



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...”

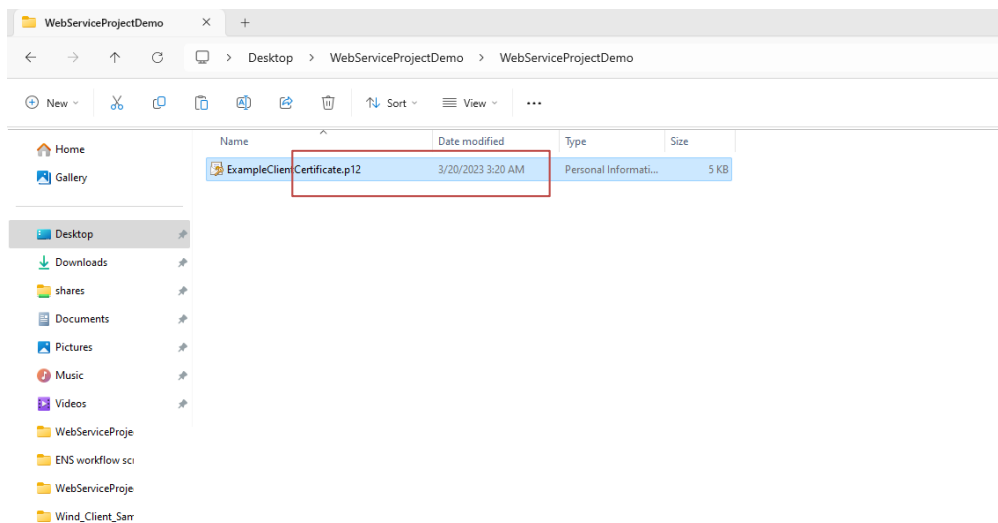
0.1.2. Keep track of this unzipped project folder; it will be used in future steps.

1. Configuring Certificates

1.1. Importing Client Certificate

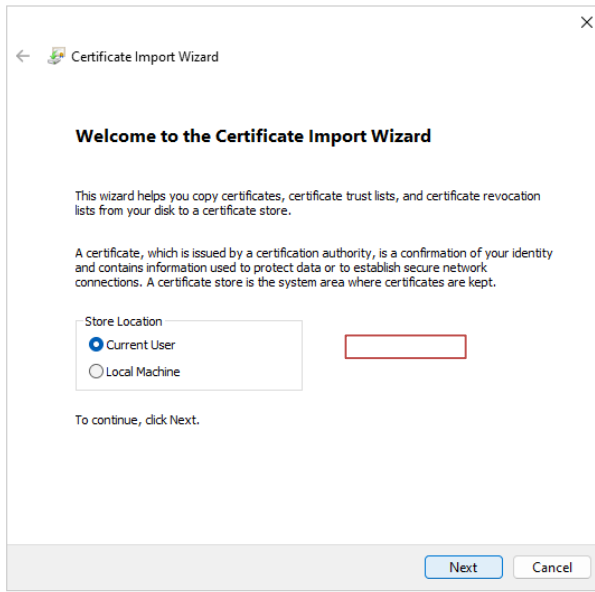
Refer to the instructions below, or follow along with this [video](#).

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



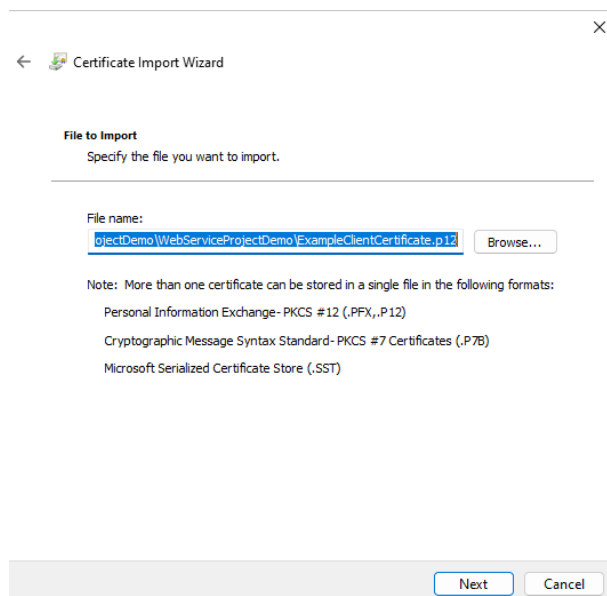
1.1.2. Store Location should be set to Current User.

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.

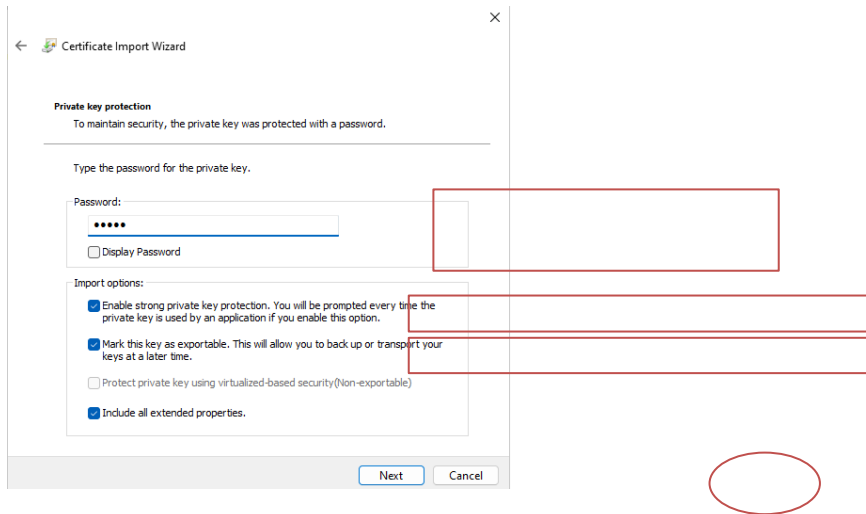


1.1.4. Input certificate password.

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.”

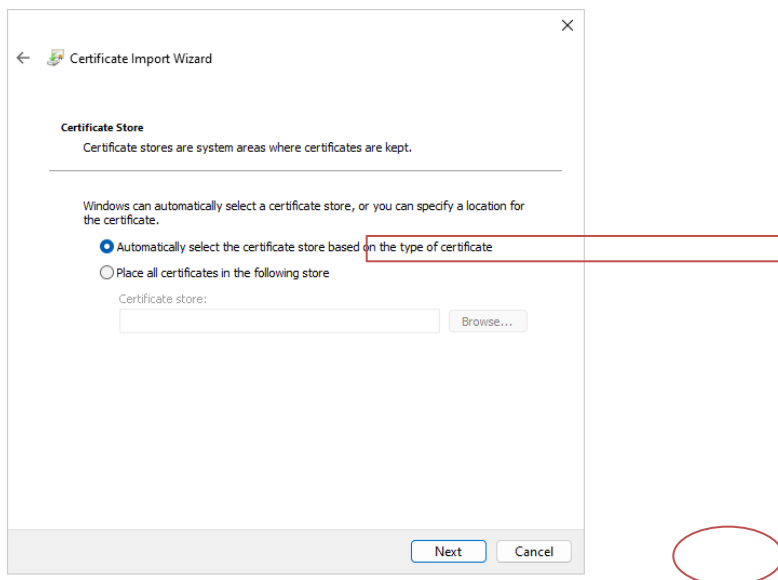
1.1.4.2. Click the box next to “Mark this key as exportable. This will allow you to back up or transport your keys at a later time.”

1.1.4.3. Click Next.

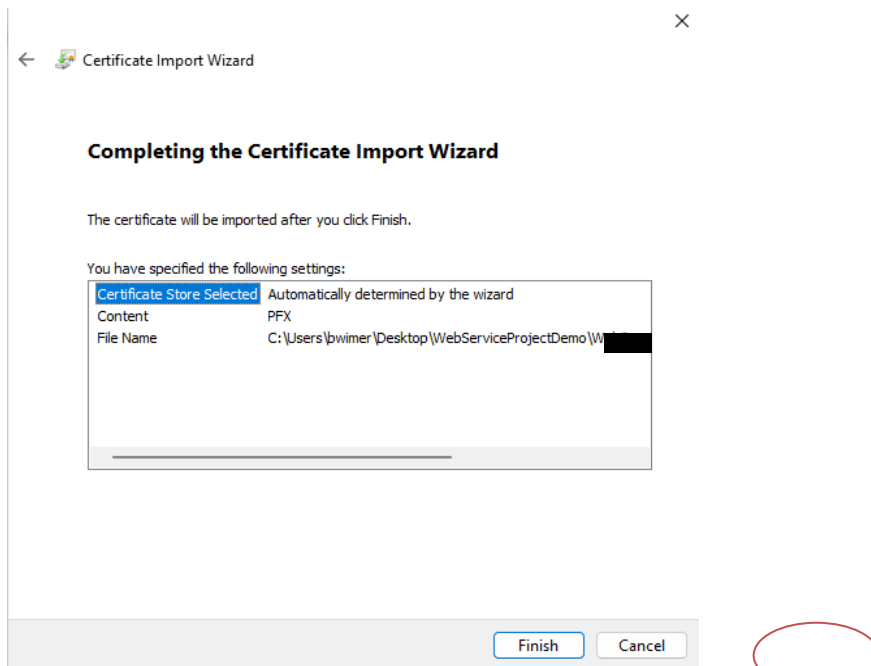


1.1.5. Select a specific keystore or allow automatic selection by selecting “Automatically select the certificate store based on the type of certificate.”

1.1.5.1. Click Next.

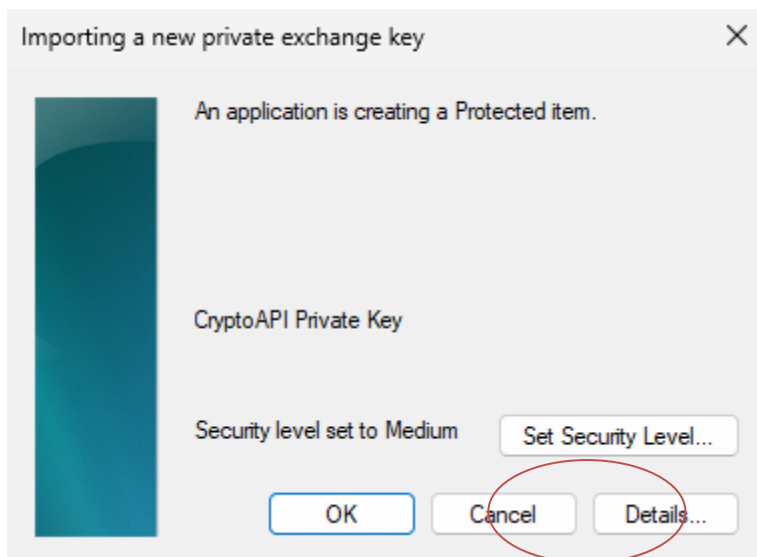


1.1.6. Click Finish.



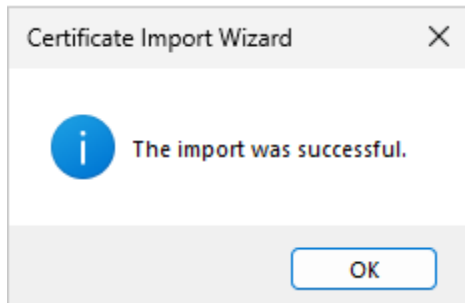
1.1.7. A pop-up should appear that looks like the figure below.

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).

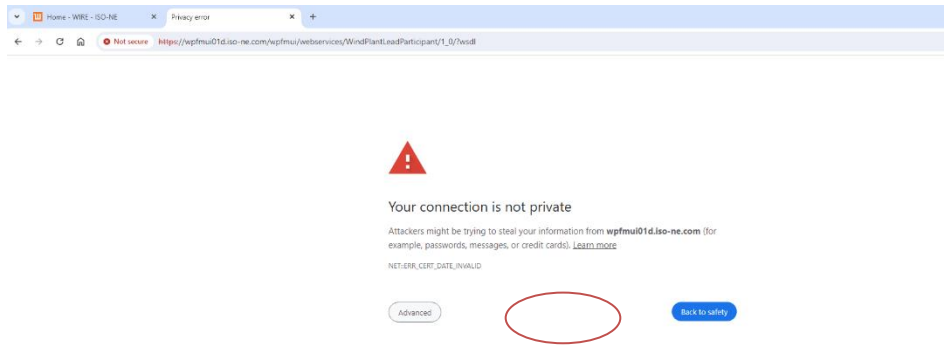
1.2.1.1. Use the **Sandbox** environment for initial setup and testing.

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

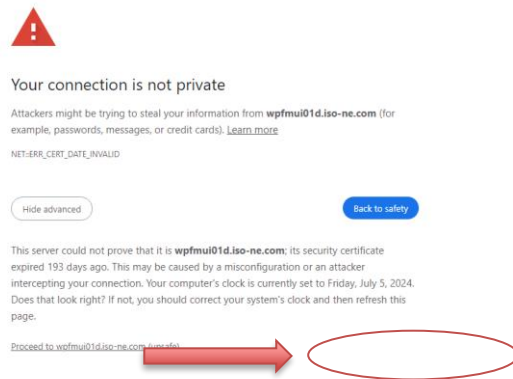
1.2.1.2. Use the **Production** environment when ready to start submitting WPFA/SPFA.

