_forecast: 50](#_Toc175917208)

[**8.** **Debugging/ Error Handling** 50](#_Toc175917209)

[8.1. Setting up debugging 50](#_Toc175917210)

[8.2. Debugging output 50](#_Toc175917211)

[8.3. XML output 51](#_Toc175917212)

[**9. Function Definitions** 52](#_Toc175917213)

[9.1. make\_request\_categories() 52](#_Toc175917214)

[9.2. make\_request\_schedules() 53](#_Toc175917215)

[9.3. make\_request\_entities(cat=None) 54](#_Toc175917216)

[9.4. make\_request\_forecasts(entityID, entityAssetID, scheduleID) 55](#_Toc175917217)

[9.5. generate\_power\_entries(start\_time, num\_entries, power\_values) 56](#_Toc175917218)

[9.6. submit\_forecast(schedule\_id, entity\_id, entity\_asset\_id, power\_values, forecast\_type) 57](#_Toc175917219)

**0. Unzip the Folder**

**0.1. Extract Files from the Zipped Project Folder**

**0.1.1.** Extract the files from the zipped project folder provided to the Lead Participant using extraction software.

Example: Right click on the folder and select “Extract All…”

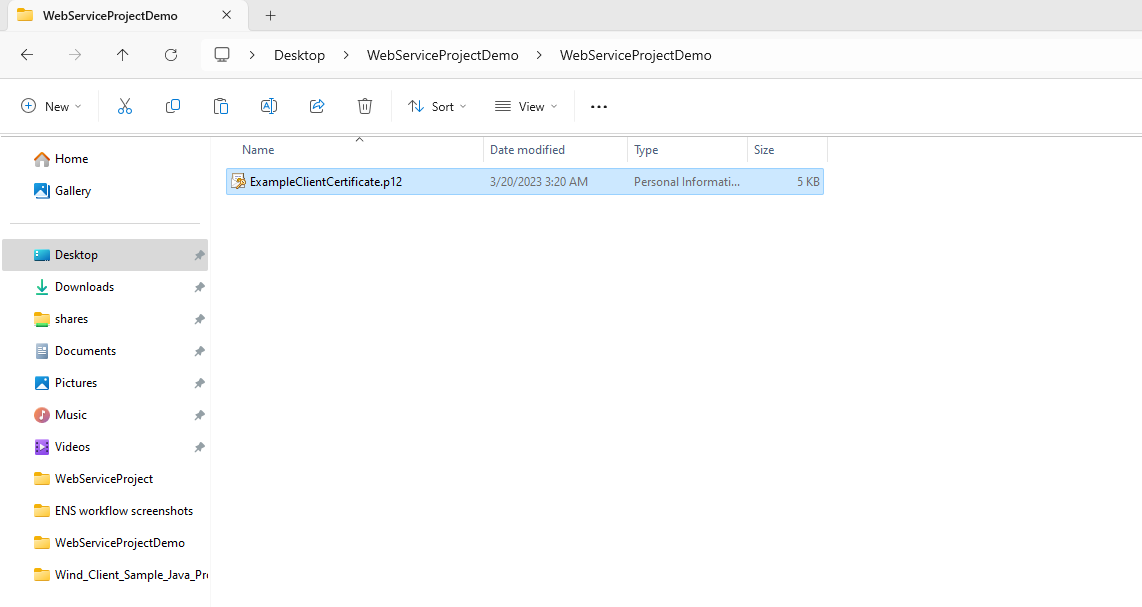
**0.1.2.** Keep track of this unzipped project folder; it will be used in future steps.

**1. Configuring Certificates**

**1.1.** **Importing Client Certificate**

Refer to the instructions below, or follow along with this [video](https://iso-ne.my.site.com/s/article/How-to-import-install-a-digital-certificate).

**1.1.1.** Locate the client certificate given by ISO-NE and click on it.

****

**1.1.2.** Store Location should be set to Current User.

**1.1.2.1.** Click Next.

**A screenshot of a certificate import wizard

Description automatically generated**

**1.1.3.** File name will default to the name of the client certificate file.

**1.1.3.1.** Click Next.

**A screenshot of a computer

Description automatically generated**

**1.1.4.** Input certificate password.

**1.1.4.1.** Deselect the box next to “Enable strong private key protection. You will be prompted every time the private key is used by an application if you enable this option.”

**1.1.4.2.** Click the box next to “Mark this key as exportable. This will allow you to back up or transport your keys at a later time.”

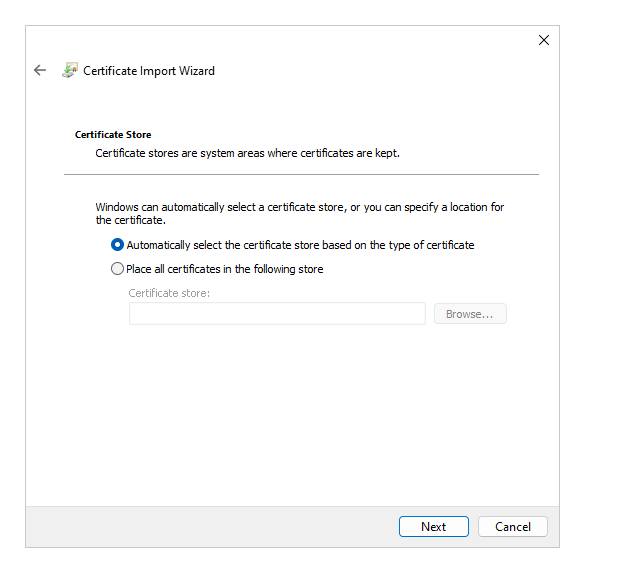
**1.1.4.3.** Click Next.

