

# AVEVA WORLD

## SAN FRANCISCO 2022

---

### AVEVAWorld 2022 Lab

### Data Views: Leveraging Cloud Data

© 2022 AVEVA Group plc and its subsidiaries. All rights reserved.

AVEVA, the AVEVA logos and AVEVA product names are trademarks or registered trademarks of aveva group plc or its subsidiaries in the United Kingdom and other countries. Other brands and products names are the trademarks of their respective companies.

AVEVA Group plc  
High Cross, Madingley Road  
Cambridge CB3 0HB, UK  
Tel +44 (0)1223 556655  
Fax +44 (0)1223 556666

---

[aveva.com](https://www.aveva.com)



## Table of Contents


1.	Introduction .....	6
1.1	Learning Objectives and Problem Statement .....	6
1.2	Data Flow .....	7
1.3	PI System software.....	8
2.	Analyzing & Configuring PI System Data in AVEVA™ Data Hub .....	9
2.1	Objective .....	9
2.2	Tasks.....	9
2.3	Approach & Details .....	9
2.3.1	Wind Turbine Overview .....	9
2.3.2	Review the Wind Farm Asset Framework Model in PI System Explorer .....	12
2.3.3	Analyzing Wind Farm Data Streamed in ADH .....	13
2.3.2.1	Logging in to ADH.....	14
2.3.2.2	Exploring Data Management Section in AVEVA™ Data Hub .....	15
2.3.2.3	Visualizing Wind Farm Data in ADH .....	19
2.3.2.4	Managing Stream Metadata in ADH .....	23
2.3.2.5	Metadata Management / Asset Management .....	25
2.3.2.6	Alternative approach .....	26
2.3.2.7	Preparing Wind Farm Data for Machine Learning Using ADH Data Views .....	27
2.3.2.8	Connecting to ADH Data View using an ADH Client.....	31
3.	Utilizing AVEVA™Data Hub Data in Machine Learning Model.....	33
3.1	Objective .....	33
3.2	Tasks.....	33
3.3	Approach & Details .....	33
3.3.1	Connecting with AVEVA Data Hub Power BI Connector .....	33
3.3.2	Using Jupyter Notebook Installed Locally on Virtual Machine .....	37
3.3.3	Connect to ADH and access the Wind Farm Data View .....	37
3.3.4	Determine variables that affect Wind Farm Active Power .....	40
3.3.5	Cleanse and prepare a Wind Farm dataset for machine learning .....	42
2.3.2.9	Generating Wind Turbine Power Curve & Identifying Outlier Data Regions .....	42
2.3.2.10	Removing Turbine GE 05.....	43
2.3.2.11	Removing negative Values .....	43
2.3.2.12	Exclude bad performing areas .....	43

2.3.2.13	Use normal operating Sate only.....	44
2.3.2.14	Plot Results.....	45
3.3.6	Train & evaluate a selected machine learning algorithm .....	46
3.3.7	Test trained machine learning model with sample input .....	48
4.	Getting Forecasted Weather Data into ML Model .....	50
4.1	Objective .....	50
4.2	Tasks.....	50
4.3	Approach & Details .....	50
4.3.1	Connect and retrieve forecasted weather data via the Open Weather API.....	50
4.3.2	Utilize retrieved weather data to predict Active Power from ML model .....	51
5.	Sending External Data Back to ADH.....	53
5.1	Objective .....	53
5.2	Tasks.....	53
5.3	Approach & Details .....	53
5.3.1	Creating a new SDS Type.....	54
2.3.2.15	Creating New SDS Type Using the User Interface.....	55
2.3.2.16	Creating New SDS Type Programmatically using ADH Python Library .....	56
5.3.2	Creating a New SDS Stream .....	57
2.3.2.17	Creating the New Stream Using the User Interface.....	57
2.3.2.18	Creating the New Stream Programmatically using ADH Python Library .....	58
5.3.3	Sending Data to New SDS Stream .....	58
5.3.4	Analyzing the New SDS Stream Data in ADH .....	59
6.	Appendix A: Streaming Data to ADH from PI Server .....	61
6.1	AVEVA™ PI System Connections .....	61
6.2	Installing the PI to AVEVA™Data Hub Agent.....	62
6.2.1	Step-by Step Installation Process.....	62
6.3	Creating a Data Transfer .....	65
7.	Appendix B: Creating an asset rule .....	67
7.1	Creating a single asset.....	67
7.2	Elevating it to Asset Type.....	71
7.3	Applying the asset type to all turbines .....	72
8.	Appendix C: Configuring a New ADH Client.....	79
9.	Appendix D: Creating an OMF Connection in ADH .....	81

