



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

0.1. Extract Files from the Zipped Project Folder 4

1. Configuring Certificates 4

1.1. Importing Client Certificate 4

1.2. Exporting CA Certificate	8
2. Setting up Python Integrated Development Environment (IDE) with PyCharm	13
2.1. Install PyCharm	13
2.2. Setting up a New Project	14
3. Configuring Properties	15
3.1. Navigate to the Properties.py Tab	15
3.2. ENDPOINT_URL	16
3.3. CA_CERT_PATH	16
3.4. CLIENT_CERT_PATH	16
3.5. CERT_PASSWORD	17
4. Converting and Combining Certificates	17
4.1. Navigate to the Unzipped Project Folder	17
5. Running the GUI and interacting with the API	18
5.1. Run the GUI	18
5.2. Fetch Categories	19
5.3. Select Category	20
5.4. Fetch Entities	21
5.5. Select Entity	22
5.6. Fetch Schedules	23
5.7. Select Schedule	24
5.8. Fetch Forecasts	26
5.9. Create and view CSV	27
5.10. Submit Forecast	30
6. Running Functions	31
6.1. Navigate to Calling_API.py tab	31
6.2. Calling make_request_categories()	32

6.3. Calling make_request_entities()	37
6.4. Calling make_request_schedules()	40
6.5. Calling make_request_forecasts()	48
6.6. Calling submit_forecast()	50

7. Getting Forecast Values 54

7.1. Save as a csv	55
7.2. Read the csv file in Python and convert to a list:	55
7.3. Use the list as input to submit_forecast:	55

8. Debugging/ Error Handling 55

8.1. Setting up debugging	55
8.2. Debugging output	56
8.3. XML output	56

9. Function Definitions 57

9.1. make_request_categories()	57
9.2. make_request_schedules()	58
9.3. make_request_entities(cat=None)	59
9.4. make_request_forecasts(entityID, entityAssetID, scheduleID)	60
9.5. generate_power_entries(start_time, num_entries, power_values)	61
9.6. submit_forecast(schedule_id, entity_id, entity_asset_id, power_values, forecast_type)	
	63

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...”

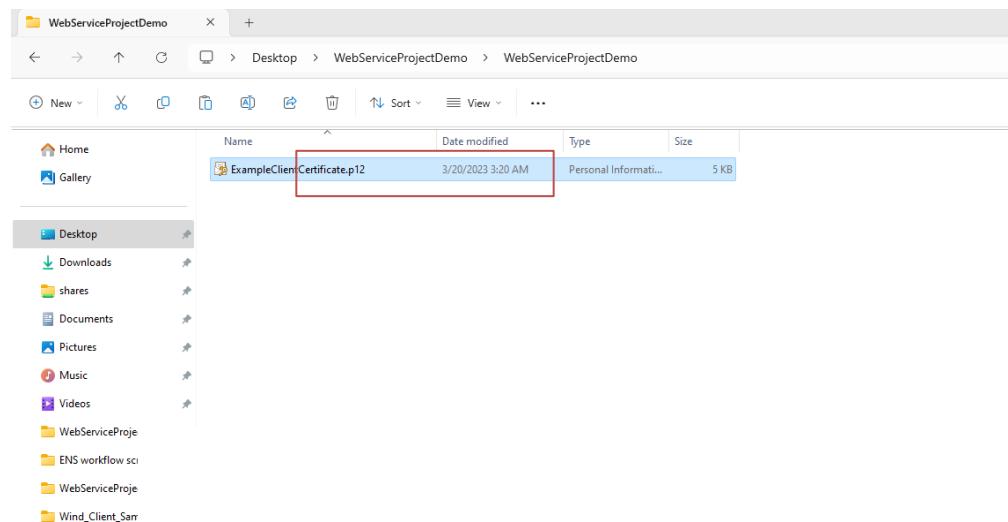
0.1.2. Keep track of this unzipped project folder; it will be used in future steps.

1. Configuring Certificates

1.1. Importing Client Certificate

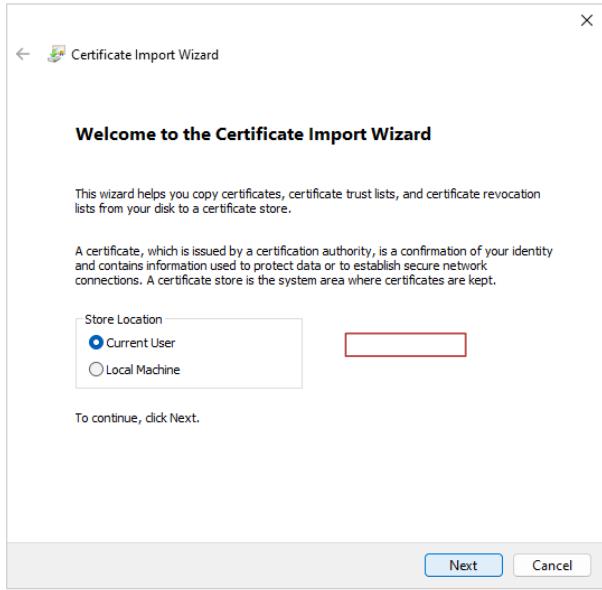
Refer to the instructions below, or follow along with this [video](#).

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



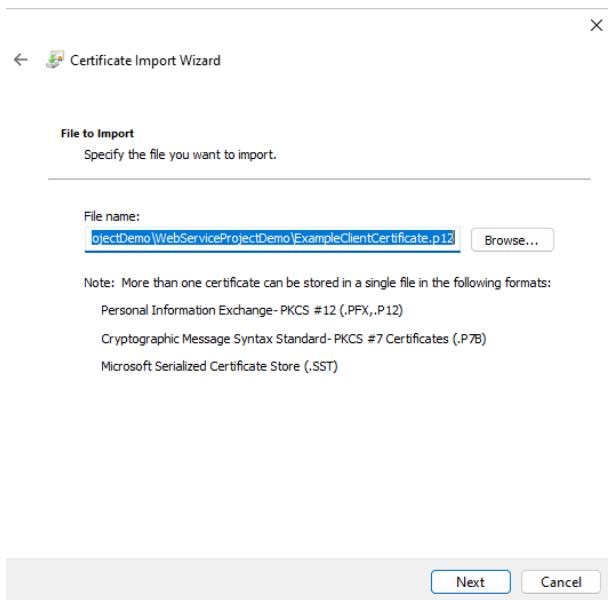
1.1.2. Store Location should be set to Current User.

1.1.2.1. Click Next.



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

1.1.3.1. Click Next.

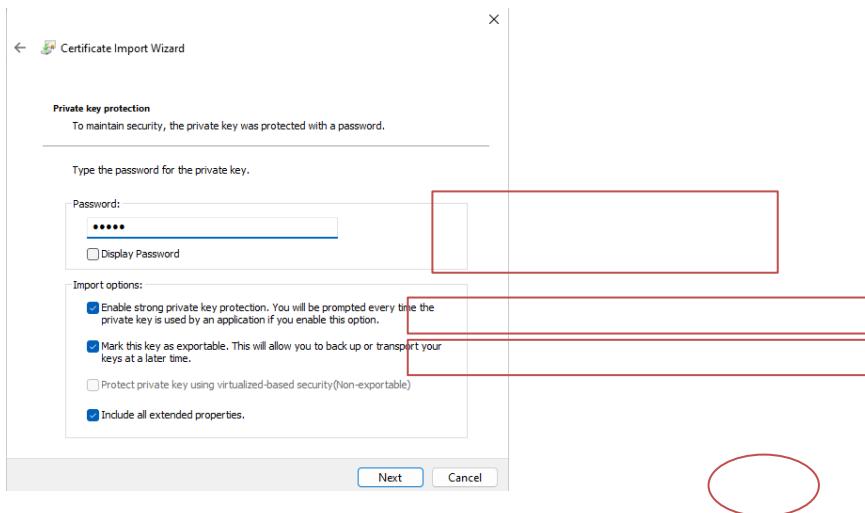


1.1.4. Input certificate password.

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.”

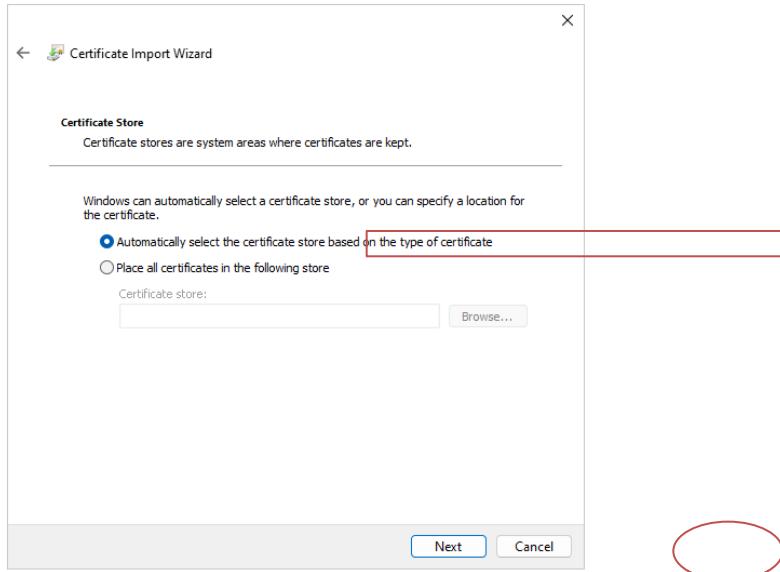
1.1.4.2. Click the box next to “Mark this key as exportable. This will allow you to back up or transport your keys at a later time.”

1.1.4.3. Click Next.

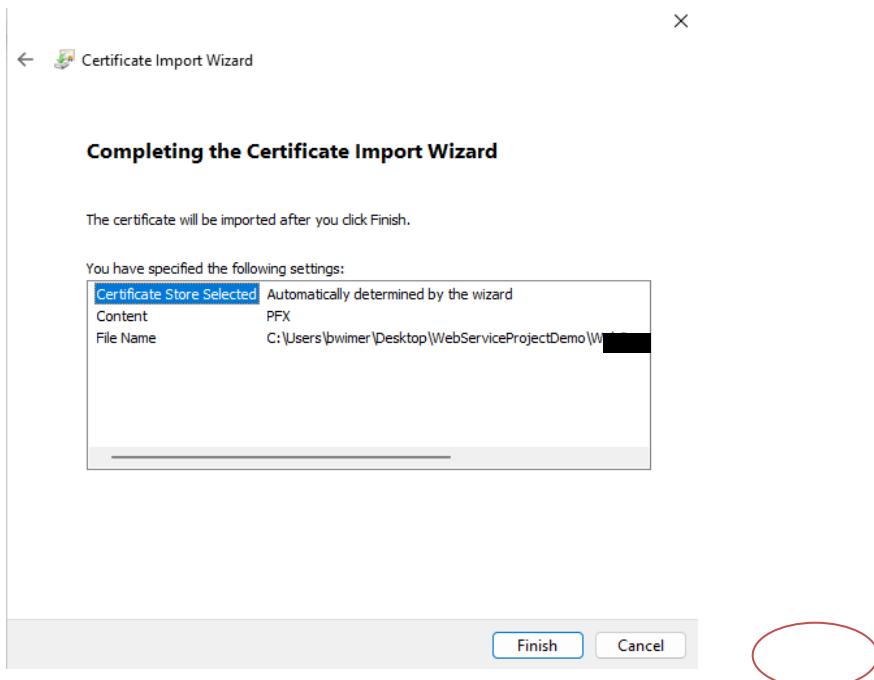


1.1.5. Select a specific keystore or allow automatic selection by selecting “Automatically select the certificate store based on the type of certificate.”

1.1.5.1. Click Next.

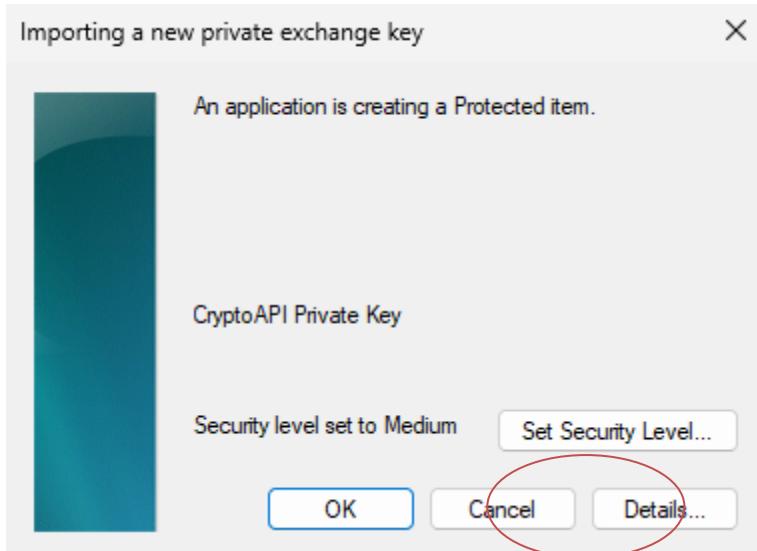


1.1.6. Click Finish.



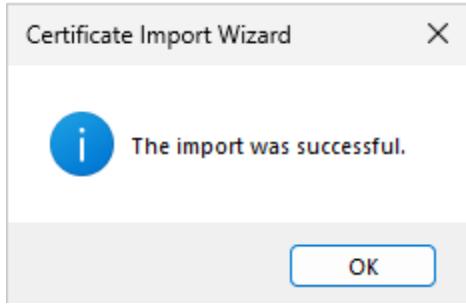
1.1.7. A pop-up should appear that looks like the figure below.

1.1.7.1. Click Ok.



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

1.1.8.1. Click Ok.



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).

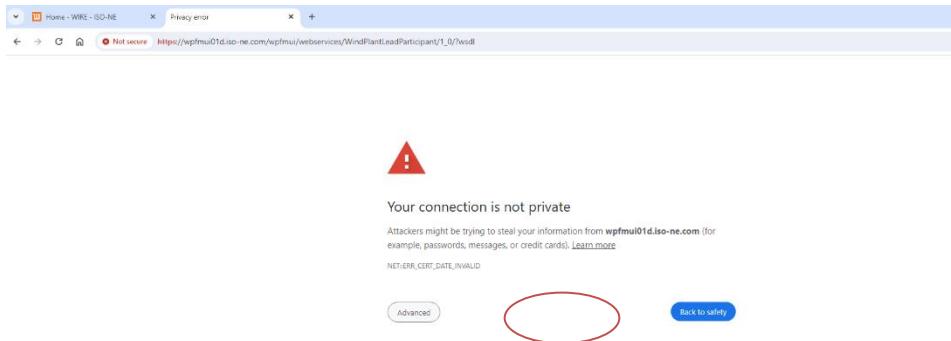
1.2.1.1. Use the **Sandbox** environment for initial setup and testing.