**A screenshot of a computer

Description automatically generated**

**1.1.5.** Select a specific keystore or allow automatic selection by selecting “Automatically select the certificate store based on the type of certificate.”

**1.1.5.1.** Click Next.

****

**1.1.6.** Click Finish.

**A screenshot of a certificate

Description automatically generated**

**1.1.7.** A pop-up should appear that looks like the figure below.

**1.1.7.1.** Click Ok.

A screenshot of a computer program

Description automatically generated

**1.1.8.** A message will pop up indicating that the import was successful.

**1.1.8.1.** Click Ok.

A screenshot of a message

Description automatically generated

**1.2.** **Exporting CA Certificate**

**1.2.1.** Copy one of these links and paste it in the search bar.

\*\*\*It is strongly recommended that Lead Participants set up and test Wind and Solar Power Forecast Web Service in the **Sandbox environment** (Sandbox WPF WebServices – WPLP), before making submissions in the Production environment (WPF WebServices – WPLP).\*\*\*

**1.2.1.1.** Use the **Sandbox** environment for initial setup and testing.

Sandbox url**:** <https://sandboxsmd.iso-ne.com/wpfmui/webservices/WindPlantLeadParticipant/1_0/?wsdl>

**1.2.1.2.** Use the **Production** environment when ready to start submitting WPFA/SPFA.

Production url**:** <https://smd.iso-ne.com/wpfmui/webservices/WindPlantLeadParticipant/1_0/?wsdl>

**1.2.2.** Click Advanced.

A screenshot of a computer

Description automatically generated

**1.2.3.** Click Proceed to wpfmui01d.iso-ne.com (unsafe).

A screenshot of a computer error message

Description automatically generated

**1.2.4.** Select the certificate you want to use and click OK.

A screenshot of a computer

Description automatically generated

**1.2.5.** Click on the Windows Security pop-up from the taskbar.

A screenshot of a computer

Description automatically generated

**1.2.6.** A pop-up should appear that looks like the figure below.

**1.2.6.1.** Click Allow.

A screenshot of a computer

Description automatically generated

**1.2.7.** Click on the “Not secure” button to the left of the url.

A screenshot of a computer screen

Description automatically generated

**1.2.8.** Click “Certificate is not valid.”

Note: In the figure below, the certificate is expired. If following these instructions, the certificate should not be expired, so the button will be labeled differently.

A screenshot of a computer screen

Description automatically generated

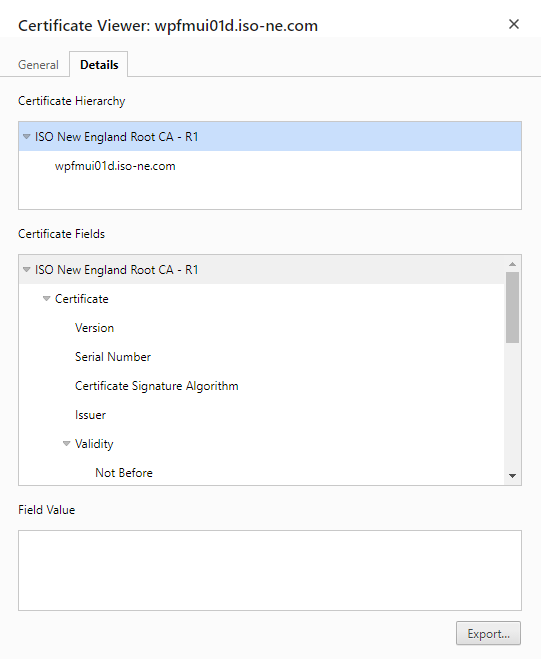
**1.2.9.** Go to the Details tab.