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

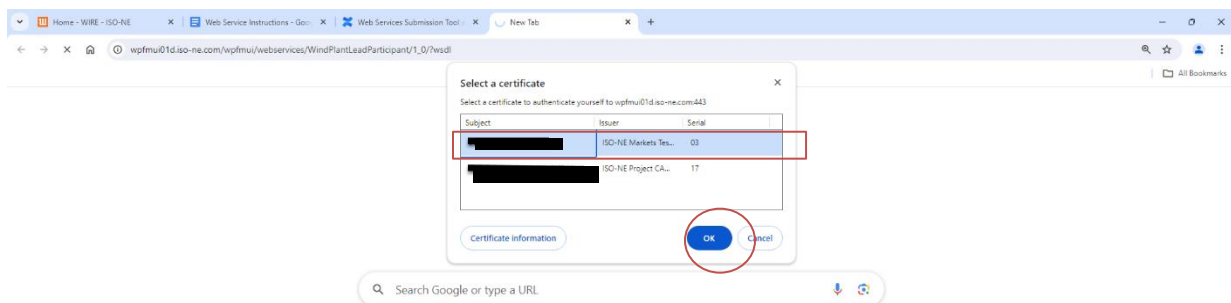
1.2.2. Click Advanced.



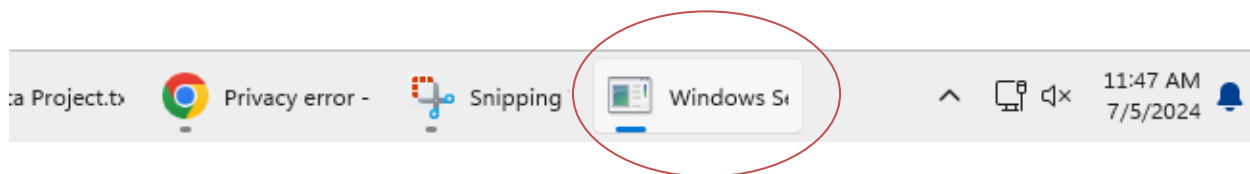
1.2.3. Click Proceed to wpfmu01d.iso-ne.com (unsafe).



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

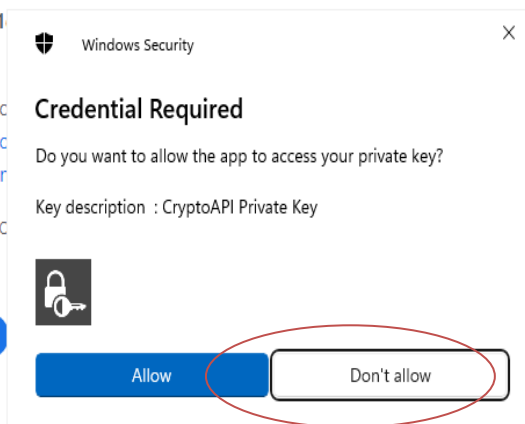
wpfmu01

Try:

- Check
- Check
- Run

ERR_TIMED_C

Reload



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



← → ↻ 🔒

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

View site information

wpfmu01d.iso-ne.com

⚠ 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 redirection Allowed (default)

Cookies and site data

Site settings

This XML file contains instructions on how to use the web service. The document tree is shown below.

▼ <definitions>

xmlns:tns="http://schemas.xmlsoap.org/wsdl/tns/"

xmlns:xsd="http://www.w3.org/2001/XMLSchema"

▼ <types>

▼ <xs:schema>

xmlns="http://schemas.xmlsoap.org/wsdl/tns/"

xmlns:tns="http://schemas.xmlsoap.org/wsdl/tns/"

xmlns:xsd="http://www.w3.org/2001/XMLSchema"

</xs:schema>

</types>

▼ <message>

<part>

</message>

▼ <message>

<part element="wint:QueryForecast" name="parameters">

</message>

▼ <message name="SubmitForecast">

<part element="wint:SubmitForecast" name="parameters">

</message>

▼ <message name="AuthorizationFaultMessage">

<part element="wint:AuthorizationFault" name="parameters">

</message>

<?xml version="1.0" encoding="UTF-8"?>

<definitions xmlns:tns="http://schemas.xmlsoap.org/wsdl/tns/" xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<types>

<xs:schema xmlns="http://schemas.xmlsoap.org/wsdl/tns/" xmlns:tns="http://schemas.xmlsoap.org/wsdl/tns/" xmlns:xsd="http://www.w3.org/2001/XMLSchema">

</xs:schema>

</types>

<message>

<part>

</message>

<message name="SubmitForecast">

<part element="wint:SubmitForecast" name="parameters">

</message>

<message name="AuthorizationFaultMessage">

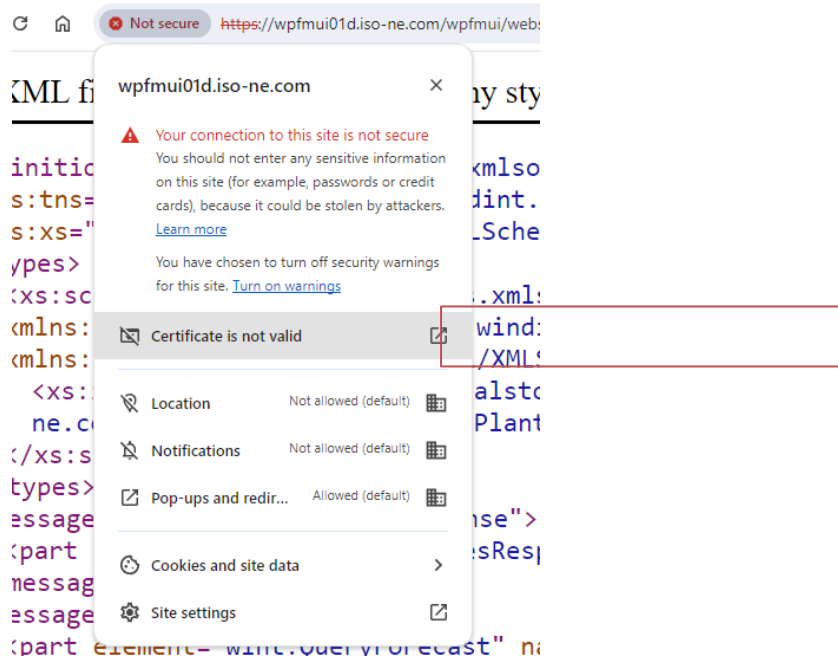
<part element="wint:AuthorizationFault" name="parameters">

</message>

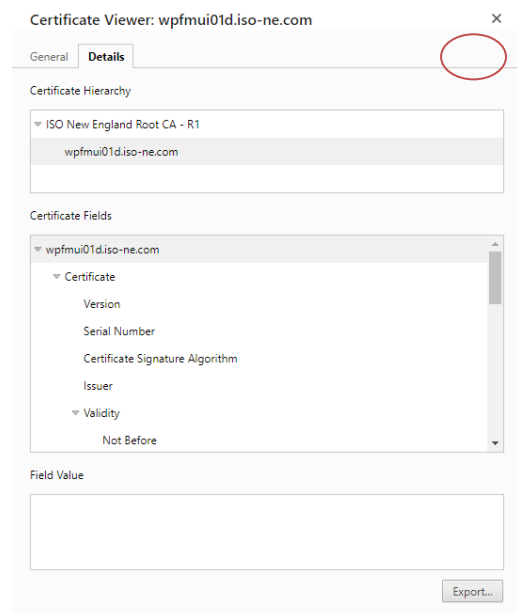
</definitions>

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.

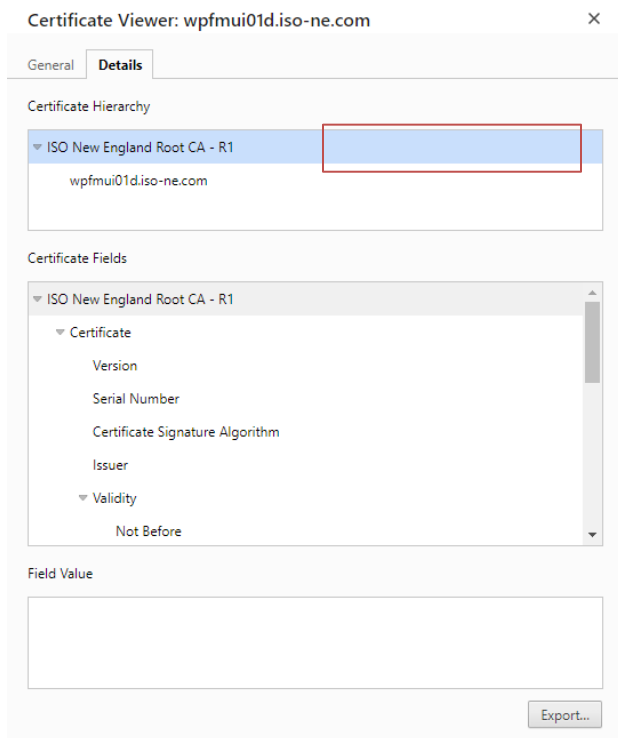


1.2.9. Go to the Details tab.



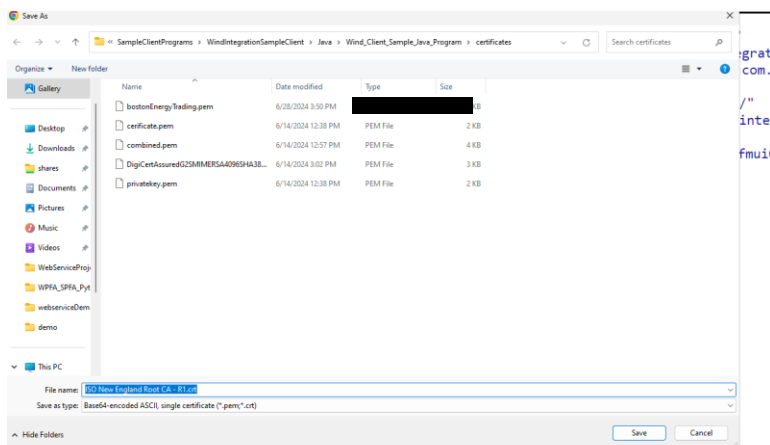
1.2.10. Click on the first certificate under “Certificate Hierarchy” and click “Export...”

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](https://www.pycharm.org/) 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>.

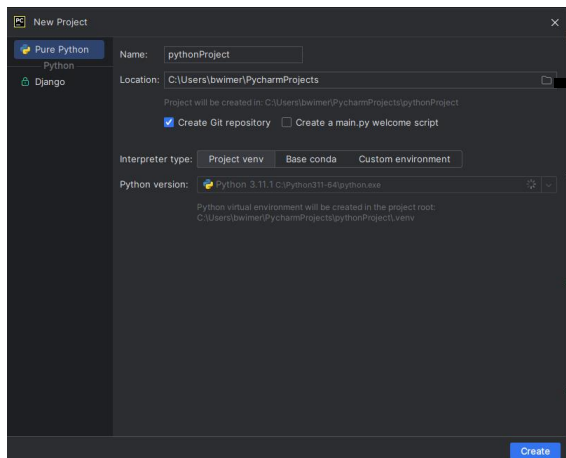
2.1.2. Follow the installation instructions.

2.2. Setting up a New Project

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

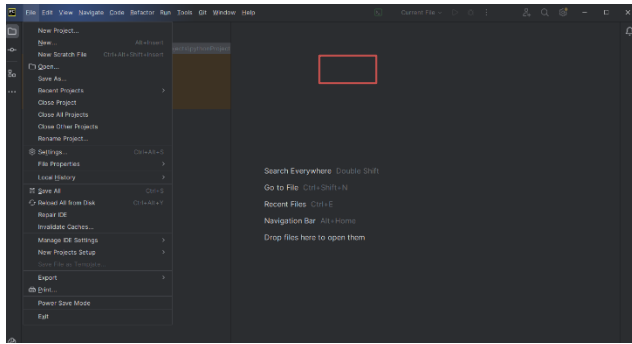
2.2.1. Open PyCharm and select “Create New Project.”