Sandbox url: https://sandboxesmd.iso-ne.com/wpfmui/webservices/WindPlantLeadParticipant/1_0/?wsdl

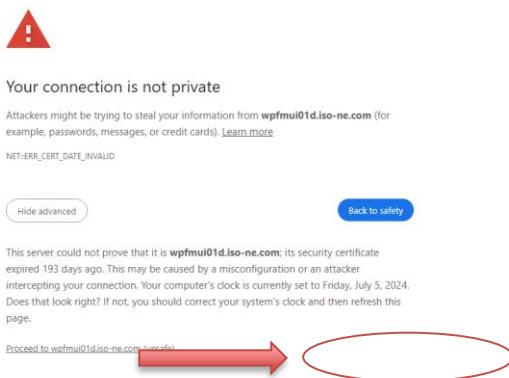
1.2.1.2. Use the **Production** environment when ready to start submitting WPFA/SPFA.

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

1.2.2. Click Advanced.



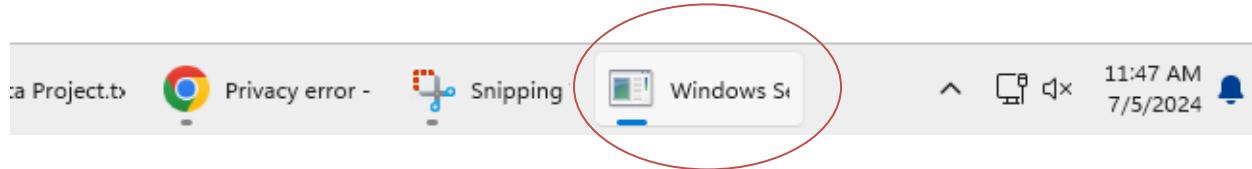
1.2.3. Click Proceed to wpfmu01d.iso-ne.com (unsafe).



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



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

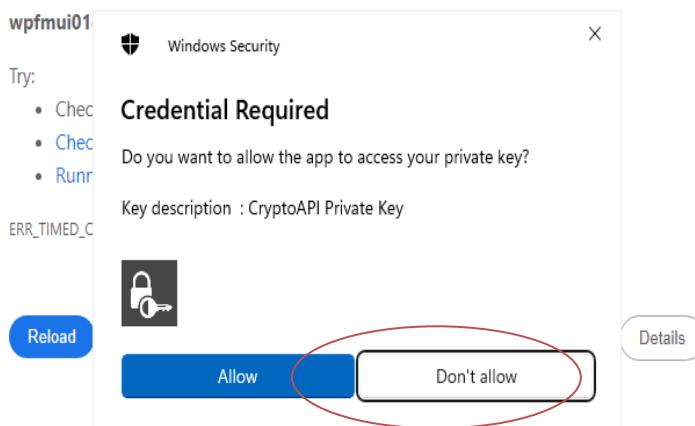


1.2.6. A pop-up should appear that looks like the figure below.

1.2.6.1. Click Allow.



This site can't be reached



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



This XML file is signed.

View site information

Not secure | https://wpfmu01d.iso-ne.com/wpfmu/webServices/WindPlantLeadParticipant/1_0/?wsdl

Your connection to this site is not secure
You should not enter any sensitive information on this site (for example, passwords or credit cards), because it could be stolen by attackers.
[Learn more](#)

You have chosen to turn off security warnings for this site. Turn on warnings

Certificate is not valid

Location Not allowed (default) Notifications Not allowed (default)
Pop-ups and redirects Allowed (default)

Use cookies and site data Site settings

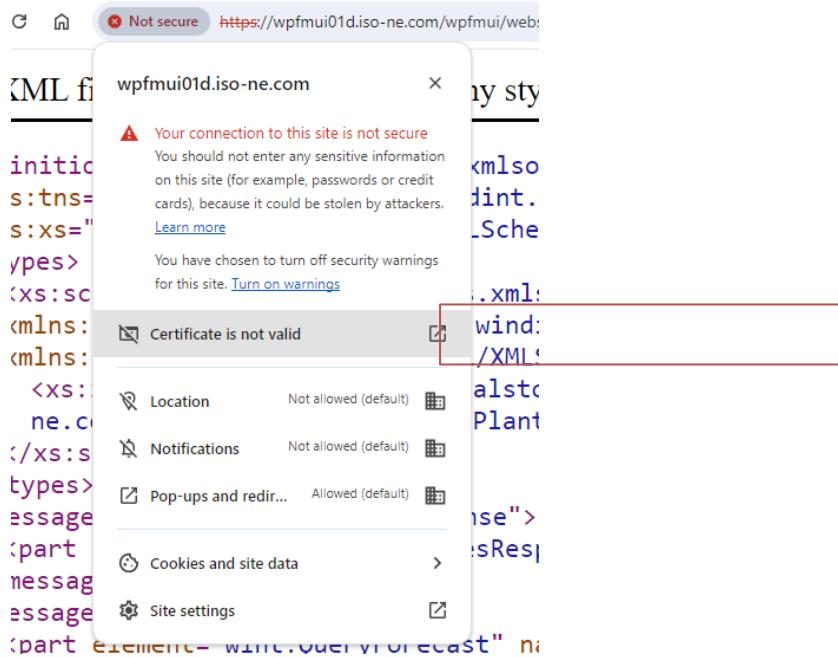
Any style information associated with it. The document tree is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:int_wsdl="urn:com.alstom.isone.windint.wsdl/XMLSchema" name="WindPlantLeadParticipantService" targetNamespace="http://alstom.isone.windint.wsdl/XMLSchema">
    <types>
        <xsd:sequence name="RPlan">
            <xsd:element name="forecast" type="xsd:string"/>
            <xsd:element name="parameters" type="xsd:string"/>
        </xsd:sequence>
        <xsd:complexType name="SubmitForecastRequest">
            <xsd:sequence>
                <xsd:element name="forecast" type="xsd:string"/>
                <xsd:element name="parameters" type="xsd:string"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="AuthorizationFaultMessage">
            <xsd:sequence>
                <xsd:element name="fault" type="xsd:string"/>
            </xsd:sequence>
        </xsd:complexType>
    </types>
    <message name="QueryForecast">
        <part element="wind:queryForecast" name="parameters"/>
    </message>
    <message name="SubmitForecast">
        <part element="wind:SubmitForecast" name="parameters"/>
    </message>
    <message name="AuthorizationFaultMessage">
        <part element="wind:AuthorizationFault" name="parameters"/>
    </message>

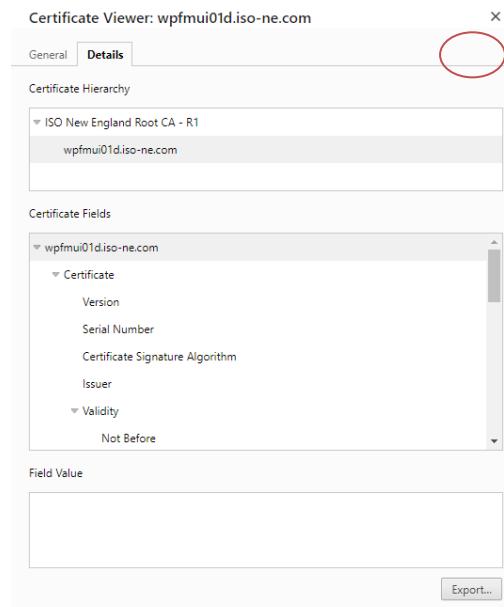
```

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.

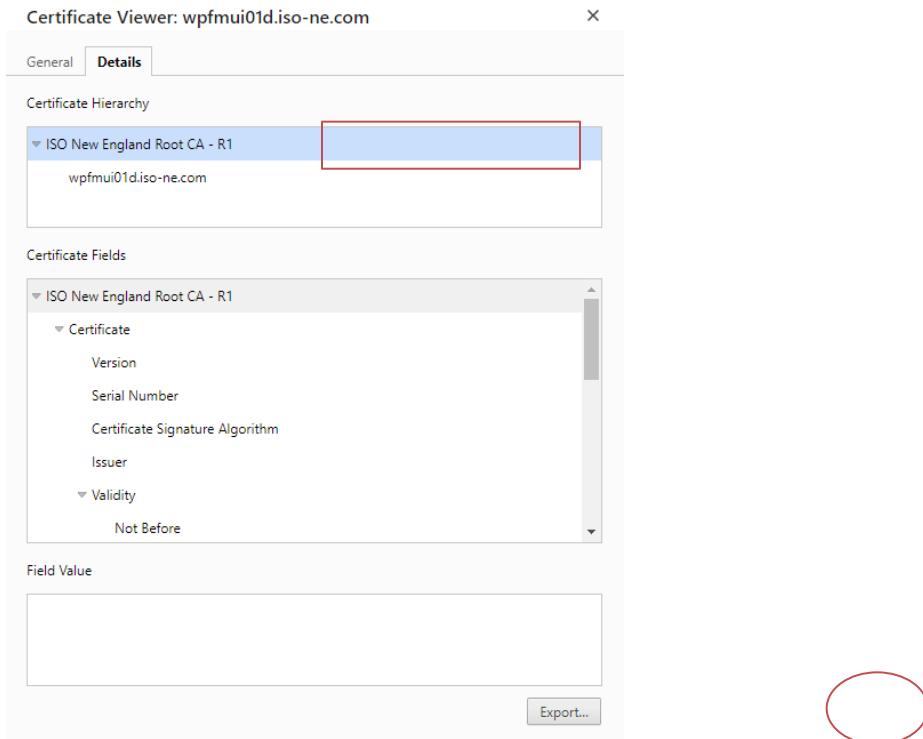


1.2.9. Go to the Details tab.



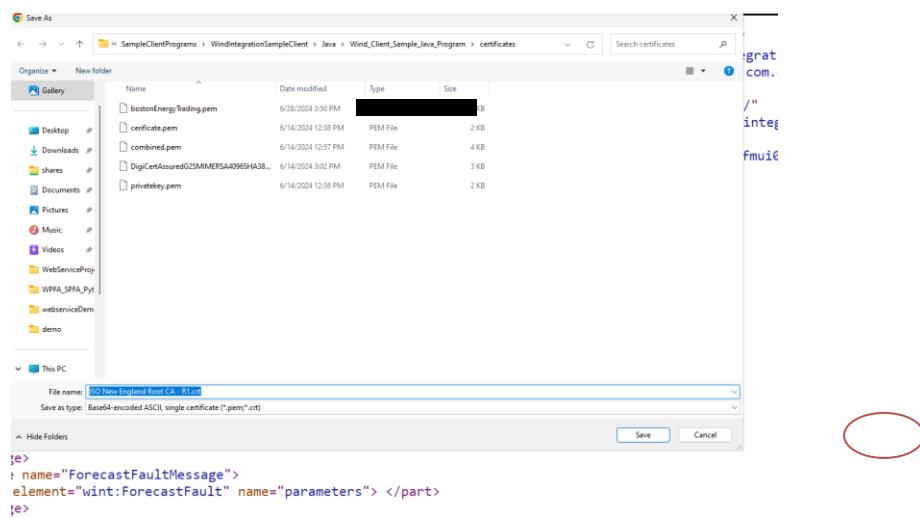
1.2.10. Click on the first certificate under “Certificate Hierarchy” and click “Export...”

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](#) 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>.

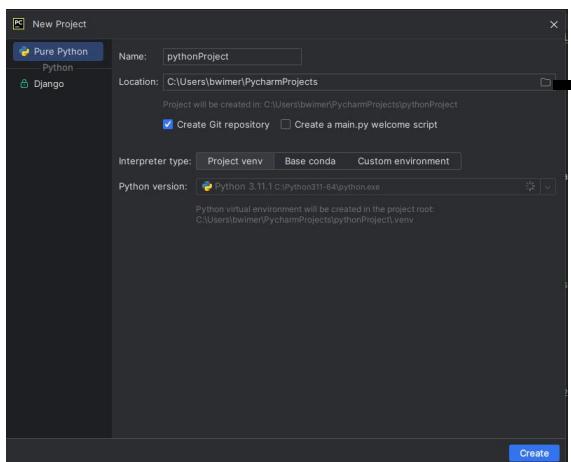
2.1.2. Follow the installation instructions.

2.2. Setting up a New Project

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

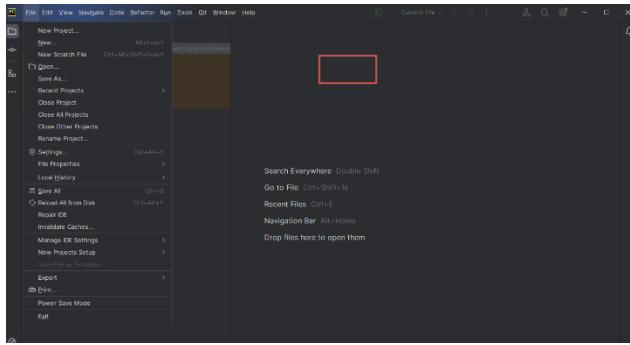
2.2.1. Open PyCharm and select “Create New Project.”

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

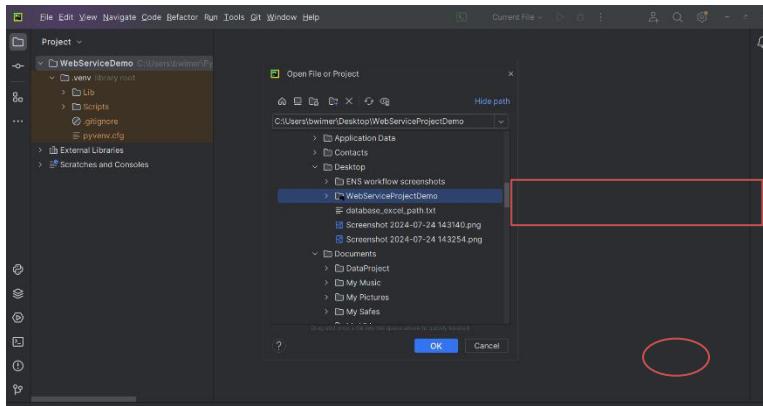


2.2.2 Select File and click Open.

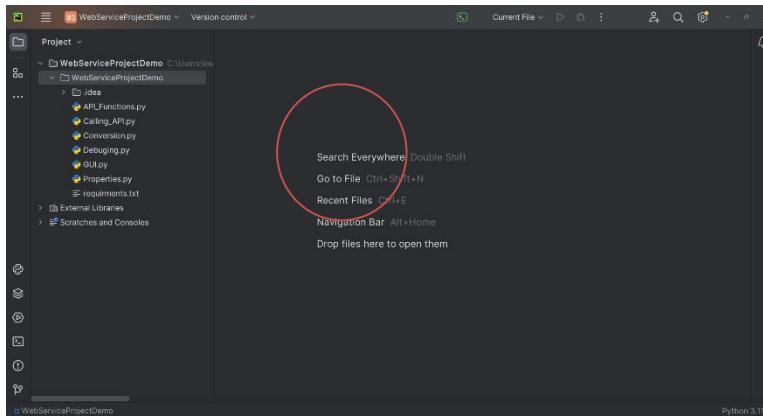




2.2.3. Navigate to the unzipped project folder and click OK.



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



3. Configuring Properties

3.1. Navigate to the Properties.py Tab