A screenshot of a computer

Description automatically generated

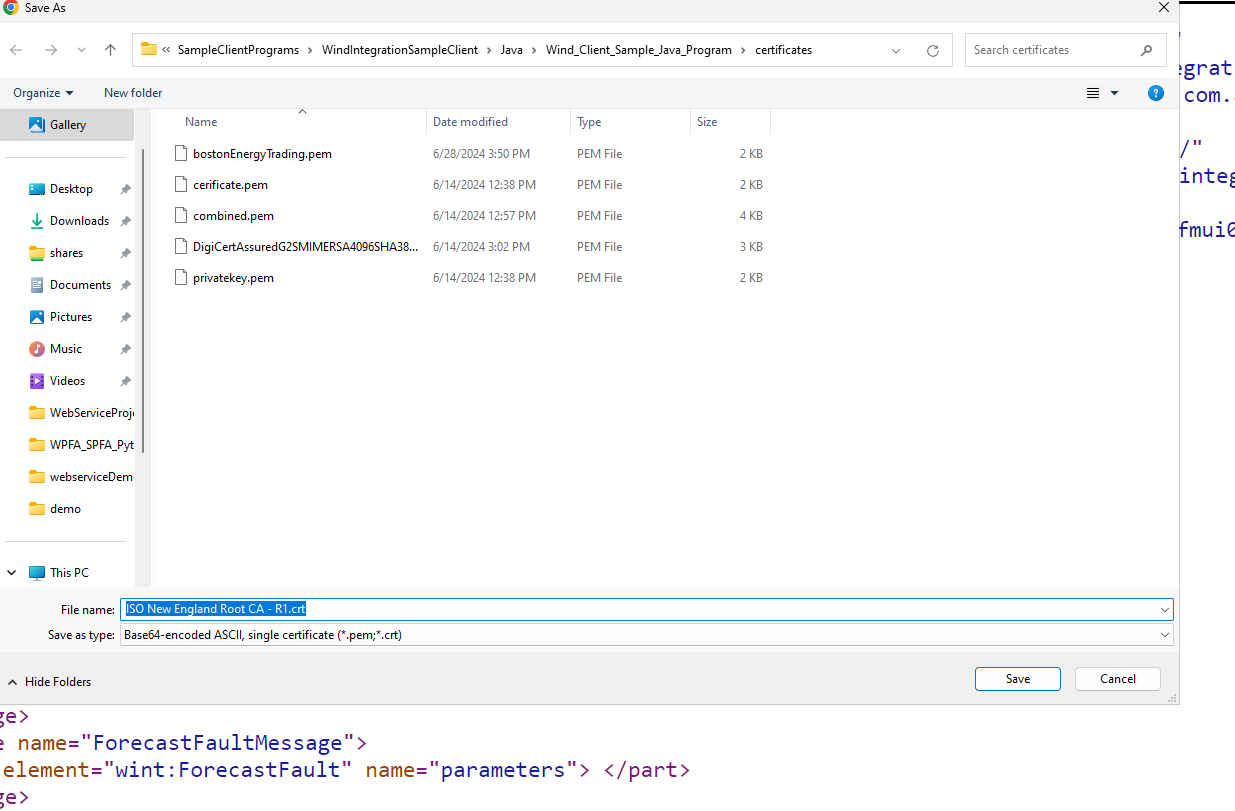
**1.2.10.** Click on the first certificate under “Certificate Hierarchy” and click “Export…”

Example: ISO New England Root CA - R1”



**1.2.11.** Navigate to the unzipped project folder that has the client certificate.

**1.2.11.1.** Click Save.



**2. Setting up Python Integrated Development Environment (IDE) with PyCharm**

**2.1.** **Install PyCharm**

Note: Skip this step if PyCharm is already installed.

**2.1.1.** Visit the [PyCharm website](https://www.jetbrains.com/pycharm/download/) and download Python version 3.11 or higher.

**2.1.1.1.** If PyCharm doesn’t automatically download Python, follow this link to download it: <https://www.python.org/ftp/python/3.11.0/python-3.11.0-amd64.exe>.

**2.1.2.** Follow the installation instructions.

**2.2.** **Setting up a New Project**

Note: If PyCharm was previously installed, select File and click New Project.

**2.2.1.** Open PyCharm and select “Create New Project.”

**2.2.1.1.** Configure the project to look like the figure below.

A screenshot of a computer program

Description automatically generated

**2.2.2** Select File and click Open.

A screenshot of a computer

Description automatically generated

**2.2.3.** Navigate to the unzipped project folder and click OK.

A screenshot of a computer

Description automatically generated

**2.2.4.** Open Python files by double clicking on each of the six .py files to view them in the Code Editor.

A screenshot of a computer

Description automatically generated

**3. Configuring Properties**

**3.1. Navigate to the Properties.py Tab**

**A screen shot of a computer

Description automatically generated**

**3.2. ENDPOINT\_URL**

**3.2.1.** Copy either the Sandbox (recommended for testing) or Production url provided below and paste it into the “ENDPOINT\_URL” variable.

* + Use the **Sandbox** environment for initial setup and testing
    - **Sandbox url:** <https://sandboxsmd.iso-ne.com/wpfmui/webservices/WindPlantLeadParticipant/1_0/?wsdl>
  + Use the **Production** environment when ready to start submitting WPFA/SPFA.
    - **Production url:** <https://smd.iso-ne.com/wpfmui/webservices/WindPlantLeadParticipant/1_0/?wsdl>

**3.2.2.** Ensure the url is surrounded by only one set of quotes with a letter r in front of it.

**3.3. CA\_CERT\_PATH**

**3.3.1.** Copy and paste the path to the CA Certificate into the “CA\_CERT\_PATH” variable.

* + Get the path by navigating to the certificate on your computer (**look for the .crt file**), right click on it, and select “Copy as path” or go to Properties 🡪 Location 🡪 Copy the Location and add the file name.