10. Appendix E: References ..... 83

# 1. Introduction

## a. Learning Objectives and Problem Statement

 Video	<i>Before reading this section, please refer to the following course YouTube video: <a href="https://youtu.be/DhR8S90X4nw">https://youtu.be/DhR8S90X4nw</a></i>
--	---

AVEVA™ Data Hub (ADH) is a cloud-native platform built for historical, real-time and future/forecasted operational data. AVEVA™ Data Hub complements existing on-premise AVEVA™ PI Systems and enables users to easily define, visualize, query and shape data sets required for data science. This course will show users how AVEVA™ Data Hub extends AVEVA™ PI infrastructure to support enterprise-wide advanced analytics and machine learning.

In this course, we will be predicting the performance of a wind farm's ability to generate power. We will use a predictive machine learning model to forecast power generation. This capability is essential for every wind operator as having a reliable forecast helps them manage the portfolio of power generation resources in their fleet in order to meet the forecasted demand.

We have collected performance data from 10 wind turbines. Our first step is to evaluate the turbine data stored in AVEVA™ PI System via PI System Explorer (PSE). We will then stream the necessary data to an AVEVA™ Data Hub namespace via the AVEVA™ PI to Data Hub Agent from the PI Data Archive.

Next, we will then view the newly streamed data in ADH, add metadata in order to contextualize the streams, trend the data, as well as create a Data View.

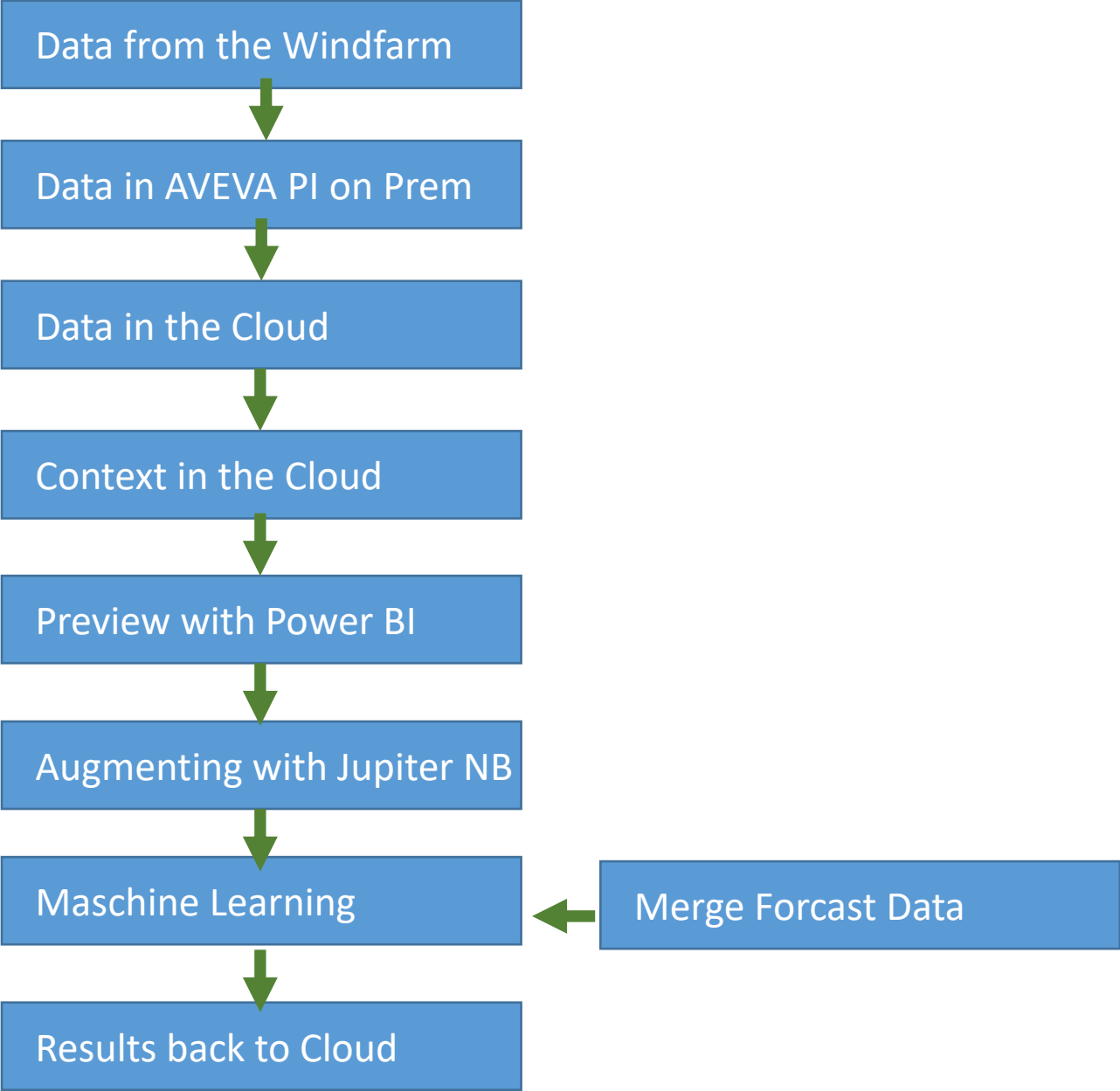
Using Power BI will have a fist quick glance at the data to see if there is some obvious data cleansing that might be needed.