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

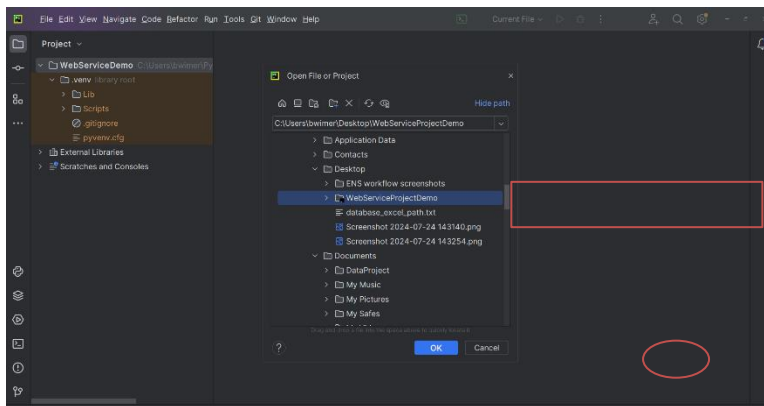


2.2.2 Select File and click Open.

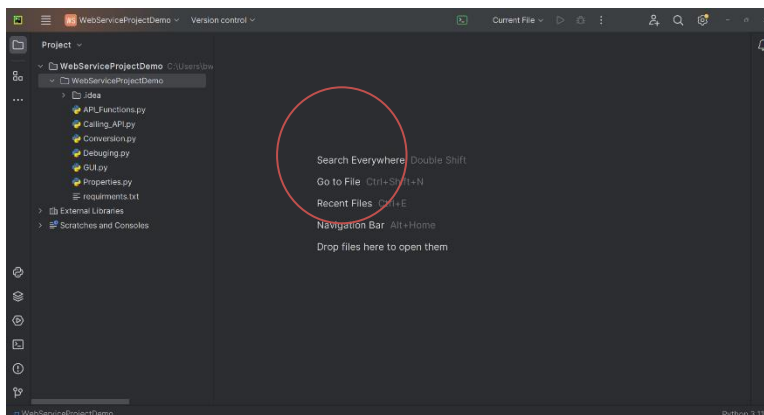




2.2.3. Navigate to the unzipped project folder and click OK.



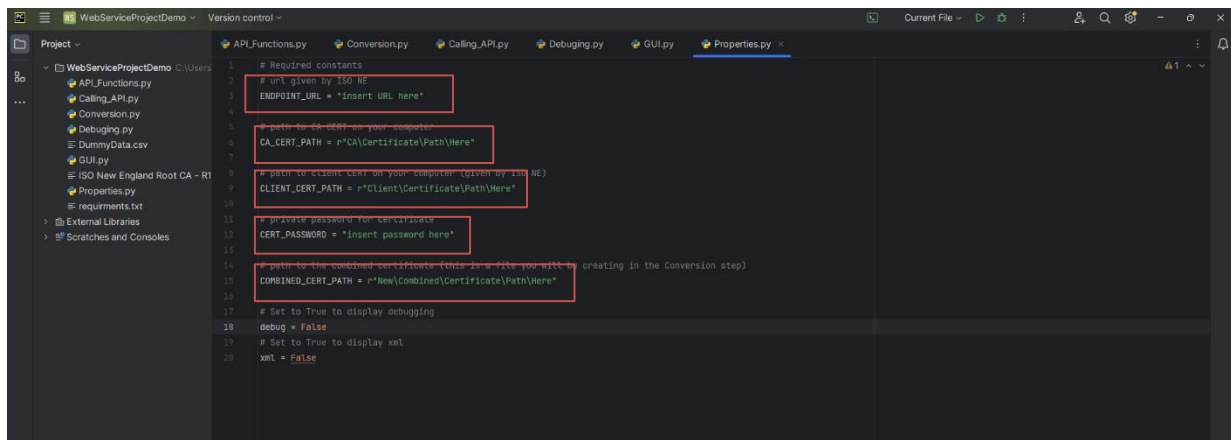
2.2.4. Open Python files by double clicking on each of the six .py files to view them in the Code Editor.



3. Configuring Properties

3.1. Navigate to the Properties.py Tab





3.2. ENDPOINT_URL

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

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

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

3.3. CA_CERT_PATH

3.3.1. Copy and paste the path to the CA Certificate into the “CA_CERT_PATH” variable.

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

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

3.4. CLIENT_CERT_PATH

3.4.1. Copy and paste the path to the Client Certificate into the “CLIENT_CERT_PATH” variable.

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

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

3.5. CERT_PASSWORD

3.5.1. Enter the certificate password into the “CERT_PASSWORD” variable.

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

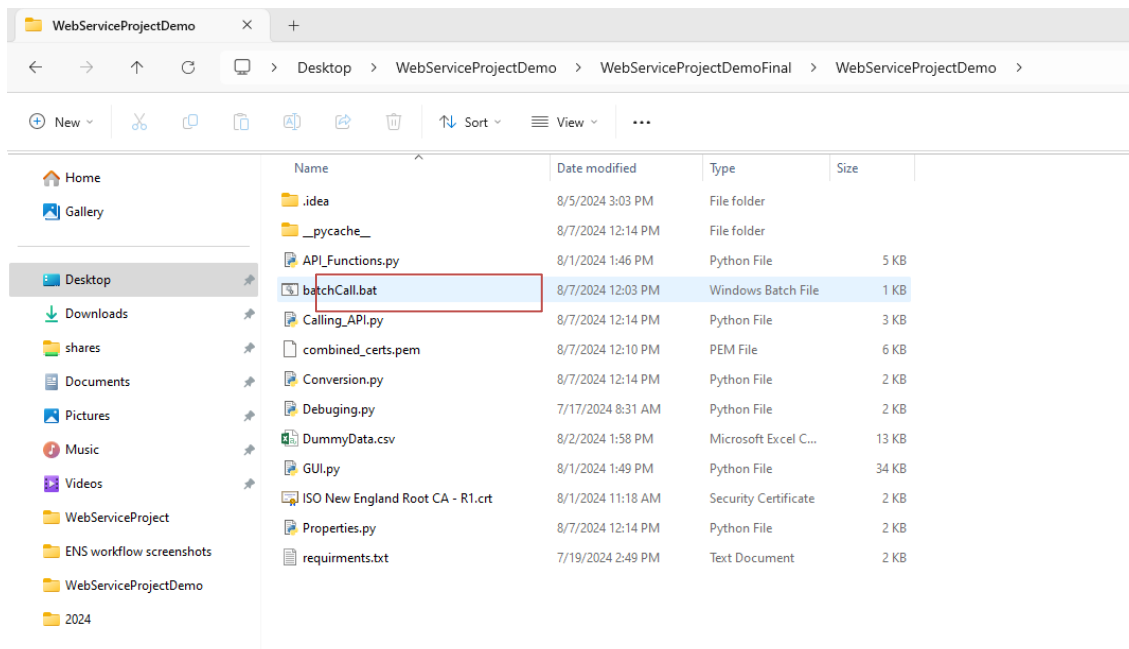
3.5.2. Ensure the password is surrounded by only one set of quotes with a letter r in front of it.

4. Converting and Combining Certificates

4.1. Navigate to the Unzipped Project Folder

4.1.1. Double click the file “batchCall.bat”

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



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

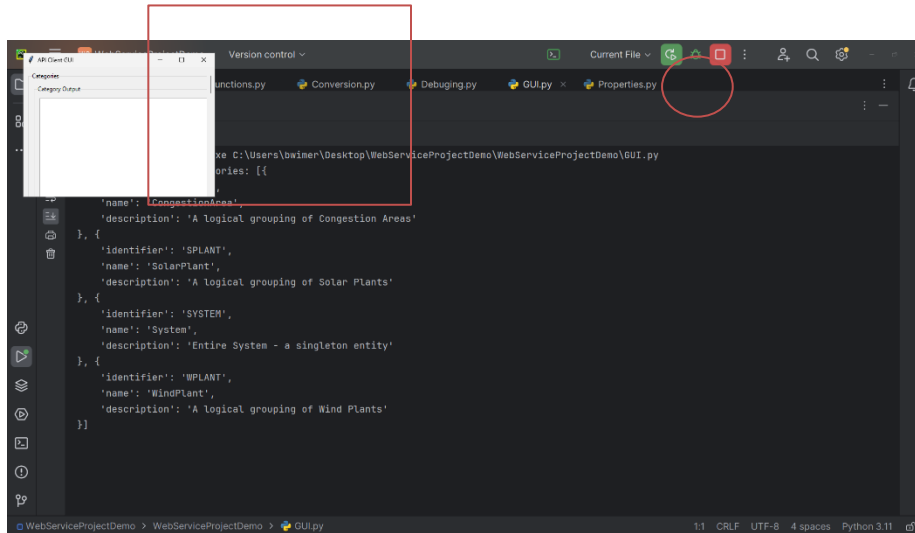
5.1.1 Navigate to the GUI.py tab.

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.

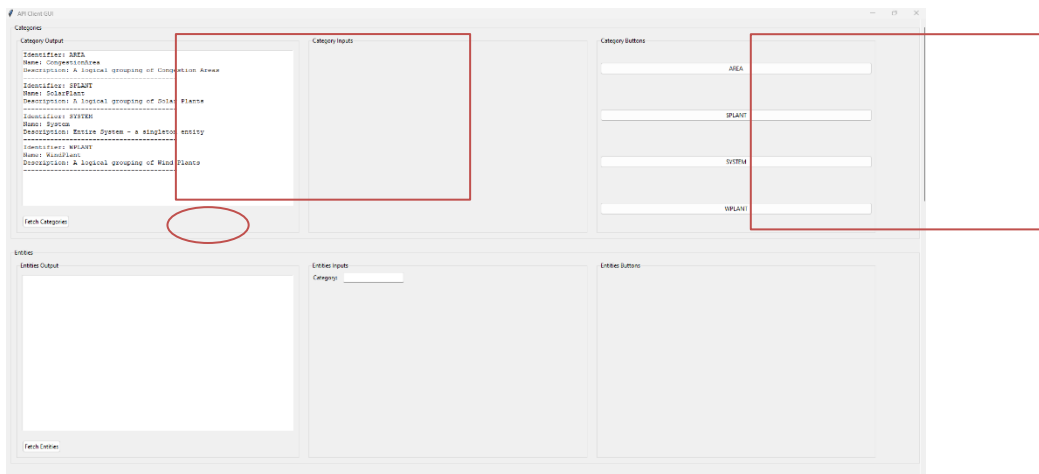
5.1.3. There should be a pop-up window like the one below; make it full screen.



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.

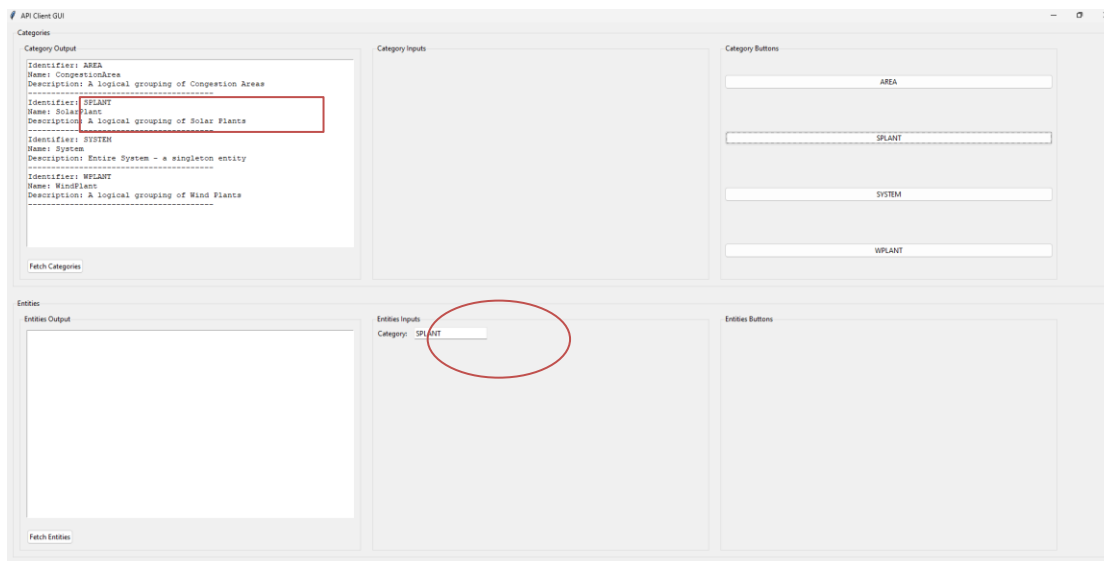


5.3. Select Category

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.