**3.3.2.** Ensure the path is surrounded by only one set of quotes with a letter r in front of it.

**3.4. CLIENT\_CERT\_PATH**

**3.4.1.** Copy and paste the path to the Client Certificate into the “CLIENT\_CERT\_PATH” variable.

* + Get the path by navigating to the certificate on your computer (**look for the .p12 file**), right click on it, and select “Copy as path” or go to Properties 🡪 Location 🡪 Copy the Location and add the file name.

**3.4.2.** Ensure the path is surrounded by only one set of quotes with a letter r in front of it.

**3.5. CERT\_PASSWORD**

**3.5.1.** Enter the certificate password into the “CERT\_PASSWORD” variable.

* + Note: This is the same password entered in step 1.1.4.

**3.5.2.** Ensure the password is surrounded by only one set of quotes with a letter r in front of it.

**4. Converting and Combining Certificates**

**4.1. Navigate to the Unzipped Project Folder**

**4.1.1.** Double click the file “batchCall.bat”

* This will install the necessary requirements for the Python project and convert and combine the certificates into a usable .pem format.

**A screenshot of a computer

Description automatically generated**

**5. Running the GUI and interacting with the API**

We will use the graphical user interface (GUI) to explain and visualize how the Wind Plant Lead Participant (WPLP) Web Service API and the Solar Plant Lead Participant (SPLP) Web Service API work and how to submit WPFA/SPFA data.

**5.1.** **Run the GUI**

**5.1.1** Navigate to the GUI.py tab.

**5.1.2.** Click the green play button at the top of the page to run the file.



* Alternatively, right click into the Code Editor and click Run ‘GUI’ or click “Ctrl” + “Shift” + “F10”
* This may take a while to run; please be patient.

**5.1.3.** There should be a pop-up window like the one below; make it full screen.

A screenshot of a computer

Description automatically generated

**5.2.** **Fetch Categories**

**5.2.1.** Click the Fetch Categories button.

* + This calls the [make\_request\_categories()](#n3r6x756jy1d) function.
  + Category information should appear in the Category Output panel on the left above the Fetch Categories button.
  + Category buttons (AREA, SPLANT, SYSTEM, WPLANT) should appear in the Category Buttons panel on the right.

**A screenshot of a computer

Description automatically generated**

**5.3. Select Category**

**5.3.1.** Select a category by clicking on one of the buttons (AREA, SPLANT, SYSTEM, WPLANT) in the Category Buttons panel on the right.

A Category Identifier (ex. SPLANT) should appear in the Entities Inputs panel after the selection.

* + Selecting the “SPLANT” Category Identifier allows one to view solar plant entities, query solar plant forecasts, and submit solar plant future availability data by passing the identifier as input to the [make\_request\_entities](#ydlgdutk1rc3)(), [make\_request\_forecasts](#ohasw8v9ly)() and [submit\_forecast](#xoqbyatjkq38)() functions.
  + Selecting the “WPLANT” Category Identifier allows one to view wind plant entities, query wind plant forecasts, and submit wind plant future availability data by passing the identifier as input to the [make\_request\_entities](#ydlgdutk1rc3)(), [make\_request\_forecasts](#ohasw8v9ly)() and [submit\_forecast](#xoqbyatjkq38)() functions.

A screenshot of a computer

Description automatically generated

**5.4.** **Fetch Entities**

**5.4.1.** Click the Fetch Entities button.

* + This calls the [make\_request\_entities](#ydlgdutk1rc3)() function.
  + Entity information should appear in the Entities Output panel on the left.
  + Entity buttons should appear in the Entities Buttons panel on the right.

Example: In the figure below, SPLANT is the selected Category Identifier, so after clicking the Fetch Entities button, information for all the available solar plant entities is displayed in the Entities Output panel and individual entities appear as clickable buttons in the Entities Buttons panel.

**A screenshot of a computer

Description automatically generated**

**5.5. Select Entity**

**5.5.1.** Select an entity by clicking a button in the Entities Buttons panel.

* + Scroll down using the scroll bar on the right**.**
  + The Entity ID, Entity’s Asset Identifier, and Entity Name fields in the Forecasts Inputs panel should now have data.
  + Selecting an entity allows one to get forecast data and submit forecast availability data for that specific entity by using the entity as input to the [make\_request\_forecasts](#ohasw8v9ly)() and [submit\_forecast](#xoqbyatjkq38)() functions.
  + Note: Only the Entity’s Asset Identifier will be used as input to the [make\_request\_forecasts](#ohasw8v9ly)() and [submit\_forecast](#xoqbyatjkq38)() functions; the Entity ID and Entity Name are only displayed for reference purposes.

A screenshot of a computer

Description automatically generated

**5.6.** **Fetch Schedules**

**5.6.1.** Click the Fetch Schedules button in the Schedules Output panel.

* + This calls the [make\_request\_schedules()](#fxfoa7r8w8wf) function.
  + This query does not require inputs.
  + Schedule information should appear in the Schedules Output panel on the left.
  + Schedule buttons should appear in the Schedules Buttons panel on the right.