We will then connect to and visualize this dataset in a Jupyter notebook using Python. After cleansing and filtering our dataset, we then use this filtered data to train, test and evaluate a model using a predictive machine learning algorithm.

The final step is to operationalize our predictive model by testing it against historical and forecasted weather data, and finally sending back the forecasted values to AVEVA™ Data Hub.

**b. Data Flow**

The data flow follows that pattern:



### c. PI System software

The VM (virtual machine) used for this course has the following PI System software installed:

Software	Version
AVEVA™ PI Data Archive	2018 SP3
AVEVA™ PI Asset Framework (PI AF) server	2018 SP3
AVEVA™ PI Asset Framework (PI AF) client (PI System Explorer)	2018 SP3
AVEVA™ PI Analysis & PI Notifications Services	2018 SP3
VEVA™ PI Vision	2019
AVEVA™ Data Hub Power BI Connector	2021

For details on PI System software, please see <http://www.osisoft.com/pi-system/pi-capabilities/product-list/>



## 2. Analyzing & Configuring PI System Data in AVEVA™ Data Hub


### a. Objective

The objective of this section is to review the Wind Farm Asset Framework (AF) Model in AVEVA™ PI System Explorer, then log in to AVEVA™ Data Hub to view the streamed data and explore all of the features and functionalities. Finally, an Data Hub Data View and Data Hub Client are created, in order to access this wind farm dataset programmatically from an external application.

### b. Tasks

1. Explore the Wind Farm Asset Framework Data in AVEVA™ PI System Explorer
2. Log in to AVEVA™ Data Hub and review the features and functionalities
3. Analyze Wind Farm data streams in the AVEVA™ Data Hub Sequential Data Store
4. Visualize Wind Farm data streams in the AVEVA™ Data Hub Trend page
5. Add & manage metadata for Wind Farm data streams in AVEVA™ Data Hub
6. Configure an AVEVA™ Data Hub Data View from Wind Farm data streams & metadata
7. Create an AVEVA™ Data Hub Client for access to AVEVA Data Hub from an external applications

### c. Approach & Details

 Video	<i>Before reading this section, please refer to the following course YouTube video: <a href="https://youtu.be/PO6yNFfOItM">https://youtu.be/PO6yNFfOItM</a></i>
--	---