```

1 # Required constants
2 # url given by ISO NE
3 ENDPOINT_URL = "insert URL here"
4
5 # path to CA CERT on your computer
6 CA_CERT_PATH = r"C:\Certificates\Path\Here"
7
8 # path to client CERT on your computer (given by ISO NE)
9 CLIENT_CERT_PATH = r"Client\Certificate\Path\Here"
10
11 # private password for certificate
12 CERT_PASSWORD = "insert password here"
13
14 # path to the combined cert (this is a file you will be creating in the Conversion step)
15 COMBINED_CERT_PATH = r"New\Combined\Certificate\Path\Here"
16
17 # Set to True to display debugging
18 debug = False
19 # Set to True to display XML
20 XML = False

```

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://sandboxesmd.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.

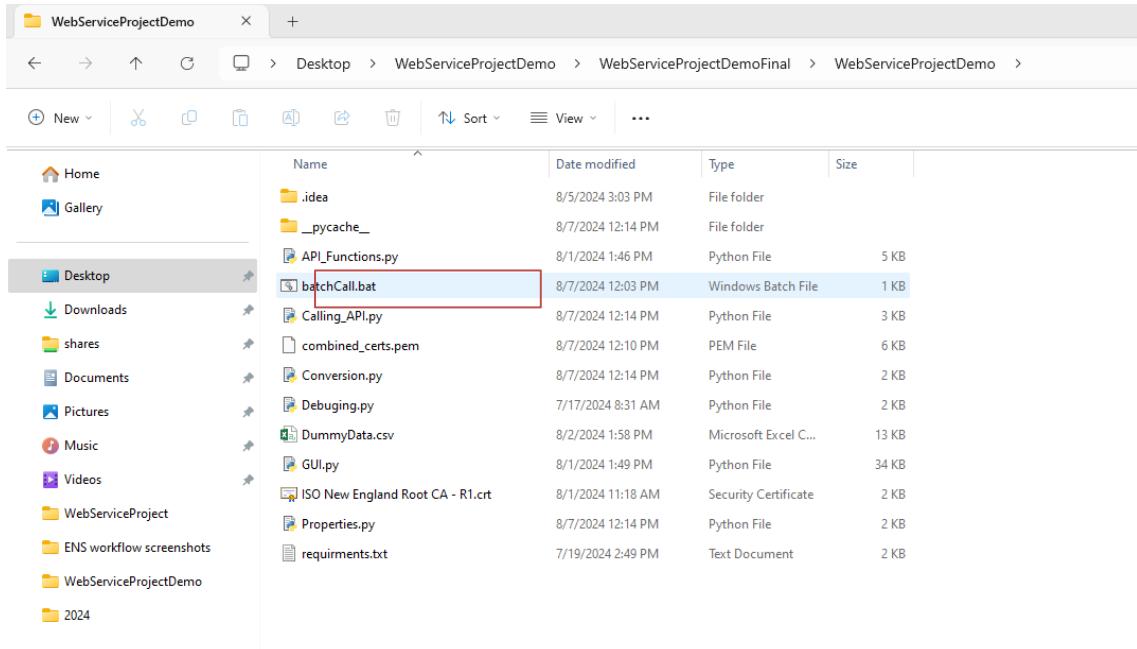
3.5.2. Ensure the password is surrounded by only one set of quotes with a letter r in front of it.

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.



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

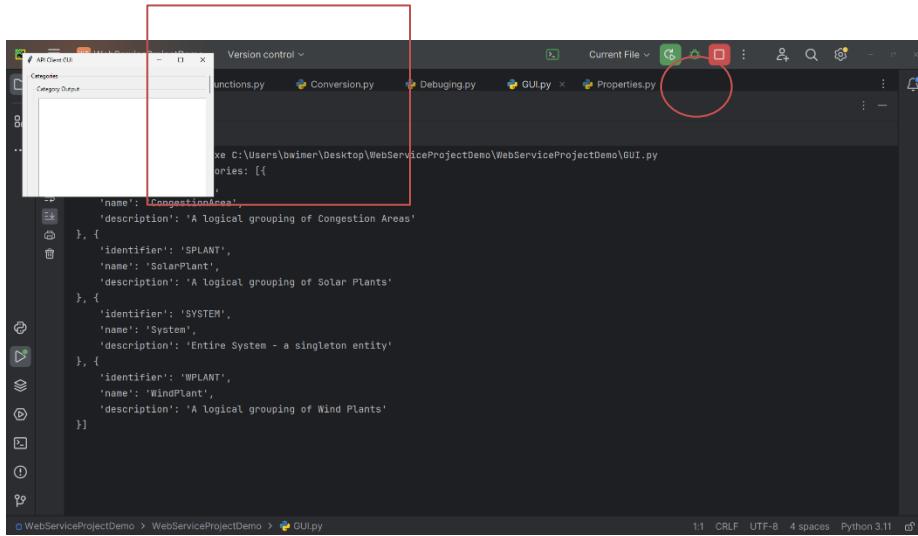
5.1.1 Navigate to the GUI.py tab.

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.

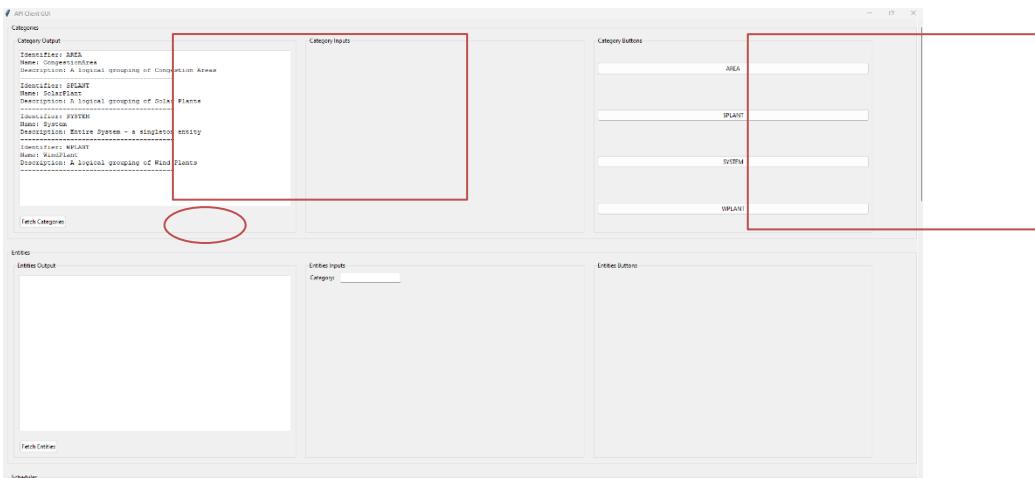
5.1.3. There should be a pop-up window like the one below; make it full screen.



5.2. Fetch Categories

5.2.1. Click the Fetch Categories button.

- This calls the [make_request_categories\(\)](#) 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.

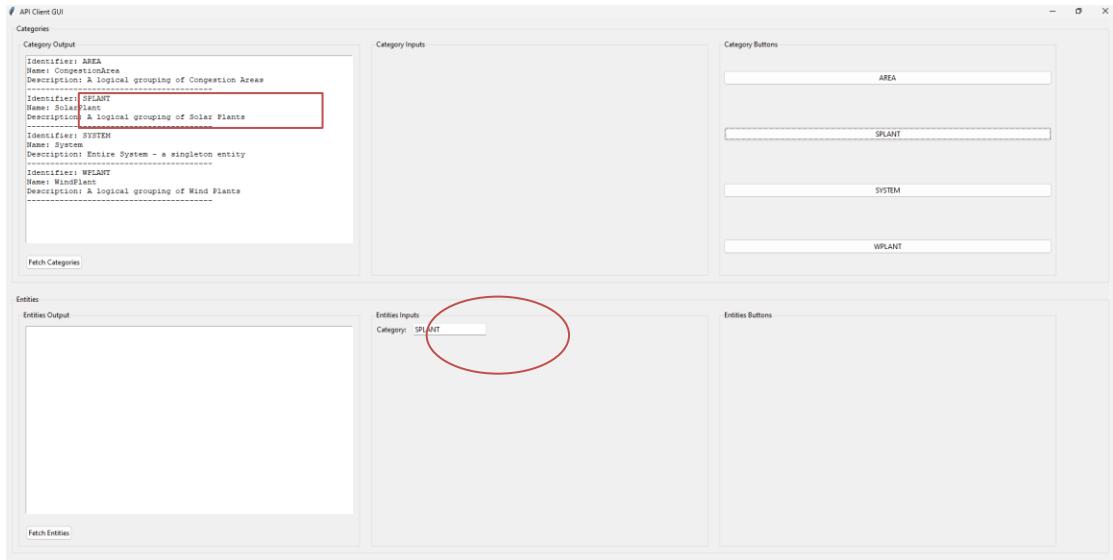


5.3. Select Category

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\(\)](#), [make_request_forecasts\(\)](#) and [submit_forecast\(\)](#) 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\(\)](#), [make_request_forecasts\(\)](#) and [submit_forecast\(\)](#) functions.

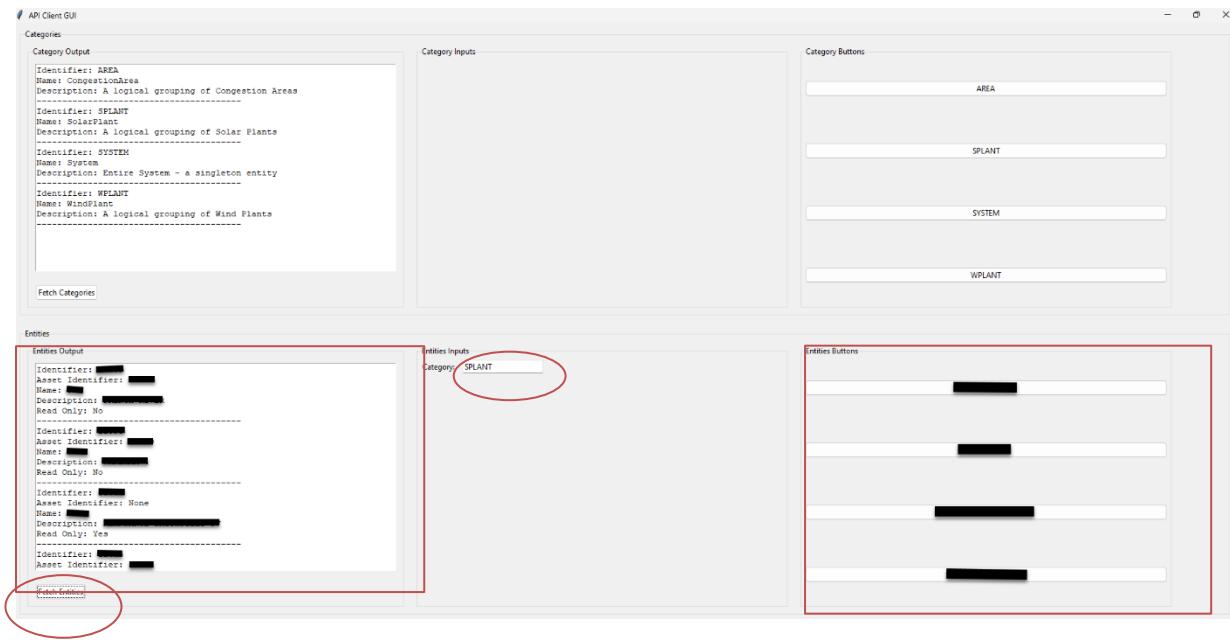


5.4. Fetch Entities

5.4.1. Click the Fetch Entities button.

- This calls the [make_request_entities\(\)](#) 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.



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\(\)](#) and [submit_forecast\(\)](#) functions.
- Note: Only the Entity's Asset Identifier will be used as input to the [make_request_forecasts\(\)](#) and [submit_forecast\(\)](#) functions; the Entity ID and Entity Name are only displayed for reference purposes.

The screenshot shows the Forecasting Application interface with two main panels: Schedules and Forecasts.

Schedules Panel:

- Schedules Output:** Displays a list of schedule entries with their details. One entry is highlighted:


```

Identifier: [REDACTED]
Name: STWPFCAST-MW
Description: Composite Short Term Wind Power Forecast computed by RPLA
M by blending STWPFCAST_SMMH and STWPFCAST_LMCHS forecasts. Power Generation
MW value.
-----
Identifier: [REDACTED]
Name: MNPFCST-MW
Description: Medium Term Wind Power Forecast. Forecast window being T
10hr to 48 hours. Power Generation MW value.
-----
Identifier: [REDACTED]
Name: LTPFCST-MW
Description: Long Term Wind Power Forecast. Forecast window being T+4
Shrs to 168 hours. Power Generation MW value.
-----
Identifier: [REDACTED]
Name: HMLWPFAL-MW
Description: Hourly Wind Plant Forecast Availability. Also known as f
uture hour RHMOL redetermination. Forecast window being T+1 hours to 48
      
```
- Schedules Inputs:** Contains fields for Entity ID, Entity Asset ID, Entity Name, Schedule ID, and Schedule Type.
- Schedules Buttons:** A vertical list of buttons for different types of forecasts and availability:
 - Composite Short Term Wind Plant Forecast
 - Medium Term Wind Power Forecast
 - Long Term Wind Power Forecast
 - Hourly Wind Plant Forecast Availability
 - Daily Wind Plant Forecast Availability
 - Composite Short-Term Forecast
 - Medium Term Solar Power Forecast
 - Long Term Solar Power Forecast
 - Hourly Solar Plant Forecast Availability
 - Daily Solar Plant Forecast Availability

Forecasts Panel:

- Forecasts Output:** Displays a list of forecast entries with their details. One entry is highlighted:


```

Identifier: [REDACTED]
Entity Asset ID: [REDACTED]
Entity Name: [REDACTED]
Schedule ID: [REDACTED]
Schedule Type: [REDACTED]
      