A screenshot of a computer

Description automatically generated

**5.7. Select Schedule**

**5.7.1.** Select a schedule by clicking one of the buttons (Composite Short Term Wind Plant Forecast, Medium Term Wind Power Forecast, etc.) in the Schedules Buttons panel.

* + Schedules represent one of five forecast types for a specific Category Identifier.
  + The buttons ending in “Forecast Availability” are used to submit WPFA/SPFA data. They give data.
  + The buttons ending in “Forecast” are used to query forecasts. They get data.

|  |  |  |
| --- | --- | --- |
| **Types of Schedules** | **Wind Entities** | **Solar Entities** |
| “Forecast” – used to query forecasts | Composite Short Term Wind Plant Forecast | Composite Short-Term Forecast |
| “Forecast” – used to query forecasts | Medium Term Wind Power Forecast | Medium Term Solar Power Forecast |
| “Forecast” – used to query forecasts | Long Term Wind Power Forecast | Long Term Solar Power Forecast |
| “Forecast Availability” – used to submit WPFA/SPFA data | Hourly Wind Plant Forecast Availability | Hourly Solar Plant Forecast Availability |
| “Forecast Availability” – used to submit WPFA/SPFA data | Daily Wind Plant Forecast Availability | Daily Solar Plant Forecast Availability |

**5.7.2.** Select a schedule used for querying forecasts by selecting a button ending in “Forecast.”

* + Ensure the previously selected Category Identifier (AREA, SPLANT, SYSTEM, WPLANT) and the Schedules Buttons type (i.e. choose Composite Short Term Wind Plant Forecast for WPLANT or Composite Short-Term Forecast for SPLANT) align.
  + After selecting a schedule, the Schedule ID and Schedule Type fields in the Forecasts Inputs panel will have data.
  + Note: Only the Schedule ID is used as input to the [make\_request\_forecasts](#ohasw8v9ly)() function; the Schedule Type is only displayed for clarity.

**A screenshot of a computer

Description automatically generated**

**5.8.** **Fetch Forecasts**

**5.8.1.** Click the Fetch Forecasts button in the Forecasts Output panel.

* + This calls the [make\_request\_forecasts](#ohasw8v9ly)() function.
  + There should now be time and value data in the scrollable Forecasts Output panel.
  + This is the predicted forecast for the entity and schedule that were selected in the steps above. This forecast data can be used for future planning, and to inform market decisions.

**A screenshot of a computer

Description automatically generated**

**5.9.** **Create and view CSV**

**5.9.1.** Scroll down to the CSV Browser panel.

**5.9.2.** Click the Create CSV button in the CSV Output panel.

**5.9.3.** Click the Browse CSV button in the CSV Output panel.

**5.9.4.** Select SampleData.csv

**5.9.5.** Click Open

Entity buttons should now be displayed in the CSV Buttons panel.

A screenshot of a computer

Description automatically generated

**5.9.6.** Click an entity button in the CSV Buttons panel.

A screenshot of a computer

Description automatically generated

**5.9.7.** Time and value data should now appear in the CSV Output panel.

* + These values represent example Forecast Availability data; we will submit this data in the next step.

A screenshot of a computer

Description automatically generated

**5.10.** **Submit Forecast**

**5.10.1.** Click the Submit Forecast button in the Submit Output panel.

* + This calls the [submit\_forecast](#xoqbyatjkq38)() function.
  + Before clicking the Submit Forecast button, make sure the Forecast Type is compatible.
    - If the forecast is not compatible (ex. Non Writeable Forecast), scroll up to the Schedules Buttons and select one that aligns with the selected Category Identifier and one ending in “Forecast Availability.” Ex. WPLANT (under Category Buttons) and Daily Wind Plant Forecast Availability (under Schedules Buttons).
    - Once aligned, return to the Submit Output panel and click the Submit Forecast button.
  + In the Submit Output panel, there should be a success message that contains a transaction ID.

A screenshot of a computer

Description automatically generated

**5.10.2.** Exit out of the GUI window to stop it from running in the background for the remaining steps.

**6. Running Functions**

**6.1. Navigate to Calling\_API.py tab**

**6.1.1.** Go to the Making API Calls (uncomment functions to run).

Note: The color of the text in this section will be gray.

**A screenshot of a computer

Description automatically generated**

**6.2.** **Calling** [**make\_request\_categories()**](#n3r6x756jy1d)

**6.2.1.** Highlight the function call starting with # # Query Categories.

**A screenshot of a computer

Description automatically generated**

**6.2.2.** Uncomment code by clicking “Ctrl” + “/”

**A screenshot of a computer

Description automatically generated**

**6.2.3.** Click the green play button at the top to query the categories.



* Alternatively, right click into the Code Editor and click Run ‘Calling\_API’ or click “Ctrl” + “Shift” + “F10”

**6.2.4.** Inspect the output.

* + The output should automatically pop-up, but if it doesn’t, select the white play button on the left.



**6.2.5.** Select an identifier from the function output.

**6.2.5.1.** Highlight the selected identifier.

Example: SPLANT for solar plants.