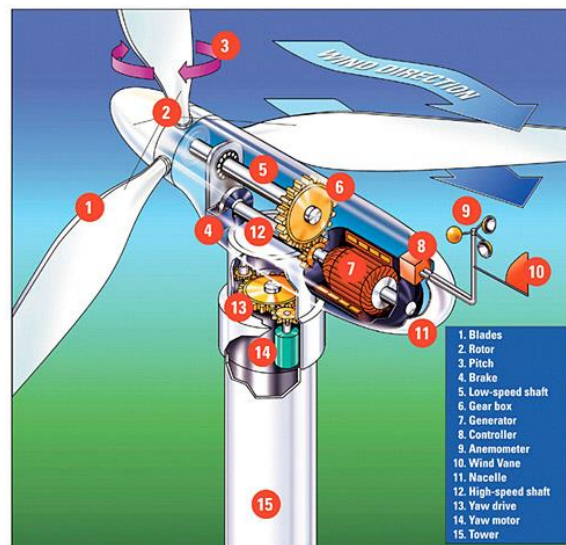
#### i. Wind Turbine Overview

Wind has kinetic energy that can be converted into electrical energy through a wind turbine. These turbines operate by capturing wind energy to turn rotor blades that run a generator. The rotors and turbines are governed by a rule known as Betz's Law, which states that a turbine may only capture 59.3% of the energy from the wind. Betz's limit refers to the fact that if more than this amount of wind energy is captured, a "build-up" in front of the rotor creates a limiting factor for future wind capture. The energy in the wind grows per windspeed<sup>3</sup>. Meaning that double the wind speed you have eight times more power. The nominal power of the generator sets the upper limit to the output so that the windspeed<sup>3</sup>

curve flattens out once nominal output has been reached. Oversizing the generator to rise that limit makes economically little sense, as very high wind conditions happen only a small fraction of the year.



Wind turbines are comprised of several main parts, which include the foundation, tower, nacelle, rotor, and transformer. The foundation is the base of the turbine that keeps the structure secured to the ground. The tower is the largest and most visual aspect of the structure. The tower size will vary based on the size of the turbine, required height to achieve optimal wind input, and size of the rotor. The nacelle is an important structure that connects the tower to the generator and eventually the rotor. The rotor is comprised of blades that ultimately capture the wind and run the turbine. Lastly, the transformer adapts the voltage produced to the grids needs. A controller inside the base of the tower ensures all aspects of the turbine are running properly, making it an autonomous generating asset.



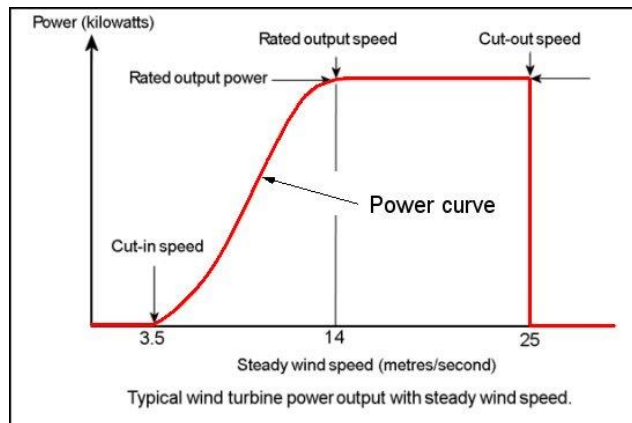
**Figure 1: Wind Turbine Cross Section**

Most of the inner workings of the turbine are located inside of the nacelle. The rotor is attached to the nacelle by way of the main shaft. As the wind turns the rotor, it turns the large main shaft, which is attached to a gearbox. The gearbox is used to turn the low rotations per minute (RPMs) of the rotor into a much higher number of rotations to generate electricity efficiently. This ratio will vary depending on the size of the turbine. As an example, the gearbox in an industrial sized turbine could turn 22 RPMs of the

rotor into around 1500 revolutions per minute. In newer turbines adaption to grid frequency is done by electronic means offering the generator to operate at different non-constant speeds.

The next essential part connected to the gear box is the yaw bearing. This is a bearing mounted right at the top of the tower. A large yaw wheel fits into the bearing and turns the nacelle and rotor in the direction of the wind when the yaw motor is engaged. A large generator is then attached to the gearbox. The current from the generator is sent down through the tower and into the transformer through large electrical cables. In addition to the large controller at the base of the tower, there is a smaller controller in the nacelle that will allow the rotor to start when the anemometer reads that there is enough wind. The anemometer is a small wind meter that will read the wind speed. The wind vane attached to the top of the anemometer will read out the dominant direction, causing the yaw motor and wheel to turn accordingly. Lastly in the nacelle, there is a mechanical brake that will allow the rotor to be stopped if mechanical repairs are necessary. For more information and references, please refer to Appendix D.