```
- Forecasts Inputs:** Contains fields for Entity ID, Entity Asset ID, Entity Name, Schedule ID, and Schedule Type.
- Forecasts Buttons:** A vertical list of buttons for different types of forecasts and availability:
 - Composite Short Term Wind Plant Forecast
 - Medium Term Wind Power Forecast
 - Long Term Wind Power Forecast
 - Hourly Wind Plant Forecast Availability
 - Daily Wind Plant Forecast Availability
 - Composite Short-Term Forecast
 - Medium Term Solar Power Forecast
 - Long Term Solar Power Forecast
 - Hourly Solar Plant Forecast Availability
 - Daily Solar Plant Forecast Availability

5.6. Fetch Schedules

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

- This calls the [make_request_schedules\(\)](#) 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.

Schedules

Schedules Output

- Identifier: [REDACTED]
Name: STWPFCAST-0HR
Description: Composite Short Term Wind Plant Forecast computed by RPLA N by blending STWPFCAST_SMIN and STWPFCAST_15MIN forecasts Power Generation MW value.
- Identifier: [REDACTED]
Name: MTWPFCAST-10H
Description: Medium Term Wind Power Forecast. Forecast window being T+1 hours to 48 hours. Power Generation MW value.
- Identifier: [REDACTED]
Name: LTWPFCAST-10H
Description: Long Term Wind Power Forecast. Forecast window being T+48 hours to 168 hours. Power Generation MW value.
- Identifier: [REDACTED]
Name: HRLYWFPA-10H
Description: Hourly Wind Plant Forecast Availability. Also known as future hour RTHOL redeclaration. Forecast window being T+1 hours to 48 hours.

Schedules Inputs

Schedules Buttons

- Composite Short Term Wind Plant Forecast
- Medium Term Wind Power Forecast
- Long Term Wind Power Forecast
- Hourly Wind Plant Forecast Availability
- Daily Wind Plant Forecast Availability
- Composite Short-Term Forecast
- Medium Term Solar Power Forecast
- Long Term Solar Power Forecast
- Hourly Solar Plant Forecast Availability
- Daily Solar Plant Forecast Availability

Forecasts

Forecasts Output

Forecasts Inputs

- Entity ID: [REDACTED]
- Entity Asset ID: [REDACTED]
- Entity Name: [REDACTED]
- Schedule ID: [REDACTED]
- Schedule Type: [REDACTED]

Forecasts Buttons

Fetch Schedules

Fetch Forecasts

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\(\)](#) function; the Schedule Type is only displayed for clarity.

The screenshot shows the software interface with three main panels:

- Schedules:** This panel displays three entries under "Schedules Output". Each entry includes an "Identifier" (e.g., SWPFCST-MW), a "Name" (e.g., SWPFCST-MW), a "Description" (e.g., "Composite Short Term Wind Plant Forecast computed by RPLA for the period between SWPFCST_MHIN and SWPFCST_LMHIN forecasts. Power Generation MW value."), and a "Type" (e.g., "Composite Short Term Wind Plant Forecast").
- Schedules Inputs:** This panel contains fields for "Entity ID", "Entity Asset ID", "Entry Name", "Schedule ID", and "Schedule Type". The "Schedule Type" dropdown is set to "DAILYWPFA-MW".
- Schedules Buttons:** This panel lists various forecast availability options: "Composite Short Term Wind Plant Forecast" (selected), "Medium Term Wind Power Forecast", "Long Term Wind Power Forecast", "Hourly Wind Plant Forecast Availability" (selected), "Daily Wind Plant Forecast Availability", "Composite Short-Term Forecast" (selected), "Medium Term Solar Power Forecast", "Long Term Solar Power Forecast", "Hourly Solar Plant Forecast Availability" (selected), and "Daily Solar Plant Forecast Availability".
- Forecasts:** This panel displays "Forecasts Output" with a table of time values from 2024-06-02 10:00:00 UTC to 2024-06-03 03:00:00 UTC. Below it is a "Forecasts Inputs" panel with fields for "Entity ID", "Entity Asset ID", "Entry Name", "Schedule ID", and "Schedule Type" (set to "DAILYWPFA-MW").

5.8. Fetch Forecasts

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

- This calls the [make_request_forecasts\(\)](#) 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.

Schedules

Schedules Output	
Identifier: [REDACTED]	Name: STWFCST-MW
Description: Composite Short Term Wind Plant Forecast computed by RPLA by blending JTWFCT_5MIN and JTWFCT_10MIN forecasts Power Generation MW values.	
Identifier: [REDACTED]	Name: MTWFCST-MW
Description: Medium Term Wind Power Forecast. Forecast window being T+1h to 48 hours. Power Generation MW value.	
Identifier: [REDACTED]	Name: LTWFCST-MW
Description: Long Term Wind Power Forecast. Forecast window being T+48 hours to 48 hours. Power Generation MW value.	
Identifier: [REDACTED]	Name: MSLWFTL-MW
Description: Hourly Wind Plant Forecast Availability. Also known as future hour RTHO redeclaration. Forecast window being T-1 hours to 48 hours.	

[Fetch Schedules](#)

Forecasts

Forecasts Output	
Time	value
2024-06-01 18:00:00 UTC	0.0
2024-06-01 19:00:00 UTC	0.0
2024-06-01 20:00:00 UTC	0.0
2024-06-01 21:00:00 UTC	0.0
2024-06-01 22:00:00 UTC	0.0
2024-06-01 23:00:00 UTC	0.0
2024-06-02 00:00:00 UTC	0.0
2024-06-02 01:00:00 UTC	0.0
2024-06-02 02:00:00 UTC	0.0
2024-06-02 03:00:00 UTC	0.0

[Fetch Forecasts](#)

Schedules Inputs

Composite Short Term Wind Plant Forecast
Medium Term Wind Power Forecast
Long Term Wind Power Forecast
Hourly Wind Plant Forecast Availability
Daily Wind Plant Forecast Availability
Composite Short-Term Forecast
Medium Term Solar Power Forecast
Long Term Solar Power Forecast
Hourly Solar Plant Forecast Availability
Daily Solar Plant Forecast Availability

Schedules Buttons

Composite Short Term Wind Plant Forecast
Medium Term Wind Power Forecast
Long Term Wind Power Forecast
Hourly Wind Plant Forecast Availability
Daily Wind Plant Forecast Availability
Composite Short-Term Forecast
Medium Term Solar Power Forecast
Long Term Solar Power Forecast
Hourly Solar Plant Forecast Availability
Daily Solar Plant Forecast Availability

5.9. Create and view CSV

5.9.1. Scroll down to the CSV Browser panel.

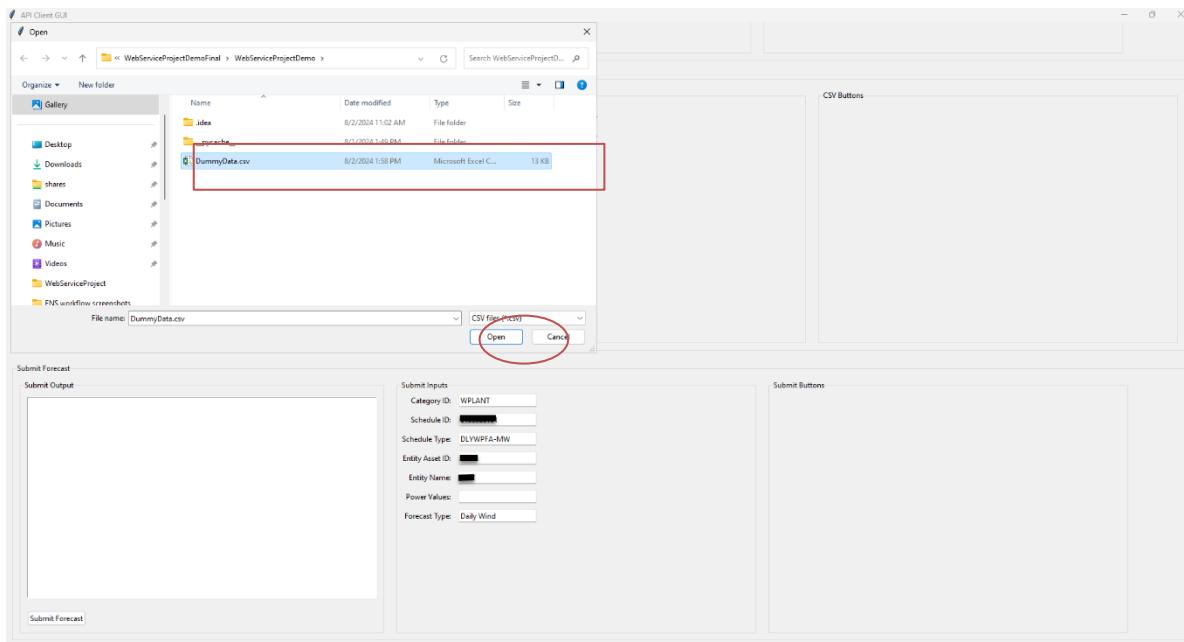
5.9.2. Click the Create CSV button in the CSV Output panel.

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

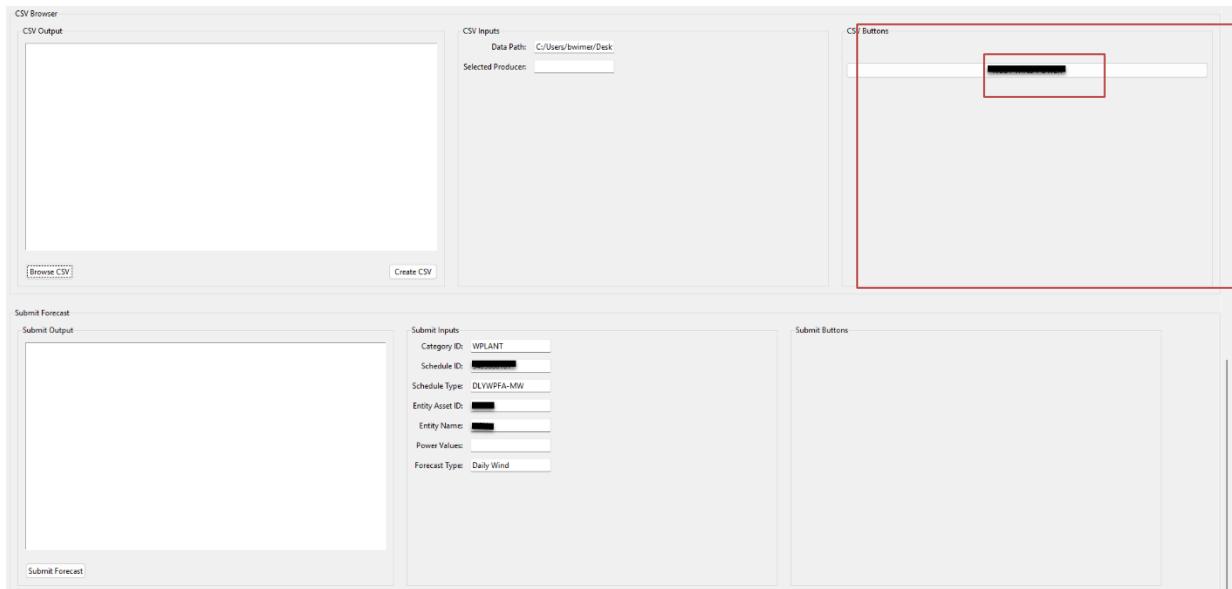
5.9.4. Select SampleData.csv

5.9.5. Click Open

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



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



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.

The screenshot shows the CSV Browser interface with several panels:

- CSV Output:** A table with columns "Time" and "Value". The "Time" column lists dates from 2024-06-03 07:00:00 to 2024-06-03 16:00:00. The "Value" column lists corresponding values: 49.0, 52.2, 35.0, 37.3, 66.1, 58.8, 30.5, 47.5, 69.9, and 37.4.
- CSV Inputs:** Data Path: C:/Users/bwimmer/Desktop. Selected Producer: [REDACTED]
- CSV Buttons:** A panel with a large button labeled [REDACTED].
- Submit Forecast:**
 - Submit Output:** A panel with fields: Category ID: WPLANT, Schedule ID: [REDACTED], Schedule Type: DLYWPPFA-MW, Entity Asset ID: [REDACTED], Entity Name: [REDACTED], Power Values: 49.0,52.2,35.0,37.3,66.1, Forecast Type: Daily Wind.
 - Submit Inputs:** A panel with fields: Category ID: WPLANT, Schedule ID: [REDACTED], Schedule Type: DLYWPPFA-MW, Entity Asset ID: [REDACTED], Entity Name: [REDACTED], Power Values: 49.0,52.2,35.0,37.3,66.1, Forecast Type: Daily Wind.
 - Submit Buttons:** A panel with a large button labeled [REDACTED].

5.10. Submit Forecast

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

- This calls the [submit_forecast\(\)](#) 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.

The screenshot displays a user interface for managing CSV files and submitting forecasts. It is divided into several sections:

- CSV Browser:** Shows a table of data with columns "Time" and "Value". The data includes rows from 2024-08-03 01:00:00 to 2024-08-03 16:00:00 with corresponding values like 49.0, 52.2, etc.
- CSV Inputs:** Displays "Data Path: C:/Users/bwimber/Desktop" and "Selected Producer: [REDACTED]".
- CSV Buttons:** A large button area with a progress bar.
- Submit Forecast:** This section contains two main areas:
 - Submit Output:** Shows a success message: "Success! Here is your transaction id: 27258094-6e9b-465f-92d9-aa34e5a13ab5b".
 - Submit Inputs:** Fields for "Category ID: WPLANT", "Schedule ID: [REDACTED]", "Schedule Type: DLWPPFA-MW", "Entity Asset ID: [REDACTED]", "Entity Name: [REDACTED]", "Power Values: 49.0,52.2,25.0,37.3,66.1," (with a red oval around it), and "Forecast Type: Daily Wind".
- Submit Buttons:** A large button area.

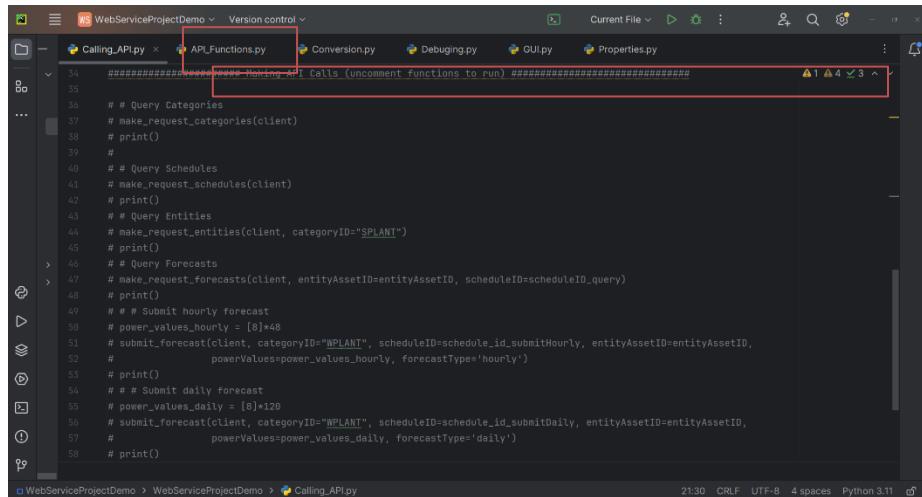
5.10.2. Exit out of the GUI window to stop it from running in the background for the remaining steps.

6. Running Functions

6.1. Navigate to Calling_API.py tab

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

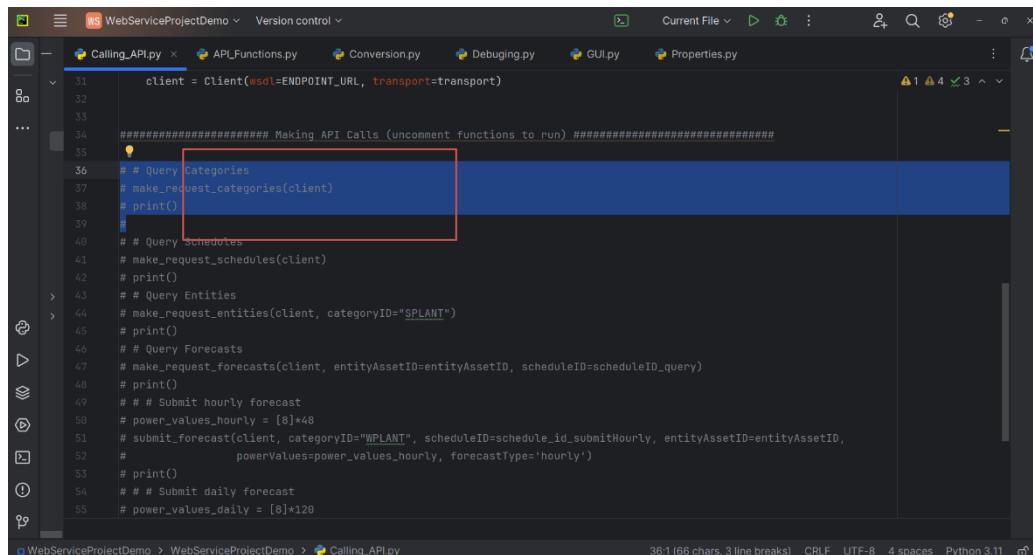
Note: The color of the text in this section will be gray.