**6.2.5.2**. Click “Ctrl” + “C” to copy the identifier.

A screenshot of a computer

Description automatically generated

**6.2.6.** Go to the Assigning Variables section.

**6.2.6.1.** Enter the selected identifier into the categoryID variable by pasting it (“Ctrl” + “V”) into the quotation marks.

A screenshot of a computer

Description automatically generated

**6.3.** **Calling** [**make\_request\_entities**](#ydlgdutk1rc3)**()**

**6.3.1.** Highlight the function call starting with # # Query Entities.

A screenshot of a computer program

Description automatically generated

**6.3.2.** Uncomment code by clicking “Ctrl” + “/”

A screenshot of a computer

Description automatically generated

**6.3.3.** Click the green play button at the top to query the entities.



* Alternatively, right click into the Code Editor and click Run ‘Calling\_API’ or click “Ctrl” + “Shift” + “F10”

**6.3.4.** Inspect the output.

* + The output should automatically pop-up, but if it doesn’t, select the white play button on the left.



**6.3.5.** Select an assetIdentifier from the function output.

**6.3.5.1.** Highlight the selected assetIdentifier.

**6.3.5.2**. Click “Ctrl” + “C” to copy the assetIdentifier.

A screenshot of a computer

Description automatically generated

**6.3.6.** Go to the Assigning Variables section.

**6.3.6.1.** Enter the selected assetIdentifier into the entityAssetID variable by pasting it (“Ctrl” + “V”) into the quotation marks.

**A screenshot of a computer program

Description automatically generated**

**6.4.** **Calling** [**make\_request\_schedules()**](#fxfoa7r8w8wf)

**6.4.1.** Highlight the function call starting with # # Query Schedules.

A screenshot of a computer

Description automatically generated

**6.4.2.** Uncomment code by clicking “Ctrl” + “/”

A screenshot of a computer

Description automatically generated

**6.4.3.** Click the green play button at the top to query the schedules.



* Alternatively, right click into the Code Editor and click Run ‘Calling\_API’ or click “Ctrl” + “Shift” + “F10”

**6.4.4.** Inspect the output.

* + The output should automatically pop-up, but if it doesn’t select the white play button on the left.



**6.4.5.** Select an identifier from the function output.

* + Ensure the description aligns with the previously selected Category Button and ends in Forecast (ex. Composite Short Term (Wind or Solar) Plant Forecast, Medium Term (Wind or Solar) Power Forecast, Long Term (Wind or Solar) Power Forecast).

**6.4.5.1.** Highlight the selected identifier.

**6.4.5.2**. Click “Ctrl” + “C” to copy the identifier.

A screenshot of a computer

Description automatically generated

**6.4.6.** Go to the Assigning Variables section.

**6.4.6.** Enter the selected identifier into the scheduleID\_query variable by pasting it (“Ctrl” + “V”) into the quotation marks.

**A screenshot of a computer program

Description automatically generated**

**6.4.7.** Select an identifier for an hourly schedule from the function output.

* + Ensure the description aligns with the selected Category Button and that it says Hourly (Wind or Solar) Plant Forecast Availability.

**6.4.7.1.** Highlight the selected identifier.

**6.4.7.2**. Click “Ctrl” + “C” to copy the identifier.

A screenshot of a computer

Description automatically generated

**6.4.8.** Enter the selected identifier into the “ScheduleID\_submitHourly” variable by pasting it (“Ctrl” + “V”) into the quotation marks.

**A screenshot of a computer

Description automatically generated**

**6.4.9.** Select an identifier for a daily schedule from the function output.

* + Ensure the description aligns with the selected Category Button and that it says Daily (Wind or Solar) Plant Forecast Availability.

**6.4.9.1.** Highlight the selected identifier.

**6.4.9.2**. Click “Ctrl” + “C” to copy the identifier.

A screenshot of a computer

Description automatically generated

**6.4.10.** Go to the Assigning Variables section.

**6.4.10.1.** Enter the selected identifier into the scheduleID\_submitDaily variable by pasting it (“Ctrl” + “V”) into the quotation marks.

**A screenshot of a computer

Description automatically generated**

**6.5.** **Calling** [**make\_request\_forecasts**](#ohasw8v9ly)**()**

**6.5.1. Highlight the function call starting with # # Query Forecasts.**

**A screenshot of a computer

Description automatically generated**

**6.5.2. Uncomment code by clicking “Ctrl” + “/”**

**A screenshot of a computer

Description automatically generated**

**6.5.3. Click the green play button at the top to query the forecasts.**

****

* **Alternatively, right click into the Code Editor and click Run ‘Calling\_API’ or click “Ctrl” + “Shift” + “F10”**

**6.5.4. Inspect the output.**

* + **The output should automatically pop-up, but if it doesn’t, select the white play button on the left.**
  + **The output should contain a list of time values and their associated power values, which is the forecast data.**