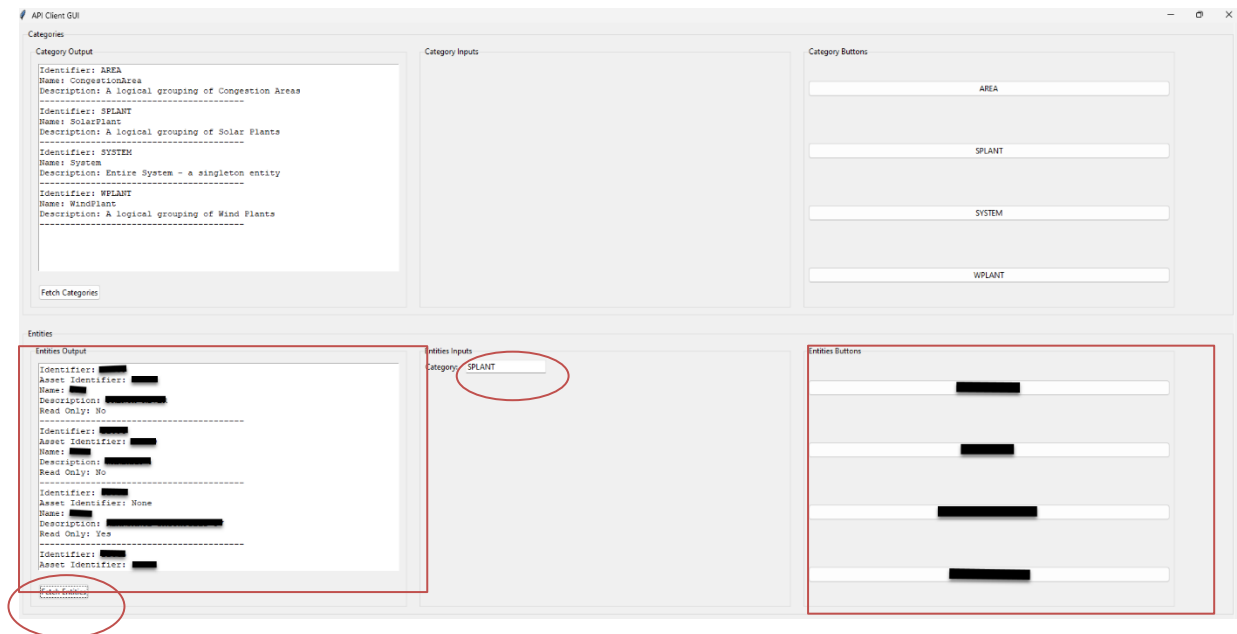


5.4. Fetch Entities

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 displays a user interface for managing schedules and forecasts. It is divided into two main sections: Schedules and Forecasts.

Schedules Section:

- Schedules Output:** A text area showing details for several schedules, including their identifiers, names, and descriptions. A "Fetch Schedules" button is located at the bottom of this panel.
- Schedules Inputs:** An empty panel for providing input to the schedule fetch function.
- Schedules Buttons:** A vertical list of buttons for different forecast types, such as "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", and "Daily Solar Plant Forecast Availability".

Forecasts Section:

- Forecasts Output:** A large empty box for displaying forecast results, with a "Fetch Forecasts" button at the bottom.
- Forecasts Inputs:** A panel with input fields for "Entity ID", "Asset ID", "Entity Name", "Schedule ID", and "Schedule Type". A red rectangular box highlights the "Entity ID", "Asset ID", and "Entity Name" fields.
- Forecasts Buttons:** An empty panel for forecast-related actions.

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: STWFFCST-15W
Description: Composite Short Term Wind Plant Forecast computed by RPLA
N by blending STWFFCST_15H1R and STWFFCST_15H1R forecasts. Power Generation MW value.
-----
Identifier: [REDACTED]
Name: MWFFCST-15W
Description: Medium Term Wind Power Forecast. Forecast window being T
+1hr to 48 hours. Power Generation MW value.
-----
Identifier: [REDACTED]
Name: LTWFFCST-15W
Description: Long Term Wind Power Forecast. Forecast window being T+4
8hrs to 144 hours. Power Generation MW value.
-----
Identifier: [REDACTED]
Name: HSLWFA-15W
Description: Hourly Wind Plant Forecast Availability. Also known as f
uture hour RTBOL redeclaration. Forecast window being T+1 hours to 48

```

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 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.

Schedules

Schedules Output

Identifier: STWFFCST-100
Name: STWFFCST-100
Description: Composite Short Term Wind Plant Forecast computed by RPLA P by blending STWFFCST_10MIN and STWFFCST_10MIN forecasts. Power Generation MW Value.

Identifier: MTWFFCST-100
Name: MTWFFCST-100
Description: Medium Term Wind Power Forecast. Forecast window being T-1hr to 48 hours. Power Generation MW Value.

Identifier: LTWFFCST-100
Name: LTWFFCST-100
Description: Long Term Wind Power Forecast. Forecast window being T-1hr to 144 hours. Power Generation MW Value.

Identifier: BDLWFFA-100
Name: BDLWFFA-100
Description: Hourly Wind Plant Forecast Availability. Also known as f value hour RTMOC declassification. Forecast window being T-1 hour to 48

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

Time	Value
2024-08-02 18:00:00 UTC	8.0
2024-08-02 19:00:00 UTC	8.0
2024-08-02 20:00:00 UTC	8.0
2024-08-02 21:00:00 UTC	8.0
2024-08-02 22:00:00 UTC	8.0
2024-08-02 23:00:00 UTC	8.0
2024-08-03 00:00:00 UTC	8.0
2024-08-03 01:00:00 UTC	8.0
2024-08-03 02:00:00 UTC	8.0
2024-08-03 03:00:00 UTC	8.0

Forecasts Inputs

Entity ID: [redacted]
Entity Asset ID: [redacted]
Entity Name: [redacted]
Schedule ID: [redacted]
Schedule Type: DWTWFFA-100

Forecasts Buttons

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: STWFFCST-09

Description: Composite Short Term Wind Plant Forecast computed by RPLA 0 by blending STWFFCST_092M and STWFFCST_10MIN forecasts. Power Generation MW value.

Identifier: [REDACTED]

Name: MTWFFCST-09

Description: Medium Term Wind Power Forecast. Forecast window being T-1hr to 48 hours. Power Generation MW value.

Identifier: [REDACTED]

Name: LTWFFCST-09

Description: Long Term Wind Power Forecast. Forecast window being T+6 hrs to 144 hours. Power Generation MW value.

Identifier: [REDACTED]

Name: ERLTWFFA-09

Description: Hourly Wind Plant Forecast Availability. Also known as E usure hour ETR0L declaration. Forecast window being T+1 hour to 48

Fetch Schedules

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

Time	Value
2024-08-01 18:00:00 UTC	8.0
2024-08-01 19:00:00 UTC	8.0
2024-08-01 20:00:00 UTC	8.0
2024-08-01 21:00:00 UTC	8.0
2024-08-01 22:00:00 UTC	8.0
2024-08-01 23:00:00 UTC	8.0
2024-08-02 00:00:00 UTC	8.0
2024-08-02 01:00:00 UTC	8.0
2024-08-02 02:00:00 UTC	8.0
2024-08-02 03:00:00 UTC	8.0

Fetch Forecasts

Forecasts Inputs

Entity ID: [REDACTED]

Entity Asset ID: [REDACTED]

Entity Name: [REDACTED]

Schedule ID: [REDACTED]

Schedule Type: DLYWFFA-MW

Forecasts Buttons

5.9. Create and view CSV

5.9.1. Scroll down to the CSV Browser panel.

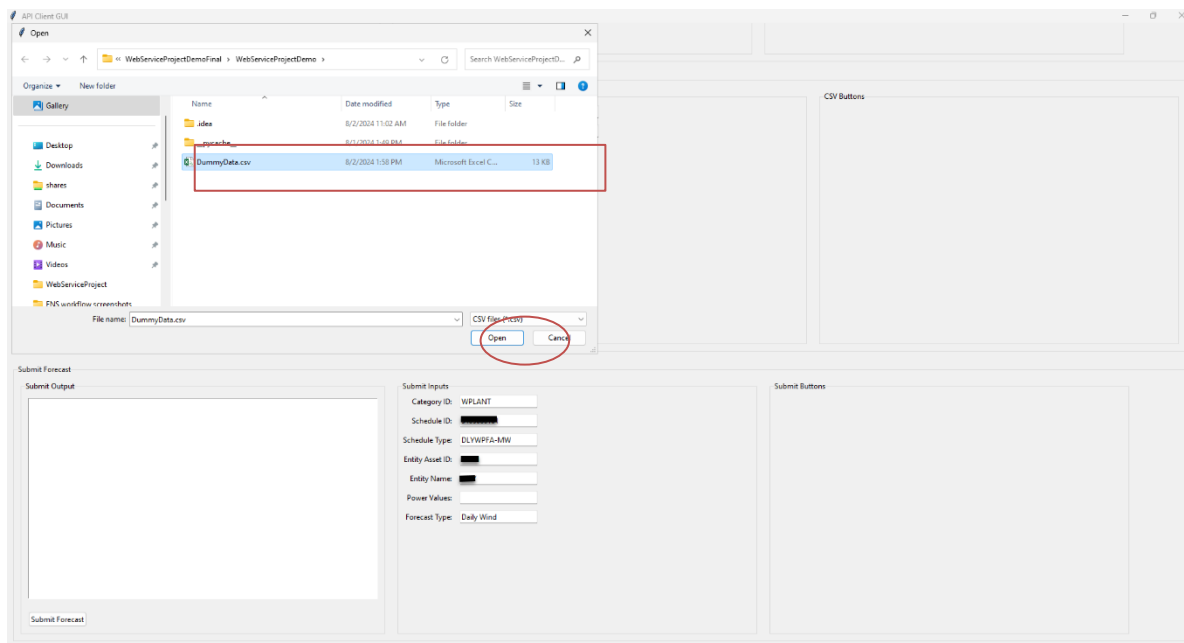
5.9.2. Click the Create CSV button in the CSV Output panel.

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.

The screenshot displays the 'CSV Browser' application window, which is divided into four main panels:

- CSV Output:** A large empty rectangular area on the top left, with a 'Browse CSV' button at the bottom left and a 'Create CSV' button at the bottom right.
- CSV Inputs:** A panel on the top right containing a 'Data Path' field with the value 'C:/Users/bwimer/Desktop' and a 'Selected Producer' dropdown menu.
- CSV Buttons:** A panel on the far right, outlined with a red rectangle. It contains a horizontal scroll bar with several buttons, one of which is highlighted with a black background.
- Submit Forecast:** A panel on the bottom left with a 'Submit Output' button at the bottom.
- Submit Inputs:** A central panel containing several input fields: 'Category ID' (set to 'WPLANT'), 'Schedule ID' (with a blacked-out value), 'Schedule Type' (set to 'OLVWPPA-MW'), 'Entity Asset ID' (with a blacked-out value), 'Entity Name' (with a blacked-out value), 'Power Values' (empty), and 'Forecast Type' (set to 'Daily Wind').
- Submit Buttons:** A large empty rectangular area on the bottom right.

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.

CSV Browser

CSV Output

Time	
2024-08-03 07:00:00	-46.0
2024-08-03 08:00:00	52.2
2024-08-03 09:00:00	35.0
2024-08-03 10:00:00	37.3
2024-08-03 11:00:00	66.1
2024-08-03 12:00:00	58.8
2024-08-03 13:00:00	30.5
2024-08-03 14:00:00	47.5
2024-08-03 15:00:00	69.9
2024-08-03 16:00:00	37.4

Browse CSV

Create CSV

CSV Inputs

Data Path: C:/Users/bsimier/Desktop

Selected Producer:

CSV Buttons

Submit Forecast

Submit Output

Submit Forecast

Submit Inputs

Category ID: WPLANT

Schedule ID:

Schedule Type: DLYWFFA-MW

Entity Asset ID:

Entity Name:

Power Values: -49.0,52.2,35.0,37.3,66.1

Forecast Type: Daily Wind

Submit Buttons

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.

CSV Browser

CSV Output

Time	
2024-08-03 07:00:00	49.0
2024-08-03 08:00:00	52.2
2024-08-03 09:00:00	35.0
2024-08-03 10:00:00	37.3
2024-08-03 11:00:00	66.1
2024-08-03 12:00:00	58.8
2024-08-03 13:00:00	30.5
2024-08-03 14:00:00	47.5
2024-08-03 15:00:00	69.9
2024-08-03 16:00:00	37.4

Browse CSV

Create CSV

CSV Inputs

Data Path: C:/Users/buime/Desktop

Selected Producer:

CSV Buttons

Submit Forecast

Submit Output

Success! Here is your transaction id:

27258094-6e9b-865f-82d9-aa34e9a13a5b

Submit Forecast

Submit Inputs

Category ID: WPLANT

Schedule ID:

Schedule Type: DLYWFFA-MW

Entity Asset ID:

Entity Name:

Power Values: 49.0,52.2,35.0,37.3,66.1

Forecast Type: Daily Wind

Submit Buttons

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.

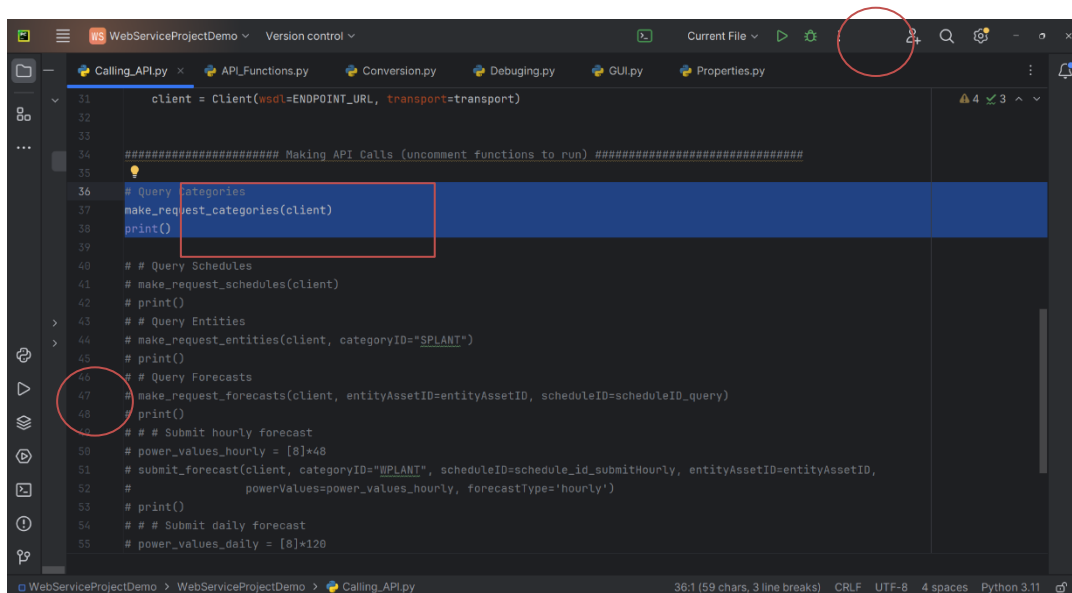
```
34 ##### Making API Calls (uncomment functions to run) #####
35
36 ## Query Categories
37 # make_request_categories(client)
38 # print()
39 #
40 ## Query Schedules
41 # make_request_schedules(client)
42 # print()
43 ## Query Entities
44 # make_request_entities(client, categoryID="SPLANT")
45 # print()
46 ## Query Forecasts
47 # make_request_forecasts(client, entityAssetID=entityAssetID, scheduleID=scheduleID_query)
48 # print()
49 ## Submit hourly forecast
50 # power_values_hourly = [0]*48
51 # submit_forecast(client, categoryID="WPLANT", scheduleID=schedule_id_submitHourly, entityAssetID=entityAssetID,
52 #               powerValues=power_values_hourly, forecastType='hourly')
53 # print()
54 ## Submit daily forecast
55 # power_values_daily = [0]*120
56 # submit_forecast(client, categoryID="WPLANT", scheduleID=schedule_id_submitDaily, entityAssetID=entityAssetID,
57 #               powerValues=power_values_daily, forecastType='daily')
58 # print()
```

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

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