```
# ##### Making API Calls (uncomment functions to run) #####
# # Query Categories
# make_request_categories(client)
# print()
# #
# # Query Schedules
# make_request_schedules(client)
# print()
# # Query Entities
# make_request_entities(client, categoryId="SPLANT")
# print()
# # Query Forecasts
# make_request_forecasts(client, entityAssetID=entityAssetID, scheduleID=scheduleID_query)
# print()
# # # Submit hourly forecast
# power_values_hourly = [8]*48
# submit_forecast(client, categoryId="WPLANT", scheduleID=schedule_id_submitHourly, entityAssetID=entityAssetID,
#                 powerValues=power_values_hourly, forecastType='hourly')
# print()
# # # Submit daily forecast
# power_values_daily = [8]*120
# submit_forecast(client, categoryId="WPLANT", scheduleID=schedule_id_submitDaily, entityAssetID=entityAssetID,
#                 powerValues=power_values_daily, forecastType='daily')
# print()
```

6.2. Calling [make_request_categories\(\)](#)

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



```
client = Client(wsdl=ENDPOINT_URL, transport=transport)

##### Making API Calls (uncomment functions to run) #####
# # Query Categories
# make_request_categories(client)
# print()

# #
# # Query Schedules
# make_request_schedules(client)
# print()
# # Query Entities
# make_request_entities(client, categoryId="SPLANT")
# print()
# # Query Forecasts
# make_request_forecasts(client, entityAssetID=entityAssetID, scheduleID=scheduleID_query)
# print()
# # # Submit hourly forecast
# power_values_hourly = [8]*48
# submit_forecast(client, categoryId="WPLANT", scheduleID=schedule_id_submitHourly, entityAssetID=entityAssetID,
#                 powerValues=power_values_hourly, forecastType='hourly')
# print()
# # # Submit daily forecast
# power_values_daily = [8]*120
```

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

```
client = Client(wsdl=ENDPOINT_URL, transport=transport)

##### Making API Calls (uncomment functions to run) #####
# Query Categories
make_request_categories(client)
print()

# # Query Schedules
# make_request_schedules(client)
# print()
# # Query Entities
# make_request_entities(client, categoryID="SPLANT")
# print()
# # Query Forecasts
# make_request_forecasts(client, entityAssetID=entityAssetID, scheduleID=scheduleID_query)
# print()
# # Submit hourly forecast
# power_values_hourly = [8]*48
# submit_forecast(client, categoryID="#PLANT", scheduleID=schedule_id_submitHourly, entityAssetID=entityAssetID,
# powerValues=power_values_hourly, forecastType='hourly')
# print()
# # Submit daily forecast
# power_values_daily = [8]*120
```

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.

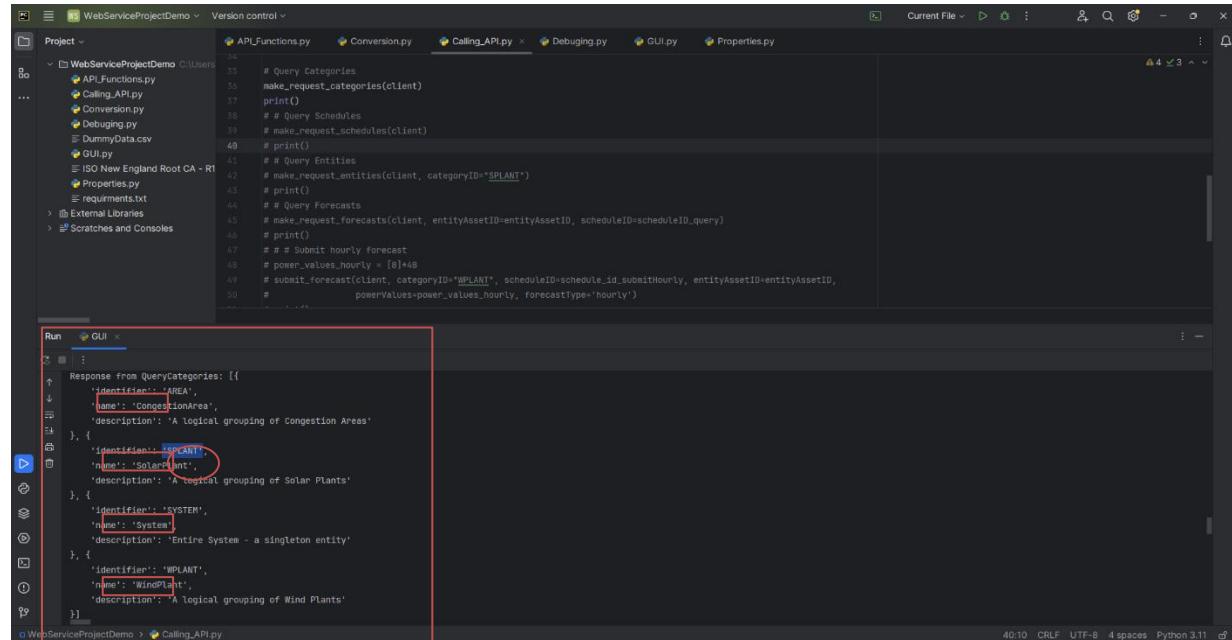


6.2.5. Select an identifier from the function output.

6.2.5.1. Highlight the selected identifier.

Example: SPLANT for solar plants.

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



The screenshot shows the PyCharm IDE interface. The top navigation bar includes 'Project', 'File', 'Edit', 'View', 'Run', 'Current File', 'File', 'Search', and 'Help'. The 'Calling_API.py' tab is active. The code editor displays Python code for querying categories and schedules, with a specific line highlighted:

```
# make_request_entities(client, categoryId='SPLANT')
```

The bottom panel shows the 'Run' tab with a JSON response from 'QueryCategories'. A red box highlights the 'SPLANT' identifier in the first category object:

```
[{"identifier": "AREA", "name": "CongestionArea", "description": "A logical grouping of Congestion Areas"}, {"identifier": "SPLANT", "name": "SolarPlant", "description": "A logical grouping of Solar Plants"}, {"identifier": "SYSTEM", "name": "System", "description": "Entire System - a singleton entity"}, {"identifier": "WPLANT", "name": "WindPlant", "description": "A logical grouping of Wind Plants"}]
```

At the bottom right of the interface, status information includes '40:10 CRLF - UTF-8 - 4 spaces - Python 3.11 - m'.

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.

The screenshot shows a code editor interface with a Python file named `Calling_API.py` open. The code is part of a project named `WebServiceProjectDemo`. A red rectangular box highlights a specific section of the code, which is annotated with `##### Assigning Variables #####`. This section contains variable assignments:

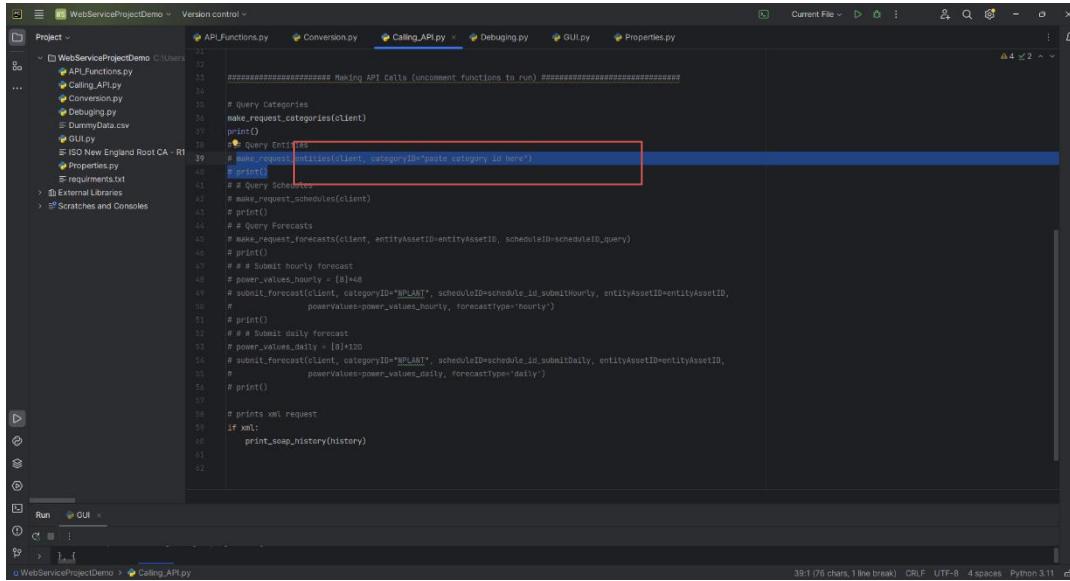
```
categoryID = "SPLANT"
entityAssetID = "paste your Entity Asset ID here"
scheduleID_query = "paste your schedule ID for querying here"
scheduleID_submitHourly = "paste your schedule ID for submitting hourly forecast here"
scheduleID_submitDaily = "paste your schedule ID for submitting daily forecast here"
```

Below this, another section is annotated with `##### Making API Calls (uncomment functions to run) #####`. This section contains several commented-out function calls:

```
# Query Categories
# make_request_categories(client)
# print()
# # Query Entities
# make_request_entities(client, categoryID=categoryID)
# print()
# # Query Schedules
# make_request_schedules(client)
# print()
# # Query Forecasts
# make_request_forecasts(client, entityAssetID=entityAssetID, scheduleID=scheduleID_query)
# print()
# # Submit hourly forecast
# power_values_hourly = [8]*48
# submit_forecast(client, categoryID=categoryID, scheduleID=scheduleID.submitHourly, entityAssetID=entityAssetID,
#                 powerValues=power_values_hourly, forecastType='hourly')
# print()
# # Submit daily forecast
# power_values_daily = [8]*120
# submit_forecast(client, categoryID="SPLANT", scheduleID=scheduleID.submitDaily, entityAssetID=entityAssetID,
```

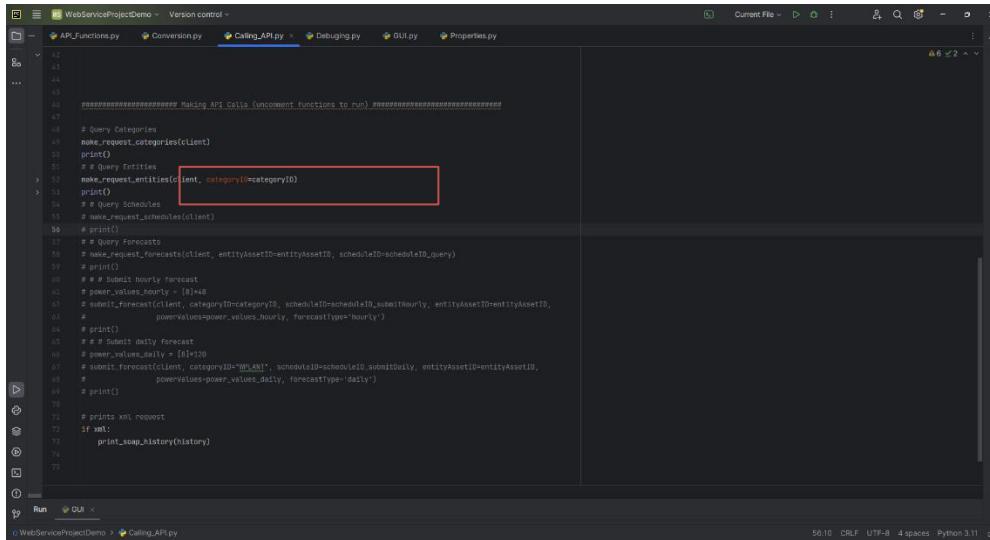
6.3. Calling `make_request_entities()`

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



```
31 ##### Making API calls (uncomment functions to run) #####
32
33 # Query Categories
34 make_request_categories(client)
35 print()
36
37 # # Query Entities
38 # make_request_entities(client, categoryID='put category id here')
39 # print()
40 # # Query Schedules
41 # make_request_schedules(client)
42 # print()
43
44 # # Query Forecasts
45 # make_request_forecasts(client, entityAssetID=entityAssetID, scheduleID=scheduleID_query)
46 # print()
47
48 # # Submit hourly forecast
49 # submit_forecast(client, categoryID='WPLANT', scheduleID=schedule_id_submitHourly, entityAssetID=entityAssetID,
50 # powerValues=power_values_hourly, forecastType='hourly')
51 # print()
52
53 # # Submit daily forecast
54 # power_values_daily = [8]*120
55 # submit_forecast(client, categoryID='WPLANT', scheduleID=schedule_id_submitDaily, entityAssetID=entityAssetID,
56 # powerValues=power_values_daily, forecastType='daily')
57 # print()
58
59 # prints XML request
60 if XML:
61     print_soap_history(history)
62
```

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



```
42
43
44 ##### Making API Calls (uncomment functions to run) #####
45
46 # Query Categories
47 make_request_categories(client)
48 print()
49
50 # # Query Entities
51 # make_request_entities(client, categoryID='put category id here')
52 # print()
53 # # Query Schedules
54 # make_request_schedules(client)
55 # print()
56
57 # # Query Forecasts
58 # make_request_forecasts(client, entityAssetID=entityAssetID, scheduleID=scheduleID_query)
59 # print()
60
61 # # Submit hourly forecast
62 # power_values_hourly = [8]*48
63 # submit_forecast(client, categoryID='WPLANT', scheduleID=schedule_id_submitHourly, entityAssetID=entityAssetID,
64 # powerValues=power_values_hourly, forecastType='hourly')
65 # print()
66
67 # # Submit daily forecast
68 # power_values_daily = [8]*120
69 # submit_forecast(client, categoryID='WPLANT', scheduleID=schedule_id_submitDaily, entityAssetID=entityAssetID,
70 # powerValues=power_values_daily, forecastType='daily')
71 # print()
72
73 # prints XML request
74 if XML:
75     print_soap_history(history)
76
```

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.

```

Response from QueryEntities: {
    "Category": {
        "identifier": "SPLANT",
        "name": None,
        "description": None
    },
    "entities": [
        {
            "Entity": [
                {
                    "identifier": 1111111,
                    "assetIdentifier": "1111111",
                    "name": "Example Plant Name",
                    "description": "Example Plant Description",
                    "isReadonly": False
                },
                {
                    "identifier": 1111111,
                    "assetIdentifier": "1111111",
                    "name": "Example Plant Name",
                    "description": "Example Plant Description",
                    "isReadonly": False
                },
                {
                    "identifier": 1111111,
                    "assetIdentifier": 1111111,
                    "name": "Example Plant Name",
                    "description": "Example Plant Description",
                    "isReadonly": True
                },
                {
                    "identifier": 1111111,
                    "assetIdentifier": 1111111,
                    "name": "Example Plant Name",
                    "description": "Example Plant Description",
                    "isReadonly": False
                }
            ]
        }
    ]
}

```

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.