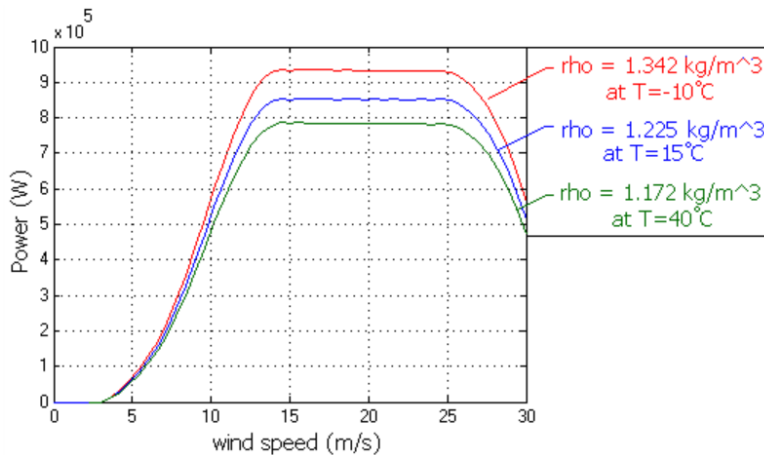
A typical wind turbine power curve is shown in Figure 2, which plots Active Power (in kW) vs. Wind Speed (in meters/second):



**Figure 2: Typical Wind Turbine Active Power vs. Wind Speed**

The Cut-in Speed is the wind speed value which causes the active power to begin rising, while the Cut-out Speed is the wind speed which causes the active power to begin dropping. The Rated Speed is the wind speed which causes the active power to plateau, while the Rated Power is the corresponding active power value at the rated speed. This graph is true for a fixed air temperature.

Changes in air temperature will cause this power curve to shift, as per Figure 3 below:




**Figure 3: Effect of Air Temperature on Typical Wind Turbine Power Curve**

Air temperature variations shift the power curve up at lower temperatures and shift it down at higher temperatures. Rho is the air density, which is affected by air temperature, humidity and altitude of the installation.

The wind turbine power curve will become important in Section 3, when we prepare our wind farm dataset for machine learning, in order to predict active power from a given wind speed and air temperature.

## ii. Review the Wind Farm Asset Framework Model in PI System Explorer

 Video	<i>Before reading this section, please refer to the following course YouTube video: <a href="https://youtu.be/9kW4ihPqFxU">https://youtu.be/9kW4ihPqFxU</a></i>
--	---

The class will use a sample wind farm database named Wind Power which consists of a fleet level element (Wind Power Generation Fleet) with a single wind farm (Big Buffalo Wind Farm) in Amarillo, TX that has 10 turbines. Open AVEVA™ PI System Explorer (PSE) and navigate to the Wind Power AF database.

Each of the turbines is built from the same AF element template, Wind Turbine, which can be viewed in the Library under Element Templates.

Each turbine has a variety of data associated with it, including PI Point real-time values, as well as metadata such as location and manufacturer information.

We will not focus on all the AF attributes, but will be primarily concerned with Active Power, Wind Speed, Air Temperature and Turbine State data for this course.

Name	Value	Time Stamp
<b>Category: Control</b>		
Turbine State	Load Operation	2/3/2020 7:44:10 PM
<b>Category: Electrical Measurements</b>		
Active Power	280.5 kW	2/3/2020 7:44:10 PM
Active Power Predicted	Pt Created	1/30/2020 9:21:40 PM
<b>Category: Generator</b>		
Active Power	280.5 kW	2/3/2020 7:44:10 PM
Active Power Predicted	Pt Created	1/30/2020 9:21:40 PM
<b>Category: Identification</b>		
Manufacturer	Truvalle	1/1/1970 12:00:00 AM
Model	T95-2MW	1/1/1970 12:00:00 AM
Rated Power	1500 kW	1/1/1970 12:00:00 AM
<b>Category: Location</b>		
Latitude	44.011	1/1/1970 12:00:00 AM
Longitude	-101.93	1/1/1970 12:00:00 AM
<b>Category: Mechanical Measurements</b>		
Blade1, Actual Value	0.19795 °	2/3/2020 7:44:10 PM
Blade2, Actual Value	-0.040059 °	2/3/2020 7:44:10 PM
Blade3, Actual Value	-0.025244 °	2/3/2020 7:44:10 PM
Nacelle Position	334.38 °	2/3/2020 7:44:10 PM
Rotor Speed	13.661 rpm	2/3/2020 7:44:10 PM
<b>Category: Met</b>		
Air Temperature	13.654 °C	2/3/2020 7:44:10 PM
Wind Speed	6.5702 m/s	2/3/2020 7:44:10 PM