****

**6.6.** **Calling** [**submit\_forecast**](#xoqbyatjkq38)**()**

**6.6.1. Submitting the Hourly Forecast.**

**6.6.1.1. Highlight the function call starting with # # Submit hourly forecast.**

**A screenshot of a computer

Description automatically generated**

**6.6.1.2. Uncomment code by clicking “Ctrl” + “/” (This will also uncomment the power\_values\_hourly variable.)**

* + - **This makes a list of 48 values of eight sample data that will be submitted when calling submit\_forecast(). Look at the** [**Getting Forecast Values**](#y11gd5vyyw7) **section below for more information on submitting power values.**

**A screenshot of a computer

Description automatically generated**

**6.6.1.3. Click the green play button at the top to submit the hourly forecast.**

****

**6.6.1.4. Inspect the output.**

* + **The output should automatically pop-up, but if it doesn’t, select the white play button on the left.**

****

* + **A message should appear indicating that the query succeeded followed by a transaction ID.**

**6.6.2. Submitting the Daily Forecast.**

**6.6.1.1. Highlight the function call starting with # Submit daily forecast.**

**A screenshot of a computer

Description automatically generated**

**6.6.1.2. Uncomment code by clicking “Ctrl” + “/” (This will also uncomment the power\_values\_daily variable.)**

* + - **This makes a list of 120 values of eight sample data that will be submitted when calling submit\_forecast(). Look at the** [**Getting Forecast Values**](#y11gd5vyyw7) **section below for more information on submitting power values.**

**A screenshot of a computer

Description automatically generated**

**6.6.1.3. Click the green play button at the top to submit the daily forecast.**

****

**6.6.1.4. Inspect the output.**

* + **The output should automatically pop-up, but if it doesn’t, select the white play button on the left.**

****

* + **A message should appear indicating that the query succeeded followed by a transaction ID.**

**7.** **Getting Forecast Values**

**There are many formats for saving numerical values and there are many ways to convert these formats into a comma separated list in Python. However it is done, the** [**submit\_forecast**](#_r4zprmwdr6fp) **function requires the “powerValues” input to be a comma separated list. Submitting the hourly forecast availability requires a list of 48 values, and submitting the daily forecast availability requires a list of 120 values. Currently, when the** [**submit\_forecast**](#_r4zprmwdr6fp) **function is called to submit an hourly forecast, it submits 48 prediction values for 48 hours into the future starting at the top of the hour. When the** [**submit\_forecast**](#_r4zprmwdr6fp) **function is called to submit a daily forecast, it submits 120 prediction values for 120 hours into the future starting 2 days in the future at 1100. For more information on this, refer to** [**OP-14 Appendix F**](https://www.iso-ne.com/static-assets/documents/rules_proceds/operating/isone/op14/op14f_rto_final.pdf)**. This time value configuration may need to be changed depending on the Lead Market Participant's schedule. Regardless, a Lead Market Participant must have their power values in a comma separated list of 48 or 120 values. Below, we demonstrate one way to import power values from a csv, convert them into a comma separated list in Python, and pass them as input to the** [**submit\_forecast**](#_r4zprmwdr6fp) **function.**

**7.1. Save as a csv**

**7.1.1. Save an excel file containing WPFA/SPFA data as a csv in the same location (folder) as the Python project. Assuming the power values are in an Excel file like this:**

**A screenshot of a spreadsheet

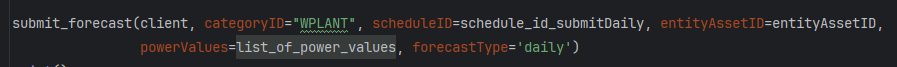
Description automatically generated**

**7.2. Read the csv file in Python and convert to a list:**

**A black background with white text

Description automatically generated**

**7.3. Use the list as input to** [**submit\_forecast**](#_r4zprmwdr6fp)**:**

****

**8.** **Debugging/ Error Handling**

**8.1. Setting up debugging**

**There are many reasons why and ways in which interacting with an API can fail. If there are unexpected results, failed queries, or blank responses:**

* **Go to the Properties.py file**
* **Set “debug = True”**
* **Set “xml = True”**
* **Run code again**
* **View response**

**8.2. Debugging output**

**Debug (red text) returns a lot of information, even if the query succeeds. Therefore, it is important to keep in mind that red does not mean error. To see if the query succeeded or failed, and if it failed, why it failed, scroll to the bottom of the output. There will be a status code. If the status is 200 then the query succeeded; otherwise, there will be more information about the error below the status code.**

**A screen shot of a computer program

Description automatically generated**

**8.3. XML output**

**XML is the language that the API communicates in. Therefore, to diagnose errors, it can sometimes be helpful to get the exact request and response that the API is receiving and giving. When viewing the XML in the output terminal, compare the results to the expected results described in the “ISO New England Wind Integration Data Exchange Specification.”**

**A screenshot of a computer program

Description automatically generated**

**9.** **Function Definitions**

**9.1.** **make\_request\_categories()**

**A computer screen shot of a program

Description automatically generated**

* **Purpose: Fetches a list of categories from WPF WebServices**
* **Parameters: None**
* **Response:**
  + **Type: Generally, a list or array of category objects that the Lead Market Participant can access. Example: A LMP with solar assets only would not have access to the wind plant category.**
  + **Contents: Each category object has an all capitalized identifier, a more detailed name, and a short description.**
  + **Usage: Use these details to display categories to users, or as filters for other queries in the system.**
  + **Example Response from QueryCategories:**