```
31 client = Client(wsdl=ENDPOINT_URL, transport=transport)
32
33
34 ##### Making API Calls (uncomment functions to run) #####
35
36 ## Query Categories
37 # make_request_categories(client)
38 # print()
39 #
40 ## Query Schedules
41 # make_request_schedules(client)
42 # print()
43 ## Query Entities
44 # make_request_entities(client, categoryID="SPLANT")
45 # print()
46 ## Query Forecasts
47 # make_request_forecasts(client, entityAssetID=entityAssetID, scheduleID=scheduleID_query)
48 # print()
49 ## Submit hourly forecast
50 # power_values_hourly = [0]*48
51 # submit_forecast(client, categoryID="WPLANT", scheduleID=schedule_id_submitHourly, entityAssetID=entityAssetID,
52 #               powerValues=power_values_hourly, forecastType='hourly')
53 # print()
54 ## Submit daily forecast
55 # power_values_daily = [0]*120
```

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



```
31 client = Client(wsdl=ENDPOINT_URL, transport=transport)
32
33
34 ##### Making API Calls (uncomment functions to run) #####
35
36 # Query Categories
37 make_request_categories(client)
38 print()
39
40 # Query Schedules
41 # make_request_schedules(client)
42 # print()
43 # Query Entities
44 # make_request_entities(client, categoryID="SPLANT")
45 # print()
46
47 # Query Forecasts
48 # make_request_forecasts(client, entityAssetID=entityAssetID, scheduleID=scheduleID_query)
49 # print()
50 # Submit hourly forecast
51 # power_values_hourly = [0]*48
52 # submit_forecast(client, categoryID="WPLANT", scheduleID=schedule_id_submitHourly, entityAssetID=entityAssetID,
53 #                 powerValues=power_values_hourly, forecastType='hourly')
54 # print()
55 # Submit daily forecast
56 # power_values_daily = [0]*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.

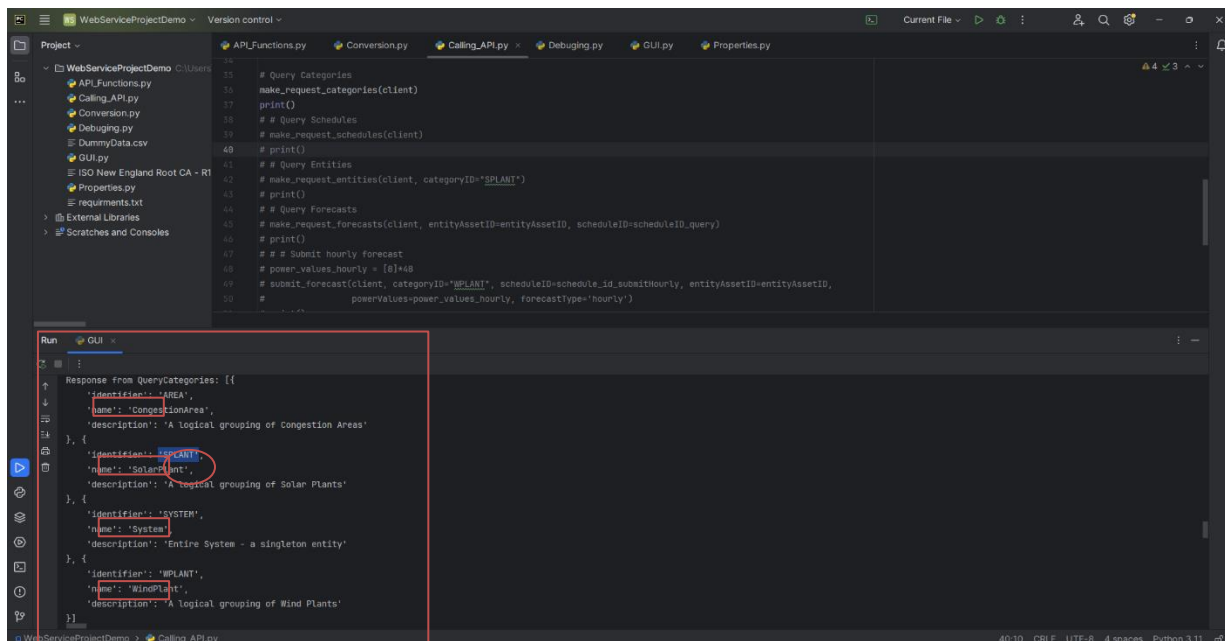


6.2.5. Select an identifier from the function output.

6.2.5.1. Highlight the selected identifier.

Example: SPLANT for solar plants.

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



6.2.6. Go to the Assigning Variables section.

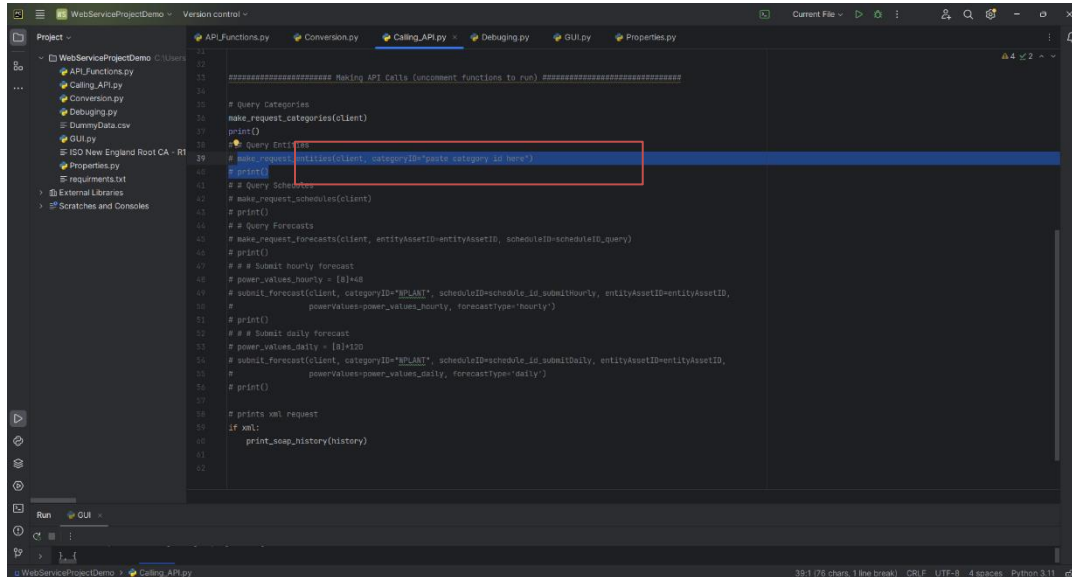
6.2.6.1. Enter the selected identifier into the categoryID variable by pasting it (“Ctrl” + “V”) into the quotation marks.

```
WebServiceProjectDemo - Version control -
API_Functions.py - Conversion.py - Calling_API.py - Debugging.py - GUI.py - Properties.py
##### Assigning Variables #####
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"

##### 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(client, categoryID=categoryID, scheduleID=scheduleID_submitHourly, entityAssetID=entityAssetID,
#                 powerValues=power_values_hourly, forecastType='hourly')
# print()
# # Submit daily forecast
# power_values_daily = [0]*120
# submit_forecast(client, categoryID="SPLANT", scheduleID=scheduleID_submitDaily, entityAssetID=entityAssetID,
```

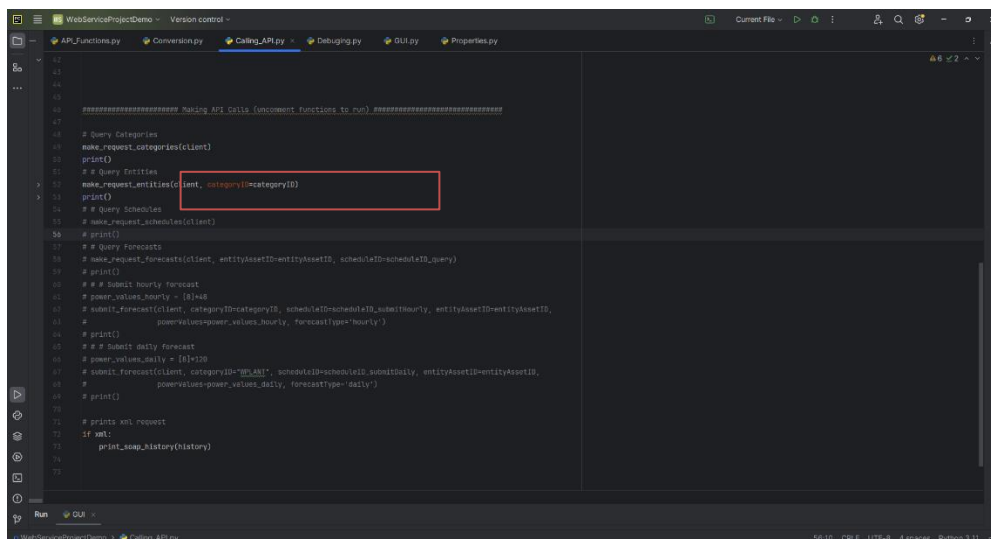
6.3. Calling `make_request_entities()`

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



```
11 ##### Making API calls (uncomment functions to run) #####
12
13 # Query Categories
14 make_request_categories(client)
15 print()
16 # Query Entities
17 make_request_entities(client, categoryID="posts", category_id="here")
18 print()
19 # Query Schedules
20 make_request_schedules(client)
21 print()
22 # Query Forecasts
23 make_request_forecasts(client, entityAssetID=entityAssetID, scheduleID=scheduleID_query)
24 print()
25 # Submit hourly forecast
26 power_values_hourly = [0]*48
27 submit_forecast(client, categoryID="MPLANT", scheduleID=scheduleID_submitHourly, entityAssetID=entityAssetID,
28                 powerValues=power_values_hourly, forecastType="hourly")
29 print()
30 # Submit daily forecast
31 power_values_daily = [0]*120
32 submit_forecast(client, categoryID="MPLANT", scheduleID=scheduleID_submitDaily, entityAssetID=entityAssetID,
33                 powerValues=power_values_daily, forecastType="daily")
34 print()
35 # Prints xml request
36 if xml:
37     print_xml_history(history)
```

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



```
11 ##### Making API calls (uncomment functions to run) #####
12
13 # Query Categories
14 make_request_categories(client)
15 print()
16 # Query Entities
17 make_request_entities(client, categoryID=categoryID)
18 print()
19 # Query Schedules
20 make_request_schedules(client)
21 print()
22 # Query Forecasts
23 make_request_forecasts(client, entityAssetID=entityAssetID, scheduleID=scheduleID_query)
24 print()
25 # Submit hourly forecast
26 power_values_hourly = [0]*48
27 submit_forecast(client, categoryID=categoryID, scheduleID=scheduleID_submitHourly, entityAssetID=entityAssetID,
28                 powerValues=power_values_hourly, forecastType="hourly")
29 print()
30 # Submit daily forecast
31 power_values_daily = [0]*120
32 submit_forecast(client, categoryID="MPLANT", scheduleID=scheduleID_submitDaily, entityAssetID=entityAssetID,
33                 powerValues=power_values_daily, forecastType="daily")
34 print()
35 # Prints xml request
36 if xml:
37     print_xml_history(history)
```

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.