The AF attributes of interest and their corresponding PI tags are shown below (**XX** refers to the Wind Turbine number, e.g. 01 for GE01, 10 for GE10)

1. Active Power : \\PISRV01\GE**XX**.P.ACT.PV
2. Wind Speed : \\PISRV01\GE**XX**.V.WIN.PV
3. Air Temperature : \\PISRV01\GE**XX**.TGEN.COOL.PV
4. Turbine State : \\PISRV01\GE**XX**.OS.T.PV

After exploring the data for this wind farm in PSE, we want to send these tags as separate data streams to ADH next.

*Before reading the next section, please refer to the following course YouTube video: <https://youtu.be/IsQhYP7t3ho>*

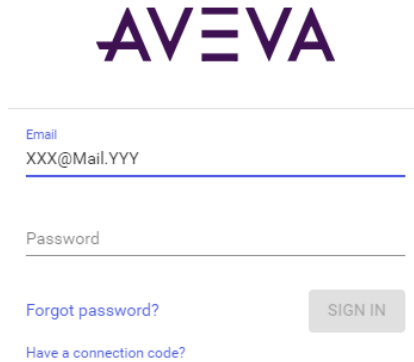
### iii. Analyzing Wind Farm Data Streamed in ADH

*Before reading this section, please refer to the following course YouTube video: <https://youtu.be/-r-osfjRdIA>*

## 1. Logging in to ADH

In a browser, navigate to <https://datahub.connect.aveva.com>

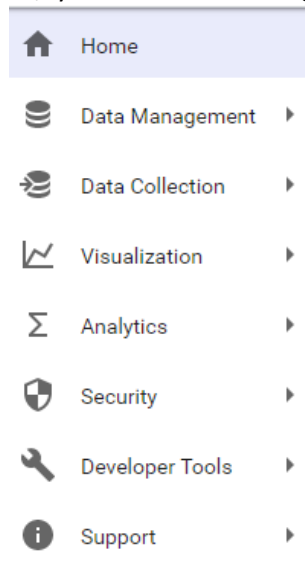
Click on the Sign In button at the top right corner, and enter your credentials.



The screenshot shows the AVEVA login interface. At the top center is the AVEVA logo in a purple, stylized font. Below the logo is a horizontal line. Underneath the line, there are two input fields: one for 'Email' containing the placeholder text 'XXX@Mail.YYY' and one for 'Password'. To the left of the password field is a link for 'Forgot password?'. To the right of the password field is a grey 'SIGN IN' button. Below the password field is another link: 'Have a connection code?'.

On the login page, please enter the email address and password that was chosen for the AVEVA Data Hub and click on the Sign In button.

Once logged in to AVEVA™ Data Hub, you will see the Navigation menu in the portal.



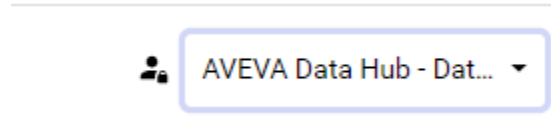
We will begin with a brief tour of the AVEVA™ Data Hub Portal, specifically the Data Management section, in order to familiarize ourselves with the features and functionalities available to subscribers.

The wind farm data streamed to AVEVA™ Data Hub is housed in a *Namespace* which has a *Connection* to an AVEVA™ PI Server, and the individual tags are stored as data streams in the *Sequential Data Store* (SDS) under that Namespace.

## Exploring Data Management Section in AVEVA™ Data Hub

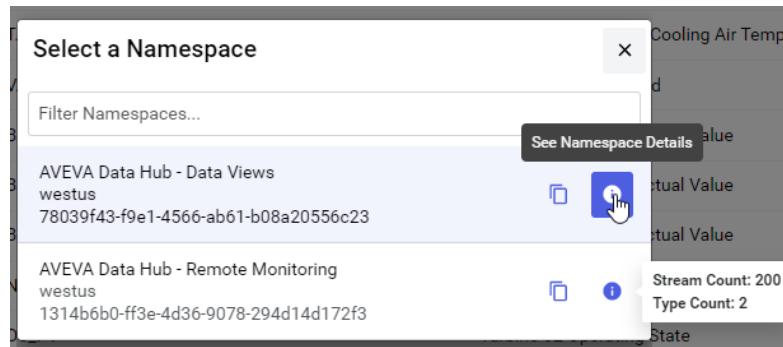
### Introduction to Namespaces

The Namespace we will be using for this course is “**AVEVA Data Hub – Data Views**”, which is available from the drop down list at the upper right part of the screen