**A screen shot of a computer program

Description automatically generated**

**9.2.** **make\_request\_schedules()**

**A screen shot of a computer program

Description automatically generated**

* **Purpose: Retrieves a list of schedules**
* **Parameters: None**
* **Response:**
  + **Type: A collection of schedule data.**
  + **Contents: Includes schedule identifiers (unique integer identifiers for a specific schedule), abbreviated name of schedule type ('STWPFCST-MW' = ‘Short Term Wind Power Forecast’), short description of what the schedule is, and a Boolean (true/false) value representing if the schedule is read only or not.**
  + **Explanation: The concept of a "schedule" within the context of the Wind Integration Data Exchange API refers to a predefined or dynamically set timeframe during which specific operations or data exchanges regarding wind or solar generation are planned or executed. A schedule might delineate periods for submitting forecasts, retrieving data, or performing other tasks that are governed by time-based rules.**
  + **Usage: Useful for obtaining information on available schedules and the unique identifiers are required for making forecast requests and submissions.**
  + **Example Response from QuerySchedules (Wind Plant):**

**A black background with white lines

Description automatically generated**

**9.3.** **make\_request\_entities(cat=None)**

**A computer screen with colorful text

Description automatically generated**

* **Purpose: Queries entities, optionally filtering by a specific category**
* **Parameters:**
  + **This function takes an optional category parameter. To filter the entities by specific categories, put the string identifier for the category you would like to filter for. For example, if you only wanted to view solar plant entities you would call: make\_request\_entities(“SPLANT”).**
* **Response:**
  + **Type: A list or array of entity objects. An entity can be a physical unit like a turbine or a logical unit like a group of turbines that are managed as a single unit.**
  + **Contents: Entities include an identifier, asset identifier, name, description, and a Boolean (true/false) value representing if the value is read only or not.**
  + **Usage: This data can be used to manage or display entity-specific information or as part of broader operational workflows. The entity identifier is required in querying and submitting forecasts.**
  + **Example Response from QueryEntities(“WPLANT”):**

**A screen shot of a computer program

Description automatically generated**

**9.4.** **make\_request\_forecasts(entityID, entityAssetID, scheduleID)**

**A computer screen shot of a program code

Description automatically generated**

* **Purpose: Requests forecast data for a specific entity and schedule.**
* **Parameters:**
  + **entityAssetID: Asset identifier for the entity; this can be a more specific identifier within a larger entity.**
  + **scheduleID: Identifier of the schedule under which the forecast should be fetched.**
* **Response:**
  + **Type: Forecast data relevant to the specified entity and schedule.**
  + **Contents: Includes UTC formatted times and forecasted power output values at those times.**
  + **Usage: This data can be used for future planning, and to inform market decisions.**
  + **Example Response from QueryForecast:**

**A computer screen shot of white text

Description automatically generated**

**9.5. generate\_power\_entries(start\_time, num\_entries, power\_values)**

**A computer screen shot of a program code

Description automatically generated**

* **Purpose: Generates a list of dictionaries with time and power values. Each dictionary represents a power value prediction for a specific hour.**
* **Parameters:**
  + **start\_time: The starting time for the sequence of power entries.**
  + **num\_entries: The number of power entries to generate (48 or 120).**
  + **power\_values: A list of power values for each hour.**
* **Response:**
  + **Type: List of dictionaries.**
  + **Contents: Each entry is a dictionary with a time and a power value.**
  + **Usage: These entries are used to create detailed and time-specific forecasts. Note: this function is not intended for direct usage, but is designed to assist the submit\_forecast function.**
  + **Example snippet of data created by generate\_power\_entries():**

**[{'time': '2024-07-09T14:00:00+00:00', 'value': 8}, {'time': '2024-07-09T15:00:00+00:00', 'value': 8},...]**

**9.6. submit\_forecast(schedule\_id, entity\_id, entity\_asset\_id, power\_values, forecast\_type)**

**A screenshot of a computer program

Description automatically generated**

* **Purpose: Submits a forecast containing multiple power entries.**
* **Parameters:**
  + **schedule\_id: Schedule under which the forecast is being submitted.**
  + **entity\_id: Identifier for the entity associated with the forecast.**
  + **entity\_asset\_id: Asset identifier for a more specific reference within an entity.**
  + **power\_values: Power value to be used in generating power entries (how much power you are expected to produce at a given time).**
  + **forecast\_type: Either “daily” or “hourly”. Must correspond with the number of power values, 120 for “daily” 48 for “hourly”.**
* **Response:**
  + **Type: Success or failure message, returns a transaction id for a successfully submitted forecast.**
  + **Contents: Confirmation of the forecast submission or details of any errors encountered.**
  + **Usage: Feedback from this function can be used to confirm successful integration with the system or to troubleshoot problems in forecast submission.**
  + **Example Response from SubmitForecast:**

****