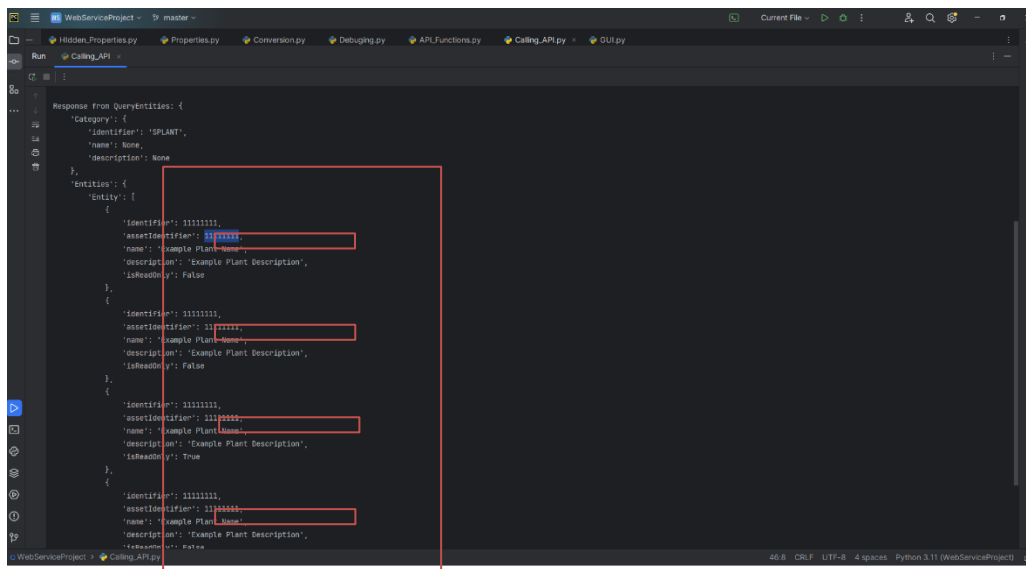
- The output should automatically pop-up, but if it doesn't, select the white play button on the left.



6.3.5. Select an assetIdentifier from the function output.

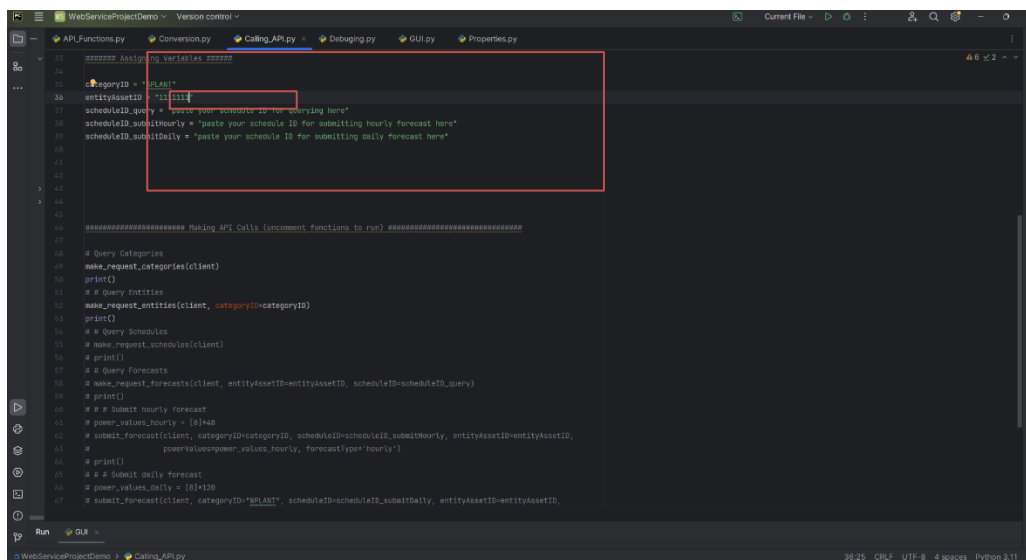
6.3.5.1. Highlight the selected assetIdentifier.

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



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.



6.4. Calling `make_request_schedules()`

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


```
39 scheduleID_submitDaily = "paste your schedule ID for submitting daily forecast here"
40
41
42
43
44
45
46 ##### Making API Calls (uncomment functions to run) #####
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 # power_values_daily = [0]*120
67 # submit_forecast(client, categoryID="MP_ABT", scheduleID=scheduleID_submitDaily, entityAssetID=entityAssetID,
68 #                 powerValues=power_values_daily, forecastType='daily')
69 # print()
70 # prints xml request
71 if xml:
72     print_soap_history(history)
```

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

```
39 scheduleID_submitDaily = "paste your schedule ID for submitting daily forecast here"
40
41
42
43
44
45
46 ##### Making API Calls (uncomment functions to run) #####
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 # power_values_daily = [0]*120
67 # submit_forecast(client, categoryID="MP_ABT", scheduleID=scheduleID_submitDaily, entityAssetID=entityAssetID,
68 #                 powerValues=power_values_daily, forecastType='daily')
69 # print()
70 # prints xml request
71 if xml:
72     print_soap_history(history)
```

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.

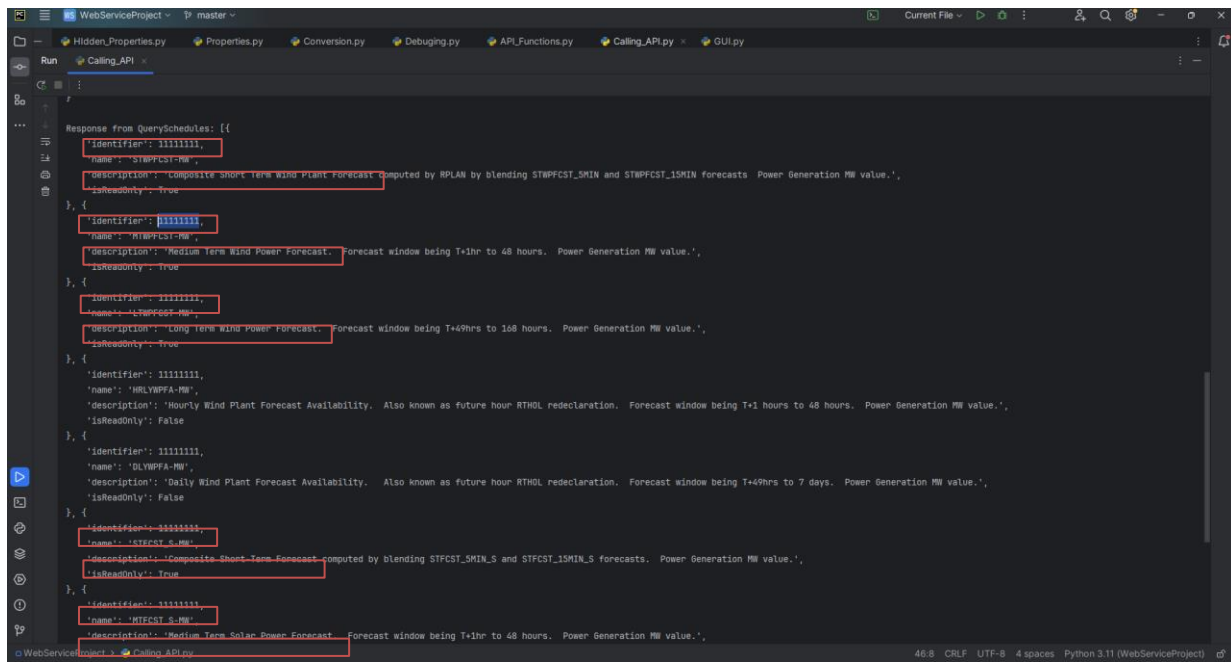


6.4.5. Select an identifier from the function output.

- 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.

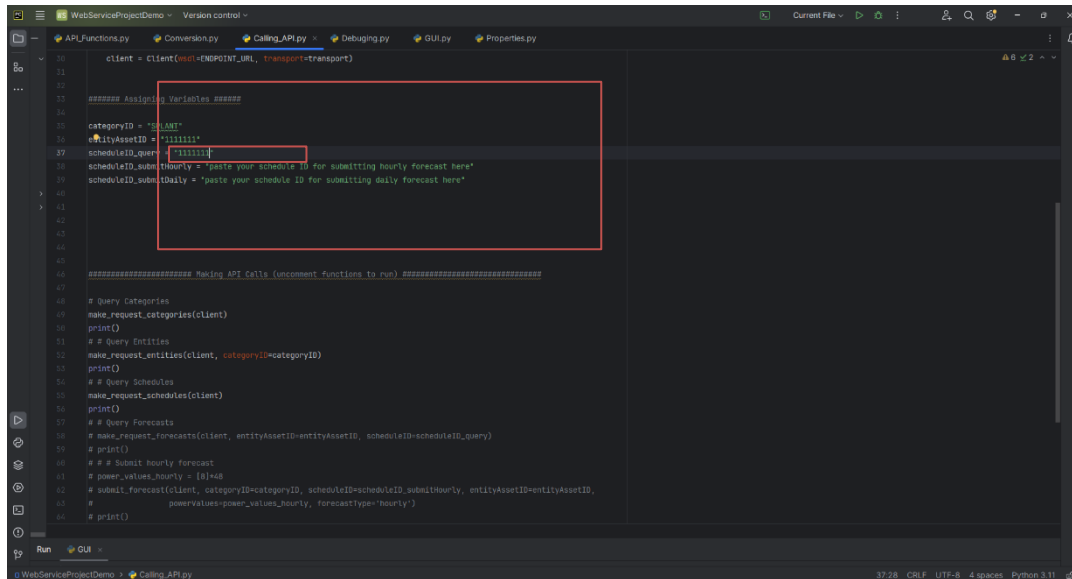
6.4.5.2. Click “Ctrl” + “C” to copy the identifier.



```
Response from QuerySchedules: [{
  "identifier": "11111111",
  "name": "STWPFCS1-MW",
  "description": "Composite Short Term Wind Plant Forecast computed by RPLAN by blending STWPFCS1_5MIN and STWPFCS1_15MIN forecasts. Power Generation MW value.",
  "isReadOnly": true
}, {
  "identifier": "11111111",
  "name": "MTWPFCS1-MW",
  "description": "Medium Term Wind Power Forecast. Forecast window being T+1hr to 48 hours. Power Generation MW value.",
  "isReadOnly": true
}, {
  "identifier": "11111111",
  "name": "LTWPFCS1-MW",
  "description": "Long Term Wind Power Forecast. Forecast window being T+49hrs to 168 hours. Power Generation MW value.",
  "isReadOnly": true
}, {
  "identifier": "11111111",
  "name": "HRLVWFA-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": "DLVWFA-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": "STWPFCS1-MW",
  "description": "Composite Short Term Forecast computed by blending STWPFCS1_5MIN and STWPFCS1_15MIN forecasts. Power Generation MW value.",
  "isReadOnly": true
}, {
  "identifier": "11111111",
  "name": "MTWPFCS1-MW",
  "description": "Medium Term Solar Power Forecast. Forecast window being T+1hr to 48 hours. Power Generation MW value.",
  "isReadOnly": true
}
```

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.



```
30 client = Client(BASEPOINT_URL, transport=transport)
31
32 ##### Assigning Variables #####
33
34 categoryID = "SOLAR"
35 entityAssetID = "111111"
36
37 scheduleID_query = "111111"
38 scheduleID_submitHourly = "paste your schedule ID for submitting hourly forecast here"
39 scheduleID_submitDaily = "paste your schedule ID for submitting daily forecast here"
40
41
42
43
44 ##### Making API Calls (uncomment functions to run) #####
45
46 # Query Categories
47 make_request_categories(client)
48 print()
49 # # Query Entities
50 make_request_entities(client, categoryID=categoryID)
51 print()
52 # # Query Schedules
53 make_request_schedules(client)
54 print()
55 # # Query Forecasts
56 make_request_forecasts(client, entityAssetID=entityAssetID, scheduleID=scheduleID_query)
57 # print()
58 # # Submit Hourly Forecast
59 # power_values_hourly = [0]*48
60 # submit_forecast(client, categoryID=categoryID, scheduleID=scheduleID_submitHourly, entityAssetID=entityAssetID,
61 #                 powerValues=power_values_hourly, forecastType="hourly")
62 # 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': '11111111',
    'name': 'STPFCST-MW',
    'description': 'Composite Short Term Wind Plant Forecast computed by BPA by blending STPFCST_SSHN and STPFCST_ISNM forecasts. Power Generation MW value.',
    'isReadOnly': True
  },
  {
    'identifier': '11111111',
    'name': 'WTRPFCST-MW',
    'description': 'Medium Term Wind Power Forecast. Forecast window being 1+hrs to 48 hours. Power Generation MW value.',
    'isReadOnly': True
  },
  {
    'identifier': '11111111',
    'name': 'LTPFCST-MW',
    'description': 'Long Term Wind Power Forecast. Forecast window being 1+0hrs to 168 hours. Power Generation MW value.',
    'isReadOnly': True
  },
  {
    'identifier': '11111111',
    'name': 'HRLWMPFA-MW',
    'description': 'Hourly Wind Plant Forecast Availability. Also known as future hour RTRM redclaration. Forecast window being 1+1 hours to 48 hours. Power Generation MW value.',
    'isReadOnly': False
  },
  {
    'identifier': '11111111',
    'name': 'DLWMPFA-MW',
    'description': 'Daily Wind Plant Forecast Availability. Also known as future hour RTRM redclaration. Forecast window being 1+0hrs to 7 days. Power Generation MW value.',
    'isReadOnly': False
  },
  {
    'identifier': '11111111',
    'name': 'STPFCST-S-MW',
    'description': 'Composite Short-Term Forecast computed by blending STPFCST_SSHN_S and STPFCST_ISNM_S forecasts. Power Generation MW value.',
    'isReadOnly': True
  },
  {
    'identifier': '11111111',
    'name': 'WTRPFCST-S-MW',
    'description': 'Medium Term Solar Power Forecast. Forecast window being 1+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.