```

#####
# ***** Assigning Variables *****
#####

categoryID = "SPLANT"
entityAssetID = "1111111"
scheduleID_query = "paste your schedule ID for querying here"
scheduleID_submitHourly = "paste your schedule ID for submitting hourly forecast here"
scheduleID_submitDaily = "paste your schedule ID for submitting daily forecast here"

#####
# ***** Making API Calls (uncomment functions to run) *****
#####

# # Query Categories
make_request_categories(client)
print()

# # Query Entities
make_request_entities(client, categoryID=categoryID)
print()

# # Query Schedules
# make_request_schedules(client)
# print()
# # Query Forecasts
# make_request_forecasts(client, entityAssetID=entityAssetID, scheduleID=scheduleID_query)
# print()
# ## Submit hourly forecast
# power_values_hourly = [8]*48
# submit_forecast(client, categoryID=categoryID, scheduleID=scheduleID_submitHourly, entityAssetID=entityAssetID,
#                 power_values=power_values_hourly, forecastType='hourly')
# print()
# ## Submit daily forecast
# power_values_daily = [8]*120
# submit_forecast(client, categoryID=categoryID, scheduleID=scheduleID_submitDaily, entityAssetID=entityAssetID,
#                 power_values=power_values_daily, forecastType='daily')


```

6.4. Calling [make_request_schedules\(\)](#)

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

```
WebServiceProjectDemo > Version control >
API_Functions.py Conversion.py Calling_API.py Debugging.py GUI.py Properties.py
...
89 scheduleID_submitDaily = "paste your schedule ID for submitting daily forecast here"
90
91
92
93
94
95
96 ##### Making API Calls (uncomment functions to run) #####
97
98 # Query Categories
99 make_request_categories(client)
100 print()
101 # Query Entities
102 make_request_entities(client, categoryId=categoryID)
103 print()
104 # # Query Services
105 # make_request_schedules(client)
106 # print()
107 # # Query Forecasts
108 # make_request_forecasts(client, entityAssetID=entityAssetID, scheduleID=scheduleID_query)
109 # print()
110 # # Submit hourly forecast
111 # power_values_hourly = [0]*48
112 # submit_forecast(client, categoryId=categoryID, scheduleID=scheduleID_submitHourly, entityAssetID=entityAssetID,
113 # powerValues=power_values_hourly, forecastType='hourly')
114 # print()
115 # # Submit daily forecast
116 # power_values_dally = [0]*120
117 # submit_forecast(client, categoryId="WPLANT", scheduleID=scheduleID_submitDaily, entityAssetID=entityAssetID,
118 # powerValues=power_values_dally, forecastType='daily')
119 # print()
120
121 # prints xml request
122 if xml:
123     print_soap_history(history)
124
125 Run GUI <
Calling_API.py
55:1 (42 chars, 1 line break) CRLF UTF-8 4 spaces Python 3.11
```

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

```
WebServiceProjectDemo > Version control >
API_Functions.py Conversion.py Calling_API.py Debugging.py GUI.py Properties.py
...
88 scheduleID_submitDaily = "paste your schedule ID for submitting daily forecast here"
89
90
91
92
93
94
95
96 ##### Making API Calls (uncomment functions to run) #####
97
98 # Query Categories
99 make_request_categories(client)
100 print()
101 # Query Entities
102 make_request_entities(client, categoryId=categoryID)
103 print()
104 # # Query Services
105 # make_request_schedules(client)
106 # print()
107 # # Query Forecasts
108 # make_request_forecasts(client, entityAssetID=entityAssetID, scheduleID=scheduleID_query)
109 # print()
110 # # Submit hourly forecast
111 # power_values_hourly = [0]*48
112 # submit_forecast(client, categoryId=categoryID, scheduleID=scheduleID_submitHourly, entityAssetID=entityAssetID,
113 # powerValues=power_values_hourly, forecastType='hourly')
114 # print()
115 # # Submit daily forecast
116 # power_values_dally = [0]*120
117 # submit_forecast(client, categoryId="WPLANT", scheduleID=scheduleID_submitDaily, entityAssetID=entityAssetID,
118 # powerValues=power_values_dally, forecastType='daily')
119 # print()
120
121 # prints xml request
122 if xml:
123     print_soap_history(history)
124
125 Run GUI <
Calling_API.py
68:72 CRLF UTF-8 4 spaces Python 3.11
```

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.

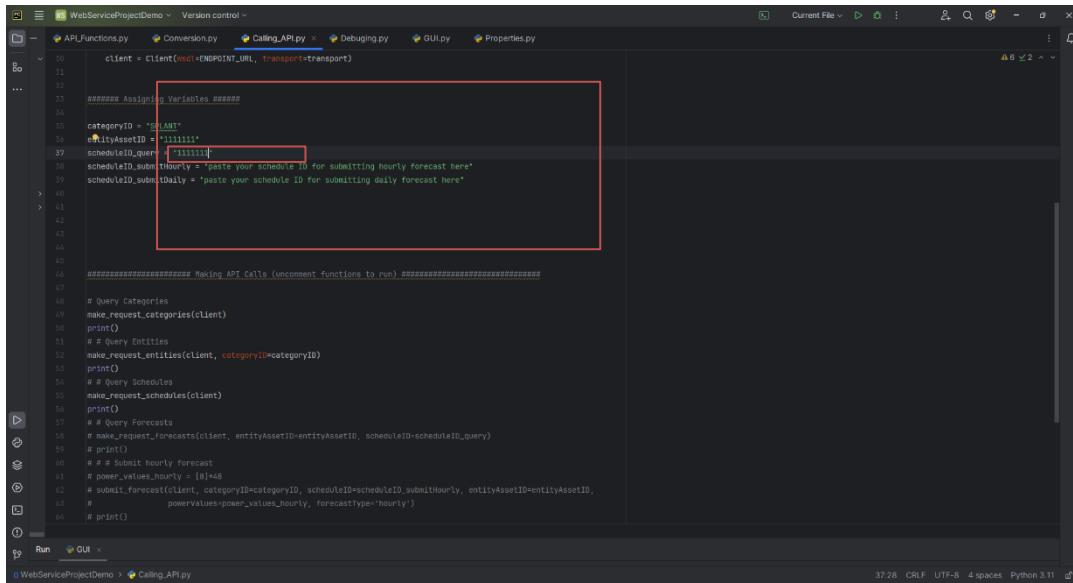
The screenshot shows a Python code editor with a file named `Calling_API.py` open. The code lists several forecast schedules as a JSON-like structure:

```
Response from QuerySchedules: [
    {
        "identifier": "1111111",
        "name": "STMPFCST-MW",
        "description": "Composite short term Wind Plant Forecast Computed by RPLAN by blending STWPFCST_5MIN and STWPFCST_15MIN forecasts Power Generation MW value.",
        "isReadOnly": True
    },
    {
        "identifier": "1111111",
        "name": "M1MPFCST-MW",
        "description": "Medium Term Wind Power Forecast. Forecast window being T+1hr to 48 hours. Power Generation MW value.",
        "isReadOnly": False
    },
    {
        "identifier": "1111111",
        "name": "LTMForecast-MW",
        "description": "Long term Wind Power Forecast. Forecast window being T+48hrs to 168 hours. Power Generation MW value.",
        "isReadOnly": True
    },
    {
        "identifier": "11111111",
        "name": "HRLYWPFA-MW",
        "description": "Hourly Wind Plant Forecast Availability. Also known as future hour RTHOL redeclaration. Forecast window being T+1 hours to 48 hours. Power Generation MW value.",
        "isReadOnly": False
    },
    {
        "identifier": "11111111",
        "name": "ULYWPFA-MW",
        "description": "Daily Wind Plant Forecast Availability. Also known as future hour RTHOL redeclaration. Forecast Window being T+48hrs to 7 days. Power Generation MW value.",
        "isReadOnly": False
    },
    {
        "identifier": "11111111",
        "name": "STFCST-S-MW",
        "description": "Composite Short-Term Forecast computed by blending STFCST_5MIN_S and STFCST_15MIN_S forecasts. Power Generation MW value.",
        "isReadOnly": True
    },
    {
        "identifier": "11111111",
        "name": "MTFCST-S-MW",
        "description": "Medium Term Solar Power Forecast. Forecast window being T+1hr to 48 hours. Power Generation MW value."
}
```

The code editor interface includes tabs for other files like `Hidden_Properties.py`, `Properties.py`, `Conversion.py`, `Debugging.py`, `API_Functions.py`, and `GUI.py`. The status bar at the bottom shows file statistics: 46.8 CRLF, UTF-8, 4 spaces, Python 3.11 (WebServiceProject).

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.



```
client = Client(url=ENDPOINT_URL, transport=transport)

#####
##### Assigning Variables #####
#####

categoryID = "1A1NT"
entityAssetID = "1111111"
scheduleID_query = "1111111"
scheduleID_submitHourly = "paste your schedule ID for submitting hourly forecast here"
scheduleID_submitDaily = "paste your schedule ID for submitting daily forecast here"

#####
##### Making API Calls (uncomment functions to run) #####
#####

# Query Categories
make_request_categories(client)
print()

## Query Entities
make_request_entities(client, categoryID=categoryID)
print()

## Query Schedules
make_request_schedules(client)
print()

## Query Forecasts
# make_request_forecasts(client, entityAssetID=entityAssetID, scheduleID=scheduleID_query)
# print()

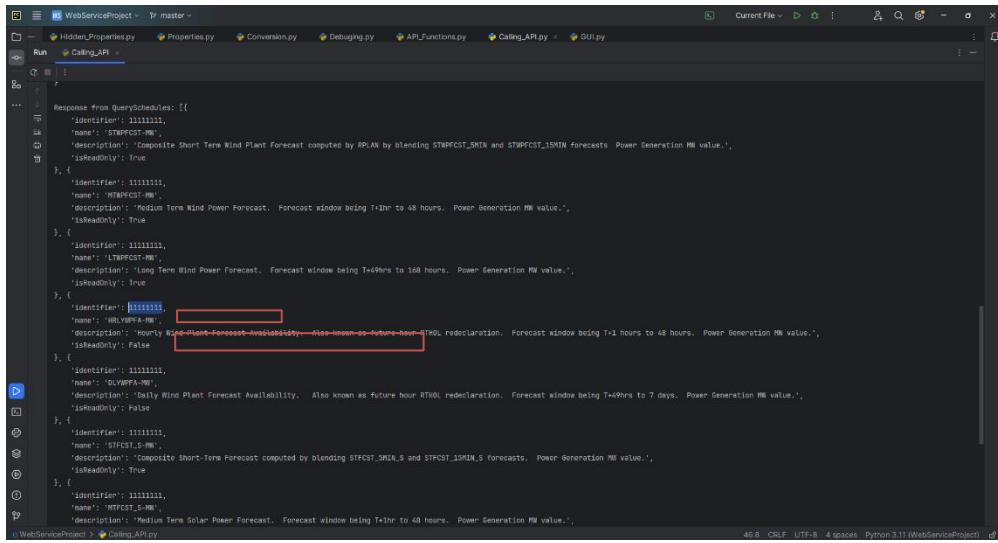
## Submit hourly forecast
# power_values_hourly = [8]*48
# submit_forecast(client, scheduleID=scheduleID_query, entityAssetID=entityAssetID,
#                 # powerValues=power_values_hourly, forecastType='hourly')
# print()
```

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.

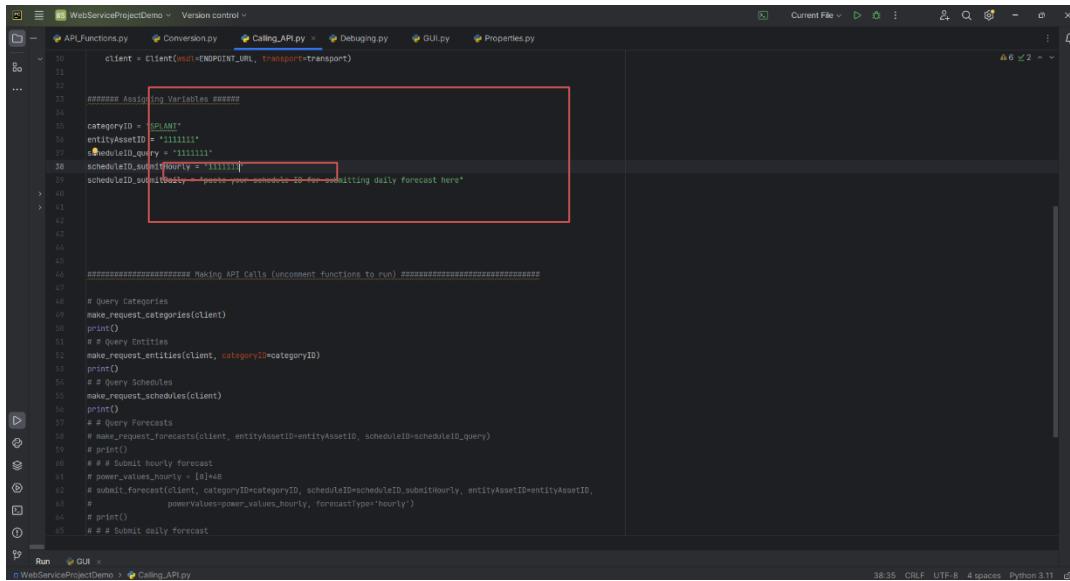


```

...
Response from QuerySchedules: [
    {
        'identifier': '1111111',
        'name': 'STMPFCST-MW',
        'description': 'Composite Short Term Wind Plant Forecast computed by RPLMN by blending STMPFCST_5MIN and STMPFCST_15MIN forecasts. Power Generation MW value.',
        'isReadOnly': True
    },
    {
        'identifier': '1111111',
        'name': 'HTMPFCST-MW',
        'description': 'Medium Term Wind Power Forecast. Forecast window being T+1hr to 48 hours. Power Generation MW value.',
        'isReadOnly': True
    },
    {
        'identifier': '1111111',
        'name': 'LTPFCST-MW',
        'description': 'Long Term Wind Power Forecast. Forecast window being T+0hrs to 168 hours. Power Generation MW value.',
        'isReadOnly': True
    },
    {
        'identifier': '1111111',
        'name': 'HBLMPFCST-MW',
        'description': 'Hourly Wind Plant Forecast Availability. Also known as future hour RTHDL redeclaration. Forecast window being T+1 hours to 48 hours. Power Generation MW value.',
        'isReadOnly': False
    },
    {
        'identifier': '1111111',
        'name': 'OLYMPFCST-MW',
        'description': 'Daily Wind Plant Forecast Availability. Also known as future hour RTHDL redeclaration. Forecast window being T+0hrs to 7 days. Power Generation MW value.',
        'isReadOnly': False
    },
    {
        'identifier': '1111111',
        'name': 'STFCST-S-MW',
        'description': 'Composite Short-Term Forecast computed by blending STFCST_5MIN_S and STFCST_10MIN_S forecasts. Power Generation MW value.',
        'isReadOnly': True
    },
    {
        'identifier': '1111111',
        'name': 'HTFCST-S-MW',
        'description': 'Medium Term Solar Power Forecast. Forecast window being T+1hr to 48 hours. Power Generation MW value.',
        'isReadOnly': True
    }
]

```

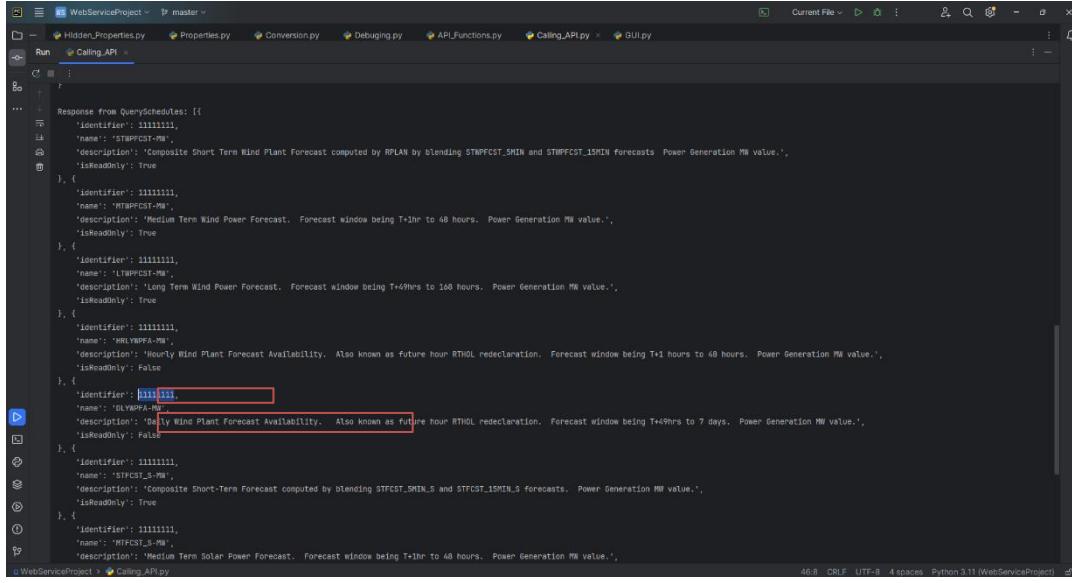
6.4.8. Enter the selected identifier into the “ScheduleID_submitHourly” variable by pasting it (“Ctrl” + “V”) into the quotation marks.