An AVEVA™ Data Hub account is divided into one or more namespaces, which correspond to a specific collection of infrastructure data. Each namespace corresponds to an instance of SDS, and holds its own set of SDS Types, SDS Streams, and SDS Stream Views. Namespaces serve as the destination for incoming stream data.

To view namespace details, select an existing namespace, and click Display Details. The type count, and stream count refer to the number of SDS types and SDS streams, respectively, that have been created for the selected namespace.

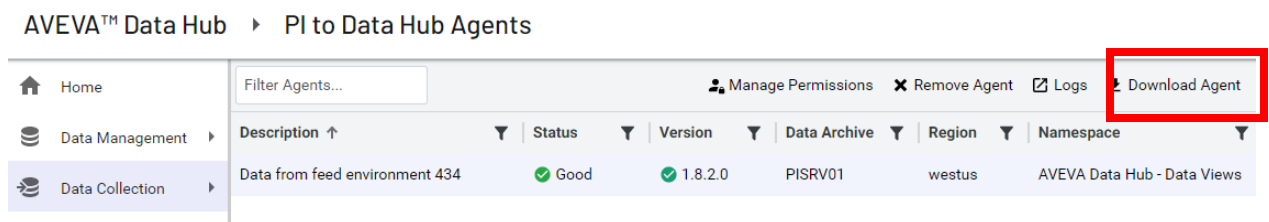


## Introduction to Connections

The specific connection that streams the wind farm data from the AVEVA™ PI System to ADH is **Data from feed environment 434**, which is already created in the **AVEVA Data Hub – Data Views** namespace. This connection was done via the AVEVA™ PI to Data Hub Agent, which has already been configured and running on a Master Training Cloud Environment (TCE) machine to stream this data to AVEVA™ Data Hub. For information regarding setting up this Agent to stream data to AVEVA™ Data Hub, please refer to Appendix A of this manual.

You can establish high-throughput data connections from a AVEVA™ PI System or any OMF-compatible data source into AVEVA™ Data Hub with the Connections feature.

If needed the agent can be downloaded under Data Collection / PI to ADH Agents



## Introduction to Sequential Data Store (SDS)


SDS is a cloud-based streaming data storage that is optimized for storing sequential data, usually time-series, but can be anything that is indexed by an ordered sequence. SDS stores these streams of events with a specific type and provides convenient ways to find and associate events.

There are 3 elements of SDS, which are SDS Types, SDS Streams and SDS Stream Views. SDS Types and SDS Streams will be presented in more detail below. SDS Stream Views will not be discussed in this course but there are OSisoft YouTube videos which explain their definition and configuration:

1. What is a Stream View? - <https://www.youtube.com/watch?v=8iTgWyc7eQ>
2. Create a Stream View: <https://www.youtube.com/watch?v=jhLqmLN0rQE>

### SDS Types

An SDS Type defines the shape of a single measured event and gives structure to data. SDS Types can define simple atomic types, such as integers, floats, strings, arrays, and dictionaries, or they can define complex types, which can be collections of simple types. In this course, we will create a complex SDS Type to store our forecasted data back in AVEVA™ Data Hub later.

 Note	<i>You cannot edit an existing type. In order to have a type with different properties than originally assigned, you must create a new type.</i>
---	--



The pre-configured PI to Data Hub Agent creates and uses seven (7) SDS Types, which are PI-Timestamp, PI-String, PI-Int32, PI-Int16, PI-Float64, PI-Float32 and PI-Digital.

Types	Search for Types...
<input type="checkbox"/> Id ↑	Name
<input type="checkbox"/> PI-Digital	PI Digital
<input type="checkbox"/> PI-Float32	PI Float32
<input type="checkbox"/> PI-Float64	PI Float64
<input type="checkbox"/> PI-Int16	PI Int16
<input type="checkbox"/> PI-Int32	PI Int32
<input type="checkbox"/> PI-String	PI String
<input type="checkbox"/> PI-Timestamp	PI Timestamp

## SDS Streams

Streams are collections of sequentially occurring values indexed by a single property, typically time series data. You define streams to organize incoming data from another system into AVEVA™ Data Hub. To define a stream, you must first define an SDS type, which defines the structure of the data you want to stream into a selected namespace. SDS Stream identifiers must be unique within a Namespace and must include a TypeId that references this SDS Type.