```

client = Client(endpoint=ENDPOINT_URL, transport=transport)

##### Assigning Variables #####
categoryID = "SOLAR"
entityAssetID = "11111111"
scheduleID_query = "11111111"
scheduleID_submitHourly = "11111111"
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 = [0]*48
# submit_forecast(client, categoryID=categoryID, scheduleID=scheduleID_submitHourly, entityAssetID=entityAssetID,
#                 powerValues=power_values_hourly, forecastType='hourly')
# print()
# # # Submit daily forecast

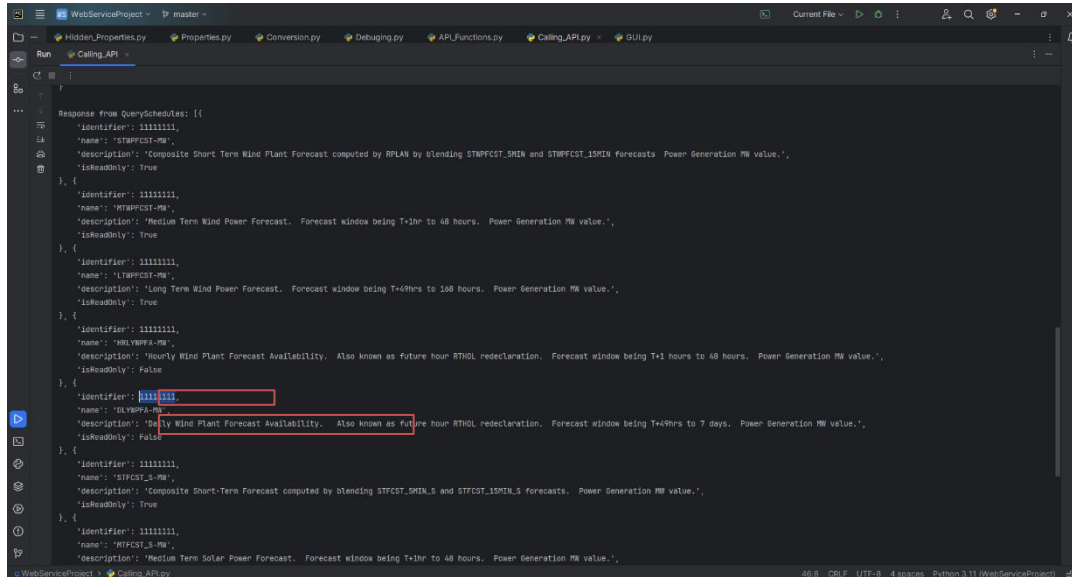
```

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

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

6.4.9.1. Highlight the selected identifier.

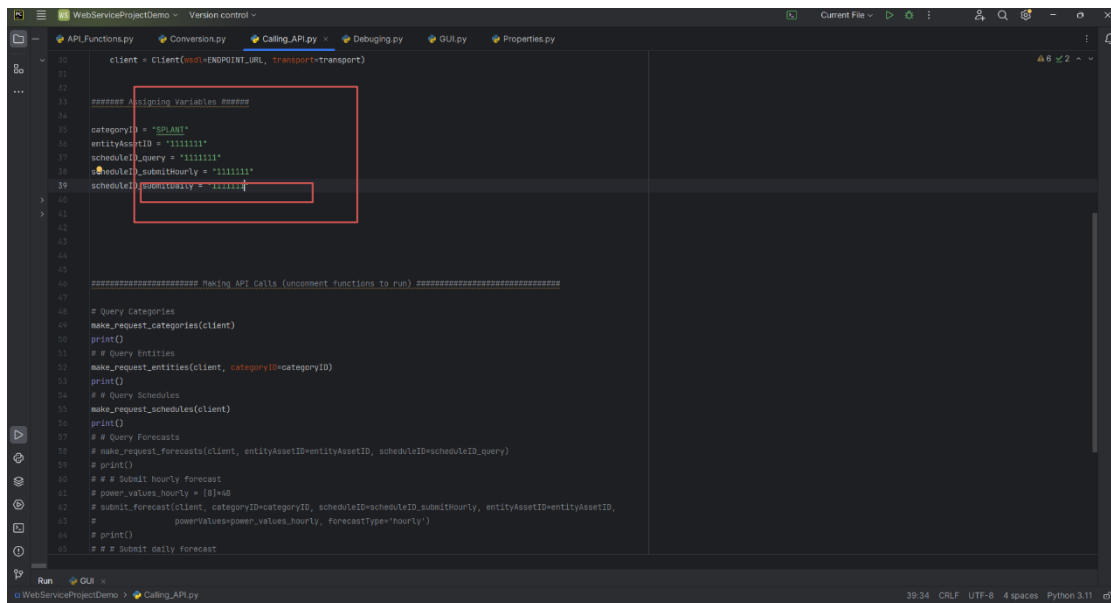
6.4.9.2. Click “Ctrl” + “C” to copy the identifier.



```
Response from QuerySchedules: [{
  "identifier": "11111111",
  "name": "STPFCST-PM",
  "description": "Composite Short Term Wind Plant Forecast computed by RPLAN by blending STPFCST_5MIN and STPFCST_15MIN forecasts. Power Generation MW value.",
  "isReadOnly": true
}, {
  "identifier": "11111111",
  "name": "MTBPPCST-PM",
  "description": "Medium Term Wind Power Forecast. Forecast window being T+3hr to 48 hours. Power Generation MW value.",
  "isReadOnly": true
}, {
  "identifier": "11111111",
  "name": "LTBPPCST-PM",
  "description": "Long Term Wind Power Forecast. Forecast window being T+49hrs to 168 hours. Power Generation MW value.",
  "isReadOnly": true
}, {
  "identifier": "11111111",
  "name": "HOLWPPFA-PM",
  "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": "DLWPPFA-PM",
  "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": "STPFCST-S-M",
  "description": "Composite Short-Term Forecast computed by blending STPFCST_5MIN.S and STPFCST_15MIN.S forecasts. Power Generation MW value.",
  "isReadOnly": true
}, {
  "identifier": "11111111",
  "name": "MTFCST-S-M",
  "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.

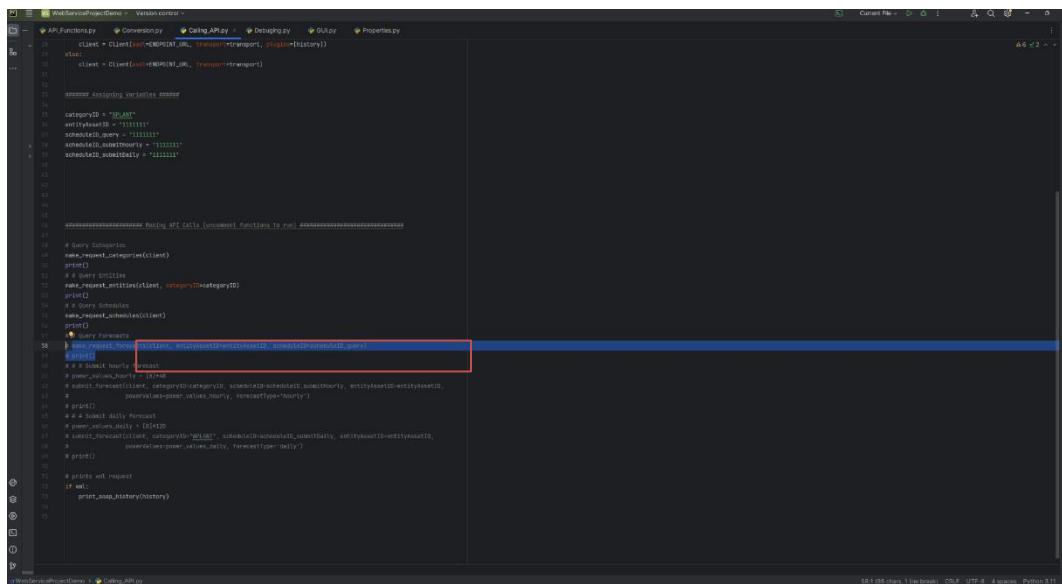
6.4.10.1. Enter the selected identifier into the scheduleID_submitDaily variable by pasting it (“Ctrl” + “V”) into the quotation marks.



```
10 client = Client(baseUrl=ENDPOINT_URL, transport=transport)
11
12
13 ##### Assigning Variables #####
14
15 categoryID = "SOLANI"
16 entityAssetID = "11111111"
17 scheduleID_query = "11111111"
18 scheduleID_submitHourly = "11111111"
19 scheduleID_submitDaily = "11111111"
20
21
22 ##### Making API Calls (uncomment functions to run) #####
23
24 # Query Categories
25 make_request_categories(client)
26 print()
27 # Query Entities
28 make_request_entities(client, categoryID=categoryID)
29 print()
30 # Query Schedules
31 make_request_schedules(client)
32 print()
33 # Query Forecasts
34 make_request_forecasts(client, entityAssetID=entityAssetID, scheduleID=scheduleID_query)
35 print()
36 # Submit hourly forecast
37 power_values_hourly = [0]*40
38 # Submit forecast(client, categoryID=categoryID, scheduleID=scheduleID_submitHourly, entityAssetID=entityAssetID,
39 # powerValues=power_values_hourly, forecastType='hourly')
40 # print()
41 # Submit daily forecast
```

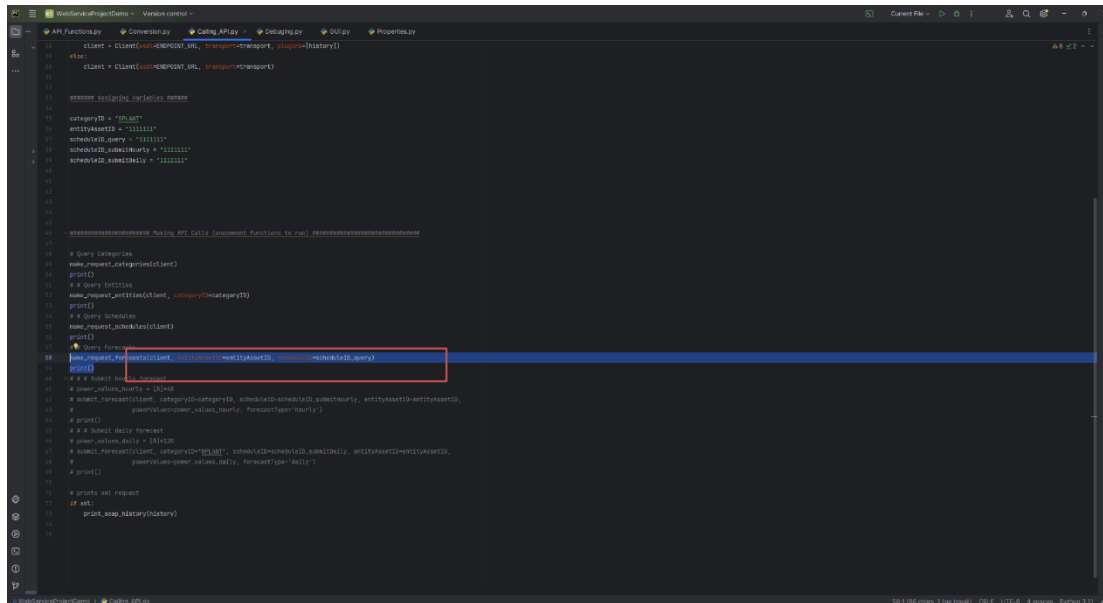
6.5. Calling [make request forecasts\(\)](#)

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



```
10 client = Client(baseUrl=ENDPOINT_URL, transport=transport)
11
12
13 ##### Assigning Variables #####
14
15 categoryID = "SOLANI"
16 entityAssetID = "11111111"
17 scheduleID_query = "11111111"
18 scheduleID_submitHourly = "11111111"
19 scheduleID_submitDaily = "11111111"
20
21
22 ##### Making API Calls (uncomment functions to run) #####
23
24 # Query Categories
25 make_request_categories(client)
26 print()
27 # Query Entities
28 make_request_entities(client, categoryID=categoryID)
29 print()
30 # Query Schedules
31 make_request_schedules(client)
32 print()
33 # Query Forecasts
34 make_request_forecasts(client, entityAssetID=entityAssetID, scheduleID=scheduleID_query)
35 print()
36 # Submit hourly forecast
37 power_values_hourly = [0]*40
38 # Submit forecast(client, categoryID=categoryID, scheduleID=scheduleID_submitHourly, entityAssetID=entityAssetID,
39 # powerValues=power_values_hourly, forecastType='hourly')
40 # print()
41 # Submit daily forecast
42 power_values_daily = [0]*10
43 # Submit forecast(client, categoryID="SOLANI", scheduleID=scheduleID_submitDaily, entityAssetID=entityAssetID,
44 # powerValues=power_values_daily, forecastType='daily')
45 # print()
46
47 # print out request
48 if not:
49     print_make_history(history)
```