```

...
30 client = Client(endpoint=ENDPOINT_URL, transport=transport)
31
32 ##### Assigning Variables #####
33
34 categoryID = 'SPLANT'
35 entityAssetID = '1111111'
36 scheduleID_query = "1111111"
37 scheduleID_submitHourly = "1111111"
38 scheduleID_availability = '1111111'
39 scheduleID_availability = '1111111' #Leave your schedule ID for submitting daily forecast here'
40
41
42
43
44 ##### Making API Calls (uncomment Functions to run) #####
45
46
47
48 # Query Categories
49 make_request_categories(client)
50 print()
51 # # Query Entities
52 make_request_entities(client, categoryID=categoryID)
53 print()
54 # # Query Schedules
55 make_request_schedules(client)
56 print()
57 # # Query Forecasts
58 # make_request_forecasts(client, entityAssetID=entityAssetID, scheduleID=scheduleID_query)
59 # print()
60 # ## Submit hourly forecast
61 # power_values_hourly = [0]*48
62 # submit_forecast(client, categoryID=categoryID, scheduleID=scheduleID.submitHourly, entityAssetID=entityAssetID,
63 #                 powervalues=power_values_hourly, forecastType='hourly')
64 # print()
65 # ## Submit daily forecast
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
337
338
339
339
340
341
342
343
344
345
345
346
347
347
348
349
349
350
351
352
353
353
354
355
355
356
356
357
357
358
358
359
359
360
360
361
361
362
362
363
363
364
364
365
365
366
366
367
367
368
368
369
369
370
370
371
371
372
372
373
373
374
374
375
375
376
376
377
377
378
378
379
379
380
380
381
381
382
382
383
383
384
384
385
385
386
386
387
387
388
388
389
389
390
390
391
391
392
392
393
393
394
394
395
395
396
396
397
397
398
398
399
399
400
400
401
401
402
402
403
403
404
404
405
405
406
406
407
407
408
408
409
409
410
410
411
411
412
412
413
413
414
414
415
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
```

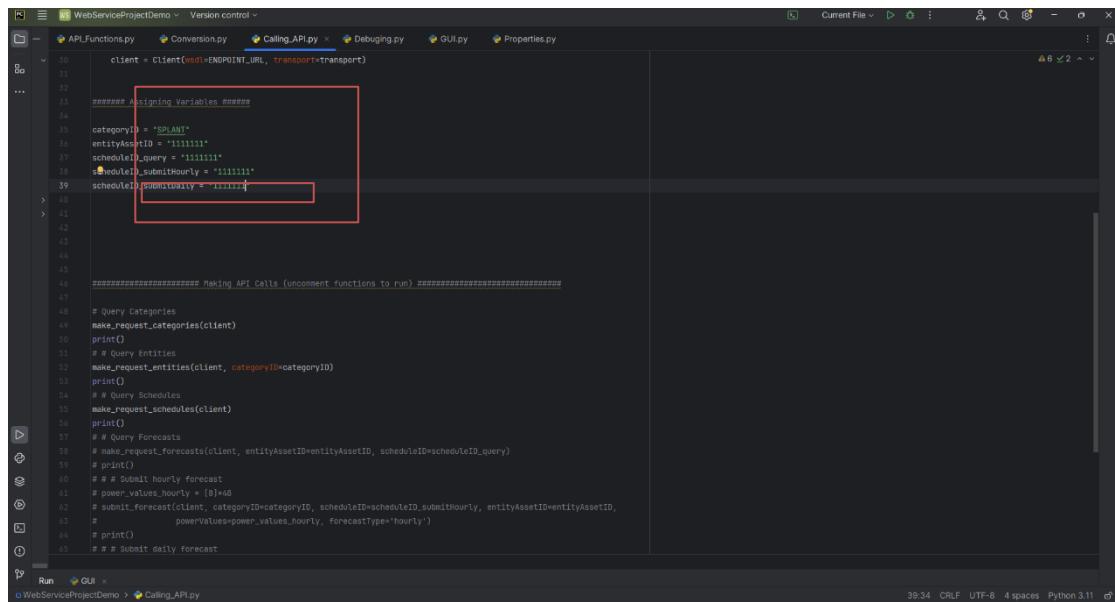
6.4.9.2. Click “Ctrl” + “C” to copy the identifier.



```
...  
    Response from QuerySchedules: [  
        {  
            'identifier': '11111111',  
            'name': 'STMPFCST_MW',  
            'description': 'Composite Short Term Wind Plant Forecast computed by RPLAN by blending STMPFCST_SWIN and STMPFCST_ISWIN forecasts Power Generation MW value.',  
            'isReadonly': True  
        },  
        {  
            'identifier': '11111111',  
            'name': 'MTMPFCST_MW',  
            'description': 'Medium Term Wind Power Forecast. Forecast window being T+1hr to 48 hours. Power Generation MW value.',  
            'isReadonly': True  
        },  
        {  
            'identifier': '11111111',  
            'name': 'LTMPFCST_MW',  
            'description': 'Long Term Wind Power Forecast. Forecast window being T+48hrs to 168 hours. Power Generation MW value.',  
            'isReadonly': True  
        },  
        {  
            'identifier': '11111111',  
            'name': 'HWLMPFCST_MW',  
            'description': 'Hourly Wind Plant Forecast Availability. Also known as future hour RTHOL redeclaration. Forecast window being T+1 hours to 48 hours. Power Generation MW value.',  
            'isReadonly': False  
        },  
        {  
            'identifier': '11111111',  
            'name': 'OLYMPFCST_MW',  
            'description': 'Daily Wind Plant Forecast Availability. Also known as future hour RTHOL redeclaration. Forecast window being T+49hrs to 7 days. Power Generation MW value.',  
            'isReadonly': False  
        },  
        {  
            'identifier': '11111111',  
            'name': 'STFCST_S_MW',  
            'description': 'Composite Short-Term Forecast computed by blending STFCST_SWIN_S and STFCST_ISWIN_S forecasts. Power Generation MW value.',  
            'isReadonly': True  
        },  
        {  
            'identifier': '11111111',  
            'name': 'MTFCST_S_MW',  
            'description': 'Medium Term Solar Power Forecast. Forecast window being T+1hr to 48 hours. Power Generation MW value.',  
            'isReadonly': True  
        }  
    ]
```

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.



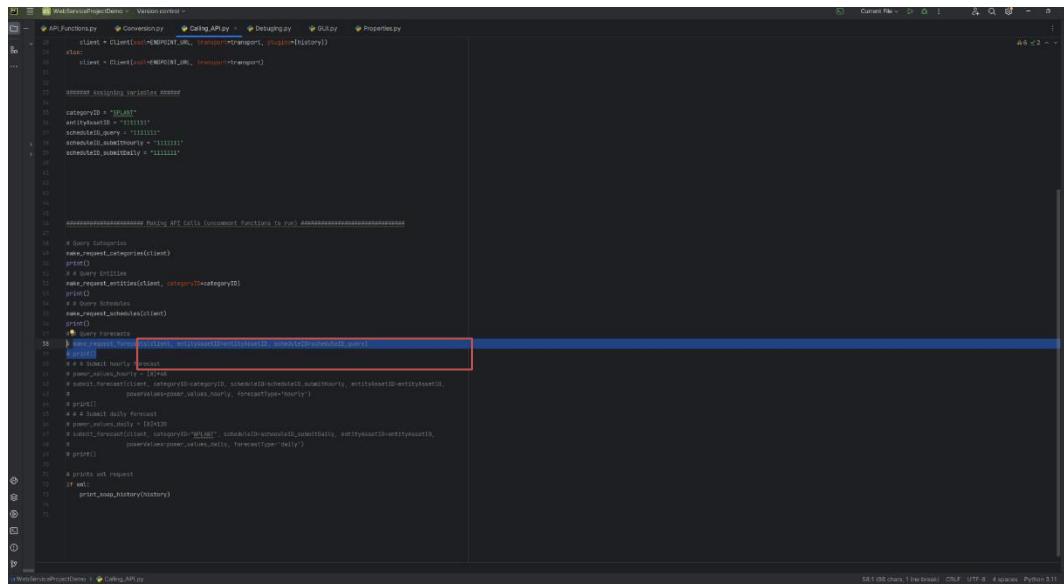
```
client = Client(url=ENDPOINT_URL, transport=transport)

#####
# Assigning Variables #####
categoryID = "SPLANT"
entityAssetID = "1111111"
scheduleID_query = "1111111"
scheduleID_submitHourly = "1111111"
scheduleID_submitDaily = "1111111"

#####
# Making API Calls (uncomment functions to run) #####
# Query Categories
make_request_categories(client)
print()
# # Query Entities
make_request_entities(client, categoryID=categoryID)
print()
# Query Schedules
make_request_schedules(client)
print()
# # Query Forecasts
# make_request_forecasts(client, entityAssetID=entityAssetID, scheduleID=scheduleID_query)
# print()
# # Submit hourly forecast
# power_values_hourly = [0]*48
# submit_forecast(cLimit, categoryID=categoryID, scheduleID=submitHourly, entityAssetID=entityAssetID,
#                 powerValues=power_values_hourly, forecastType='hourly')
# print()
# # Submit daily forecast
```

6.5. Calling make_request_forecasts()

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

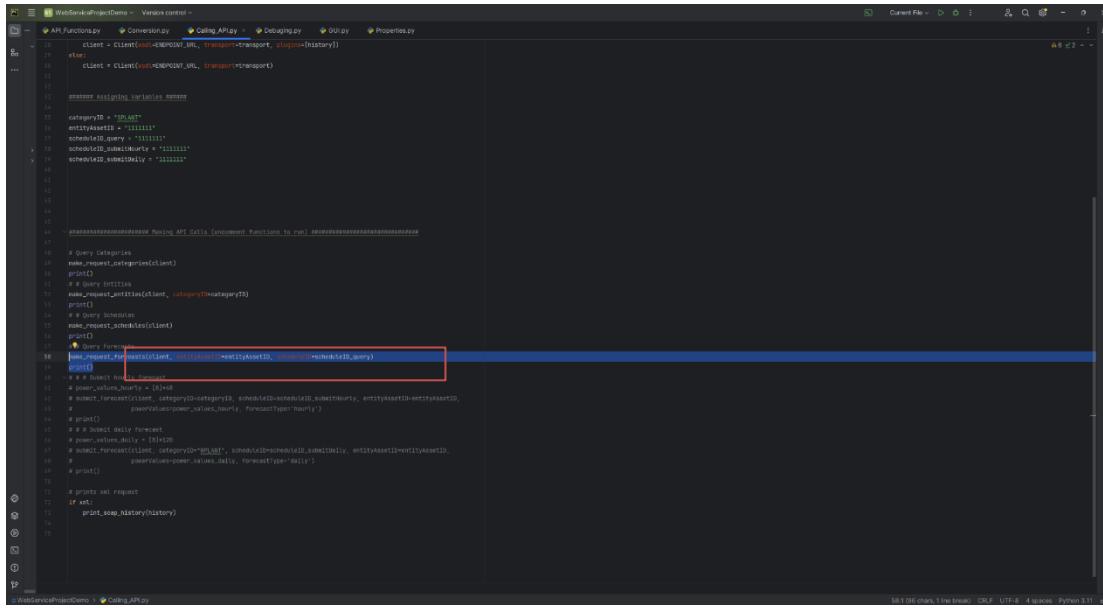


```
client = Client(url=ENDPOINT_URL, transport=transport, disableHistory=True)
client = Client(url=ENDPOINT_URL, transport=transport)

#####
# Assigning Variables #####
categoryID = "SPLANT"
entityAssetID = "1111111"
scheduleID_query = "1111111"
scheduleID_submitHourly = "1111111"
scheduleID_submitDaily = "1111111"

#####
# Making API Calls (uncomment Functions to run) #####
# Query Categories
make_request_categories(client)
print()
# # Query Entities
make_request_entities(client, categoryID=categoryID)
print()
# Query Schedules
make_request_schedules(client)
print()
# # Query Forecasts
# make_request_forecasts(client, entityAssetID=entityAssetID, scheduleID=scheduleID_query)
# print()
# # Submit hourly forecast
# power_values_hourly = [0]*48
# submit_forecast(cLimit, categoryID=categoryID, scheduleID=submitHourly, entityAssetID=entityAssetID,
#                 powerValues=power_values_hourly, forecastType='hourly')
# print()
# # Submit daily forecast
```

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