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



```
1 client = Client(host=DEMOINT_URL, transport=Transport, request=Request)
2 else:
3     client = Client(host=DEMOINT_URL, transport=Transport)
4
5 # ===== Running the script =====
6
7 categoryID = "101001"
8 entityassetID = "1111111"
9 scheduleID_query = "1111111"
10 scheduleID_assetID = "1111111"
11 scheduleID_assetID = "1111111"
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
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
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

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.`

```

10 client = Client(ClientConfigURL, TransportTransport)
11
12 ##### Assigning Variables #####
13
14 categoryID = "00000"
15 entityID = "111111"
16 scheduleID_query = "111111"
17 scheduleID_submitHourly = "111111"
18 scheduleID_submitDaily = "111111"
19
20
21 ##### Uncommented Making API calls (uncomment functions to run) #####
22
23 # Query Categories
24 make_request(client)
25 # print()
26 # Query Entities
27 make_request_entity(client, categoryID=categoryID)
28 # print()
29 # Query Schedules
30 make_request_schedule(client)
31 # print()
32 # Query Forecasts
33 make_request_forecast(client, entityID=entityID, scheduleID=scheduleID_query)
34 # print()
35 ## Submit hourly forecast
36 # make_request_forecast(client, categoryID=categoryID, scheduleID=scheduleID_submitHourly, entityID=entityID,
37 #                       powerValuesHourly=powerValuesHourly, forecastType="Hourly")
38 # print()
39 # ## Submit daily forecast
40 # powerValuesDaily = [0] * 24
41 # make_request_forecast(client, categoryID="00000", scheduleID=scheduleID_submitDaily, entityID=entityID,
42 #                       powerValuesDaily=powerValuesDaily, forecastType="Daily")
43 # print()
44
45 # print out request
46 if not:
47     print_out_history(history)

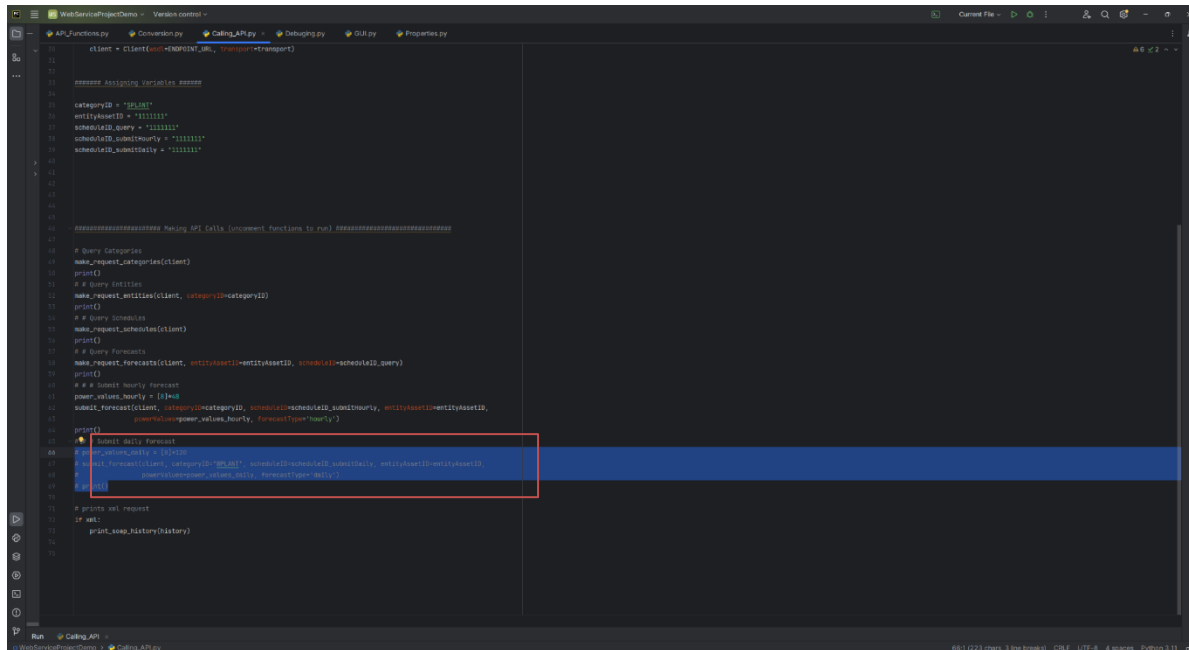
```

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.

6.6.2. Submitting the Daily Forecast.

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



```
10 client = Client(endpoint=ENDPOINT_URL, transport=transport)
11
12
13 ##### Assigning variables #####
14
15 categoryID = "BULK"
16 entityAssetID = "11111111"
17 scheduleID_query = "11111111"
18 scheduleID_submitHourly = "11111111"
19 scheduleID_submitDaily = "11111111"
20
21
22
23 ##### ===== Calling API calls (comment functions to run) ===== #####
24
25 # Query Categories
26 make_request_categories(client)
27 print()
28 # # Query Entities
29 make_request_entities(client, categoryID=categoryID)
30 print()
31 # # Query Schedules
32 make_request_schedules(client)
33 print()
34 # # Query Forecasts
35 make_request_forecasts(client, entityAssetID=entityAssetID, scheduleID=scheduleID_query)
36 print()
37 # # Submit Hourly Forecast
38 power_values_hourly = [1000]
39 submit_forecast(client, categoryID=categoryID, scheduleID=scheduleID_submitHourly, entityAssetID=entityAssetID,
40                powerValues=power_values_hourly, forecastType="hourly")
41
42 print()
43 # Submit daily forecast
44 # make_request_daily = [10000]
45 # submit_forecast(client, categoryID="BULK", scheduleID=scheduleID_submitDaily, entityAssetID=entityAssetID,
46 #                powerValues=power_values_daily, forecastType="daily")
47 # print()
48
49 # print out request
50 if not:
51     print_new_history(history)
```

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.

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.

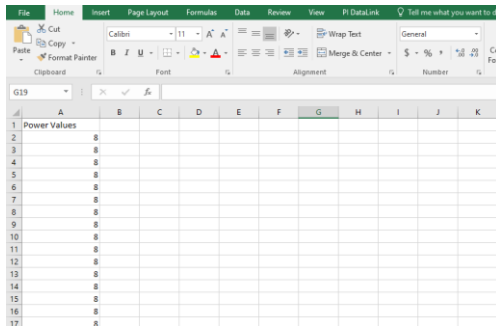
7. Getting Forecast Values

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:



The screenshot shows an Excel spreadsheet with a single column of data. The first cell in the column is labeled "Power Values". The subsequent 16 cells contain the number "8". The spreadsheet has a standard Excel interface with a ribbon at the top and a grid of cells below.

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

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

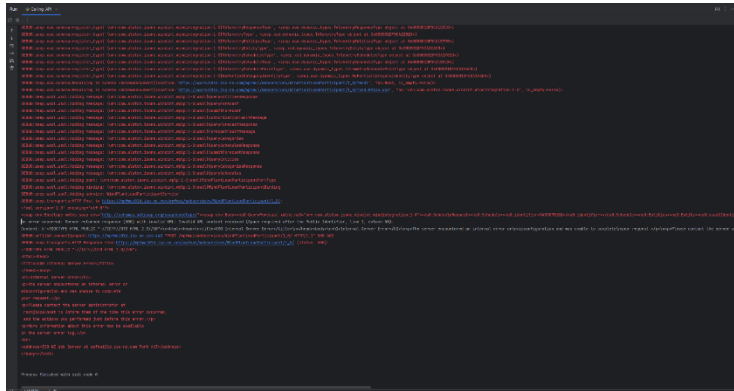
8.1. Setting up debugging

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>[REDACTED]</Identifier>
          <assetIdentifier>[REDACTED]</assetIdentifier>
          <name>[REDACTED]</name>
          <description>[REDACTED]</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 QuerySchedule: [
  {
    'Identifier': '11111111',
    'name': 'STWPFCS-MW',
    'description': 'Composite Short Term Wind Plant Forecast computed by RPLAN by blending STWPFCS_SMIN and STWPFCS_1SMIN forecasts. Power Generation MW value.',
    'isReadOnly': True
  },
  {
    'Identifier': '11111111',
    'name': 'MTWPFCS-MW',
    'description': 'Medium Term Wind Power Forecast. Forecast window being T+1hr to 48 hours. Power Generation MW value.',
    'isReadOnly': True
  },
  {
    'Identifier': '11111111',
    'name': 'LTWPFCS-MW',
    'description': 'Long Term Wind Power Forecast. Forecast window being T+4hrs to 168 hours. Power Generation MW value.',
    'isReadOnly': True
  },
  {
    'Identifier': '11111111',
    'name': 'HRLWPPA-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': 'DLVWPPA-MW',
    'description': 'Daily Wind Plant Forecast Availability. Also known as future hour RTHOL redeclaration. Forecast window being T+0hrs to 7 days. Power Generation MW value.',
    'isReadOnly': False
  },
  {
    'Identifier': '11111111',
    'name': 'STFCST-S-MW',
    'description': 'Composite Short-Term Forecast computed by blending STFCST_SMIN_S and STFCST_1SMIN_S 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': 11111111,
    'Name': 'Example Plant',
    'Description': 'Example Plant Description'
  },
  'Entities': {
    'Entity': [
      {
        'Identifier': 11111111,
        'AssetIdentifier': 11111111,
        'Name': 'Example Plant',
        'Description': 'Example Plant Description',
        'IsReadOnly': False
      },
      {
        'Identifier': 11111111,
        'AssetIdentifier': 11111111,
        'Name': 'Example Plant',
        'Description': 'Example Plant Description',
        'IsReadOnly': False
      },
      {
        'Identifier': 11111111,
        'AssetIdentifier': 11111111,
        'Name': 'Example Plant',
        'Description': 'Example Plant Description',
        'IsReadOnly': True
      },
      {
        'Identifier': 11111111,
        'AssetIdentifier': 11111111,
        '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': 'SRLVWPFA-MB',
      'Description': None,
      'IsReadOnly': None
    }
  ],
  'TimeRange': {
    'FromTime': datetime.datetime(2024, 8, 1, 15, 4, 42, tzinfo=isodate.tzinfo.Utc object at 0x0000010889407C5b),
    'ToTime': datetime.datetime(2024, 8, 8, 13, 4, 42, tzinfo=isodate.tzinfo.Utc object at 0x0000010889407C5b)
  },
  '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 0x0000010889407C5b),
            'Value': Decimal('0')
          },
          {
            'Time': datetime.datetime(2024, 8, 1, 15, 0, tzinfo=isodate.tzinfo.Utc object at 0x0000010889407C5b),
            'Value': Decimal('0')
          },
          {
            'Time': datetime.datetime(2024, 8, 1, 16, 0, tzinfo=isodate.tzinfo.Utc object at 0x0000010889407C5b),
            'Value': Decimal('0')
          }
        ]
      }
    ]
  }
}]

```

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, __format__: "%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, categoryID, scheduleID, entityassetID, powerValues, forecastType):
    url = api_endpoint + submitForecast
    # Set the current date and time to UTC
    now_utc = datetime.now(tz=timezone.utc)

    # forecast_start_time = forecast_start_time.replace(hour=0, minute=0, second=0) # Round to the top of the hour
    # forecast_type = "hourly"
    num_entries = 48
    # calculate the date and time the data from now to UTC
    forecast_start_time = now_utc + timedelta(days=-1)
    forecast_start_time = forecast_start_time.strftime("%Y-%m-%dT%H:%M:%S")
    forecast_start_time = forecast_start_time.replace(hour=0, minute=0, second=0)

    elif:
        num_entries = 120
        # calculate the date and time the data from now to UTC
        two_days_from_now_utc = now_utc + timedelta(days=2)
        # Convert the UTC time to DST
        forecast_start_time = two_days_from_now_utc.strftime("%Y-%m-%dT%H:%M:%S")
        forecast_start_time = forecast_start_time.replace(hour=0, minute=0, second=0)

    power_entries = generate_power_entries(forecast_start_time.strftime("%Y-%m-%dT%H:%M:%S"), num_entries, powerValues)

    forecast_data = {
        "forecastSchedule": {
            "categoryID": categoryID,
            "scheduleID": scheduleID,
            "entityID": entityassetID,
            "entity": {
                "name": "entityassetID",
                "power": power_entries
            }
        }
    }

    response = client.service.SubmitForecast(CreateForecastRequest(forecast_data["forecastSchedule"]))
    print("Response from SubmitForecast:", response)
    return response
except Exception as e:
    print("An error occurred: %s" % 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
```