```
 31     else:
 32         client = Client(wsdl=ENDPOINT_URL, transport=transport, timeout=history)
 33
 34
 35     # enumerating variables memory
 36
 37     categoryID = "1010101"
 38     entityAsset = "11111111"
 39     scheduleID_query = "11111111"
 40     scheduleID_subentityHourly = "11111111"
 41     scheduleID_subentityDaily = "11111111"
 42
 43
 44
 45
 46     # uncommented code below Making API Calls (uncomment Functions to run) uncommmented
 47
 48     # Query Categories
 49     make_request_categories(client)
 50     print()
 51     # Query Assets
 52     make_request_entities(client, categoryID=categoryID)
 53     print()
 54     # Query Generators
 55     make_request_schedules(client)
 56     print()
 57     # Query Forecast
 58     make_request_forecast(client, categoryID=categoryID, scheduleID=scheduleID_query)
 59
 60
 61     # # Reset hour forecast
 62     # power_values_hourly = []
 63     # client_forecast(client, categoryID=categoryID, scheduleID=scheduleID_subentityHourly, entityAssetID=entityAssetID,
 64     #                 forecastType='hourly')
 65     # print()
 66     # # Reset daily forecast
 67     # power_values_daily = []
 68     # client_forecast(client, categoryID=categoryID, scheduleID=scheduleID_subentityDaily, entityAssetID=entityAssetID,
 69     #                 forecastType='daily')
 70     # print()
 71
 72     # prints xml request
 73     if ask:
 74         print_soap_history(history)
 75
 76
```

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()

6.6.1. Submitting the Hourly Forecast.

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

```

1  client = Client(url=ENDPOINT_URL, transport=transport)
2
3  ...
4
5  # Assigning Variables
6
7  categoryID = "00_001"
8  entitySetID = "1111111"
9  scheduleID_query = "111111"
10 scheduleID_submitDaily = "1111111"
11 scheduleID_submitHourly = "1111111"
12
13
14
15  # Making API Calls (Uncomment Functions as you)
16
17
18  # Query Categories
19  make_request_categories(client)
20  print()
21  # Query Entities
22  make_request_entities(client, categoryID=categoryID)
23  print()
24  # Query Schedules
25  make_request_schedules(client)
26  print()
27  # Query Forecasts
28  make_request_forecasts(client, entitySetID=entitySetID, scheduleID=scheduleID_query)
29  print()
30  # Power Values - Hourly
31  # power_values_hourly = [None]*120
32  # scheduleID_submitHourly, entitySetID_submitHourly, entitySetID_submitDaily,
33  # power_values_hourly, forecastType='hourly'
34  # print()
35  # # Submit hourly forecast
36  # power_values_hourly = [None]*120
37  # submit_forecast(client, categoryID="00_001", scheduleID_submitDaily, entitySetID=entitySetID,
38  #                 power_values=power_values_daily, forecastType='daily')
39  # print()
40
41  # prints last request
42  if val:
43      print_last_request()
44
45  # prints history
46  print_map_history(history)
47
48

```

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](#) section below for more information on submitting power values.

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.

The screenshot shows a code editor window with a Python script named 'Calling_API.py'. A red rectangular box highlights the following code block:

```
    # Submit daily forecast
    power_values_daily = [0]*120
    submit_forecast(client, categoryID=scheduleID.scheduleID_hourly, entityAssetID=entityAssetID,
    power_values=power_values_daily, forecastType='daily')
    print(power_values)
```

The code is part of a larger script that interacts with a web service, performing various queries and submissions. The highlighted section is specifically for submitting a daily forecast.

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](#) section below for more information on submitting power values.

```

10 client = Client('https://monint.url', transport=transport)
...
30
31 # Create New Entity Asset
32
33 categoryID = "500001"
34 entityAssetID = "111111"
35 scheduleID_query = "111111"
36 scheduleID_assetQuery = "111111"
37 assetID_query = "111111"
38
39
40 # ***** Making API Calls (comment functions to run) *****
41
42 # Query Categories
43 raw_request_categories(client)
44 print()
45 # Query Entities
46 raw_request_entities(client, categoryID=categoryID)
47 print()
48 # Query Schedules
49 raw_request_schedule(client)
50 print()
51 # Query Forecasts
52 raw_request_forecast(client, entityAssetID=entityAssetID, scheduleID=scheduleID_query)
53 print()
54 # X Submit hourly forecast
55 power_values_hourly = []
56 submit_forecast(client, entityAssetCategoryID, scheduleID_assetQuery, entityAssetID=entityAssetID,
57                 power_values_hourly, ForecastType='Hourly')
58 print()
59
60 # Submit daily forecast
61 submit_forecast(client, entityAssetCategoryID, entityAssetID, assetID_query, entityAssetID,
62                 power_values_daily, ForecastType='Daily')
63 print()
64
65 # prints out request
66 if wml:
67     printSoapHistory(wml)
68
69
70

```

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.

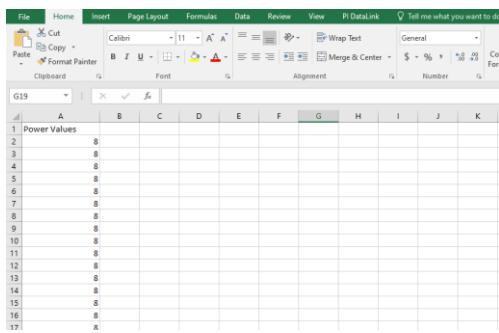
7. Getting Forecast Values

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](#) 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](#) 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](#) 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](#). 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](#) 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	B	C	D	E	F	G	H	I	J	K
1	Power Values	B									
2		B									
3		B									
4		B									
5		B									
6		B									
7		B									
8		B									
9		B									
10		B									
11		B									
12		B									
13		B									
14		B									
15		B									
16		B									
17		R									

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

```
power_values = pd.read_csv("dummyData.csv")
list_of_power_values = power_values["Power Values"].tolist()
```

7.3. Use the list as input to [submit_forecast](#):

```
submit_forecast(client, categoryID="WPLANT", scheduleID=schedule_id_submitDaily, entityAssetID=entityAssetID,
                powerValues=list_of_power_values, forecastType='daily')
```

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.

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.”

```
last request: <soap-env:Envelope xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">
<soap-env:Body>
  <ns0:QueryEntities xmlns:ns0="urn:com.alstom.isone.windint.windintegration:1-0">
    <ns0:Category>
      <ns0:Identifier>WPLANT</ns0:Identifier>
    </ns0:Category>
    </ns0:QueryEntities>
  </soap-env:Body>
</soap-env:Envelope>

last response: <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
  <QueryEntitiesResponse xmlns="urn:com.alstom.isone.windint.windintegration:1-0">
    <Category>
      <Identifier>WPLANT</Identifier>
    </Category>
    <Entities>
      <Entity>
        <Identifier>____</Identifier>
        <AssetIdentifier>____</AssetIdentifier>
        <Name>____</Name>
        <Description>____</Description>
        <IsReadOnly>False</IsReadOnly>
      </Entity>
    </Entities>
  </QueryEntitiesResponse>
</soap:Body>
</soap:Envelope>
```

9. Function Definitions

9.1. make_request_categories()

```
def make_request_categories():
    try:
        response = (client.service.QueryCategories())
        print("Response from QueryCategories:", response)
    except Exception as e:
        print(f"An error occurred: {e}")
```

- **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:**

```

Response from QueryCategories: [{  

    'identifier': 'AREA',  

    'name': 'CongestionArea',  

    'description': 'A logical grouping of Congestion Areas'  

}, {  

    'identifier': 'SPLANT',  

    'name': 'SolarPlant',  

    'description': 'A logical grouping of Solar Plants'  

}, {  

    'identifier': 'SYSTEM',  

    'name': 'System',]  

    'description': 'Entire System - a singleton entity'  

}, {  

    'identifier': 'WPLANT',  

    'name': 'WindPlant',  

    'description': 'A logical grouping of Wind Plants'  

}]

Process finished with exit code 0

```

9.2. make_request_schedules()

```

def make_request_schedules():  

    try:  

        response = (client.service.QuerySchedules())  

        print("Response from QuerySchedules:", response)  

    except Exception as e:  

        print(f"An error occurred: {e}")

```

- **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):**

```
Response from QuerySchedules: [
    {
        'identifier': '11111111',
        'name': 'STWPFCST-MW',
        'description': 'Composite Short Term Wind Plant Forecast computed by RPLAN by blending STWPFCST_SMIN and STWPFCST_ISMIN forecasts. Power Generation MW value.',
        'isReadOnly': True
    },
    {
        'identifier': '11111111',
        'name': 'MTWPFCST-MW',
        'description': 'Medium Term Wind Power Forecast. Forecast window being T+1hr to 48 hours. Power Generation MW value.',
        'isReadOnly': True
    },
    {
        'identifier': '11111111',
        'name': 'LTWPFCST-MW',
        'description': 'Long Term Wind Power Forecast. Forecast window being T+49hrs to 108 hours. Power Generation MW value.',
        'isReadOnly': True
    },
    {
        'identifier': '11111111',
        'name': 'HRLYWPFA-MW',
        'description': 'Hourly Wind Plant Forecast Availability. Also known as future hour RTHOL redeclaration. Forecast window being T+1 hours to 48 hours. Power Generation MW value.',
        'isReadOnly': False
    },
    {
        'identifier': '11111111',
        'name': 'DLYWPFA-MW',
        'description': 'Daily Wind Plant Forecast Availability. Also known as future hour RTHOL redeclaration. Forecast window being T+49hrs to 7 days. Power Generation MW value.',
        'isReadOnly': False
    },
    {
        'identifier': '11111111',
        'name': 'STFCST-5-MW',
        'description': 'Composite Short-Term Forecast computed by blending STFCST_SMIN_5 and STFCST_ISMIN_5 forecasts. Power Generation MW value.',
        'isReadOnly': True
    }
]
```

9.3. make_request_entities(cat=None)

```
def make_request_entities(cat = None):
    category = {
        'identifier': cat,
        'name': None,
        'description': None
    }
    try:
        if cat == None:
            response = (client.service.QueryEntities())
        else:
            response = (client.service.QueryEntities(Category=category))
        print("Response from QueryEntities:", response)
    except Exception as e:
        print(f"An error occurred: {e}")
```

- **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”):**

```
Response from QueryEntities: {
  'Category': {
    'identifier': 1111111,
    'name': 'Example Plant',
    'description': 'Example Plant Description'
  },
  'Entities': {
    'Entity': [
      {
        'identifier': 1111111,
        'assetIdentifier': 1111111,
        'name': 'Example Plant',
        'description': 'Example Plant Description',
        'isReadOnly': False
      },
      {
        'identifier': 1111111,
        'assetIdentifier': 1111111,
        'name': 'Example Plant',
        'description': 'Example Plant Description',
        'isReadOnly': False
      },
      {
        'identifier': 1111111,
        'assetIdentifier': 1111111,
        'name': 'Example Plant',
        'description': 'Example Plant Description',
        'isReadOnly': True
      },
      {
        'identifier': 1111111,
        'assetIdentifier': 1111111,
        'name': 'Example Plant',
        'description': 'Example Plant Description',
        'isReadOnly': False
      }
    ]
  }
}
```

9.4. make_request_forecasts(entityID, entityAssetID, scheduleID)

```
3 usages  ▲ Wimer, Brooks
def make_request_forecasts(client, entityAssetID, scheduleID):
    schedule_request_data = {
        'Schedule': {
            'identifier': scheduleID
        },
        'Entities': {
            'Entity': {
                'identifier': None,
                'assetIdentifier': entityAssetID,
            }
        }
    }
    try:
        response = client.service.QueryForecast(ScheduleRequest=schedule_request_data)
        scrambled = scramble_forecast(response)
        print("Response from QueryForecasts:", scrambled)
        return response
    except Exception as e:
        print(f"An error occurred: {e}")
```

- **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:**

```
Response from QueryForecasts: [
  {
    'category': None,
    'schedule': [
      {
        'identifier': '11111111',
        'name': 'HRLYWPFA-MP',
        'description': None,
        'isReadOnly': None
      }
    ],
    'timeRange': {
      'fromTime': datetime.datetime(2024, 8, 1, 13, 0, 42, tzinfo=<isodate.tzinfo.Utc object at 0x0000010889407C50>),
      'toTime': datetime.datetime(2024, 8, 8, 13, 0, 42, tzinfo=<isodate.tzinfo.Utc object at 0x0000010889407C50>)
    },
    'timeInterval': 3600,
    'entities': {
      'entity': [
        {
          'identifier': '11111111',
          'assetIdentifier': '11111111',
          'name': 'Example Plant',
          'description': 'Example Plant Description',
          'isReadOnly': None,
          'Power': [
            {
              'time': datetime.datetime(2024, 8, 1, 14, 0, tzinfo=<isodate.tzinfo.Utc object at 0x0000010889407C50>),
              'value': Decimal('8')
            },
            {
              'time': datetime.datetime(2024, 8, 1, 15, 0, tzinfo=<isodate.tzinfo.Utc object at 0x0000010889407C50>),
              'value': Decimal('8')
            },
            {
              'time': datetime.datetime(2024, 8, 1, 16, 0, tzinfo=<isodate.tzinfo.Utc object at 0x0000010889407C50>),
              'value': Decimal('8')
            }
          ]
        }
      ]
    }
  }
]
```

9.5. generate_power_entries(start_time, num_entries, power_values)

```

def generate_power_entries(start_time, num_entries, power_values):
    power_entries = []
    start_time = datetime.strptime(start_time, "%Y-%m-%dT%H:%M:%S%z")
    for i in range(num_entries):
        entry_time = start_time + timedelta(hours=i)
        entry_time_str = entry_time.strftime("%Y-%m-%dT%H:%M:%S%z")
        entry_time_str = entry_time_str[:-2] + ':' + entry_time_str[-2:] # Adjusting timezone format
        power_entries.append({
            'time': entry_time_str,
            'value': power_values[i-1]
        })
    return power_entries

```

- **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)

```
def submit_forecast(client, scheduleId, entityId, entityAssetId, powerValues, forecastType):
    now = pytz.timezone('UTC').localize(datetime.now())
    if forecastType == "hourly":
        # Set the current date and time to UTC
        now_utc = datetime.now(pytz.UTC)

        # Forecast start time is Forecast start time.replace(hour=0, second=0) if hour is the top of the hour
        if forecast_start_time := Forecast.start_time.replace(hour=0, second=0):
            forecast_start_time = forecast_start_time.astimezone(utc).replace(hour=0, second=0, microsecond=0)

        else:
            now_entries = 120
            # Calculate the date and time two days from now in UTC
            two_days_later = now_utc + timedelta(days=2)
            # Convert the UTC time to EST
            forecast_start_time = two_days_later.astimezone(utc).astimezone(est).replace(hour=0, minute=0, second=0, microsecond=0)

        forecast_start_time = forecast_start_time.astimezone(utc).replace(hour=0, minute=0, second=0, microsecond=0)

    power_entries = generate_power_entries(forecast_start_time.astimezone("EST"), now_entries, powerValues)

    forecast_data = {
        "ForecastSchedule": {
            "Category": [
                {"CategoryID": category_id, "CategoryName": category_name}
            ],
            "Schedule": [
                {"ScheduleID": scheduleId, "ScheduleName": schedule_name}
            ],
            "Entries": [
                {
                    "Entity": [
                        {"entityIdentifier": entityAssetId, "Power": power_entries}
                    ]
                }
            ]
        }
    }

    try:
        response = client.service.SubmitForecast(CreateForecastRequest(forecastData=forecast_data))
        print("Response from SubmitForecast:", response)
        return response
    except Exception as e:
        print(f"An error occurred: {e}")


```

- 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:**

```
Response from SubmitForecast: {  
    'transactionId': '61fc2f0e-e091-4ac5-94c3-d7ad5ecf5b2f'  
}  
  
Process finished with exit code 0
```