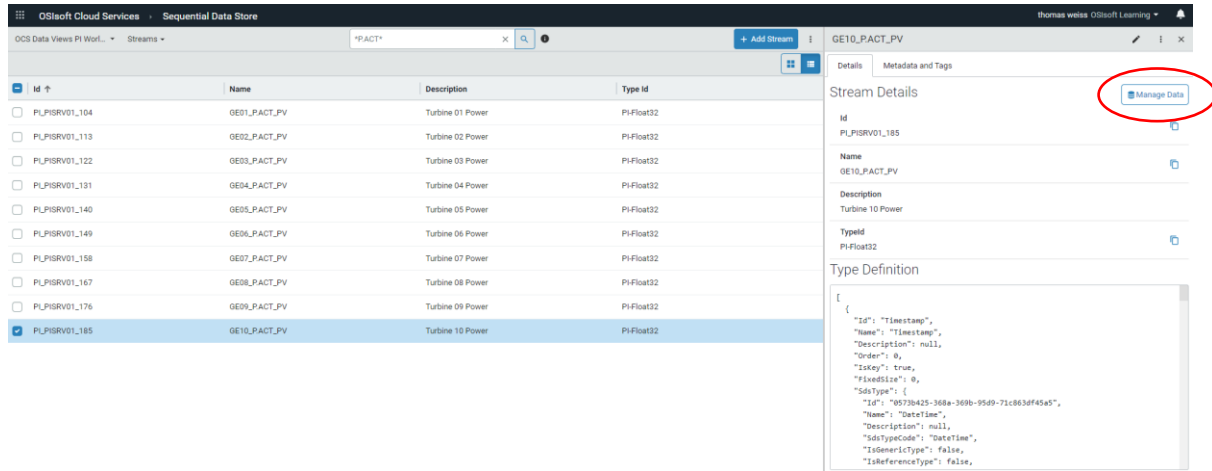
You create and write data to streams using a REST (*REpresentational State Transfer*) API (*Application Programming Interface*) or OMF (*OSIsoft Message Format*). The streams created can be used to store simple or complex types. In this course, we will create a new data stream and send values using the ADH Python library functions later.

Streams	Search for Streams...	+ Add Stream	
<input type="checkbox"/> Id ↑	Name	Description	Type Id
<input type="checkbox"/> PLPISRV01_100	GE01_BL3.ACT_PV	Turbine 01 Blade 3, Actual Value	PI-Float32
<input type="checkbox"/> PLPISRV01_101	GE01_N_ROT.PLC_PV	Turbine 01 Rotor Speed (PLC)	PI-Float32
<input type="checkbox"/> PLPISRV01_102	GE01_OS_PV	Turbine 01 Operating State	PI-Digital
<input type="checkbox"/> PLPISRV01_103	GE01_POS.NAC_PV	Turbine 01 Nacelle Position	PI-Float32
<input type="checkbox"/> PLPISRV01_104	GE01_PACT_PV	Turbine 01 Power	PI-Float32
<input type="checkbox"/> PLPISRV01_105	GE01_T_GEN.COOL_FV	Turbine 01 Generator Cooling Air Temp.	PI-Float32
<input type="checkbox"/> PLPISRV01_106	GE01_V_WIN_PV	Turbine 01 Wind Speed	PI-Float32
<input type="checkbox"/> PLPISRV01_107	GE02_BL1.ACT_PV	Turbine 02 Blade 1, Actual Value	PI-Float32
<input type="checkbox"/> PLPISRV01_108	GE02_BL2.ACT_PV	Turbine 02 Blade 2, Actual Value	PI-Float32
<input type="checkbox"/> PLPISRV01_109	GE02_BL3.ACT_PV	Turbine 02 Blade 3, Actual Value	PI-Float32

We would like to now confirm and verify that all relevant data/events for the wind farm have been sent from the PI System and are present in the SDS, as well as perform some preliminary data exploration and analysis.

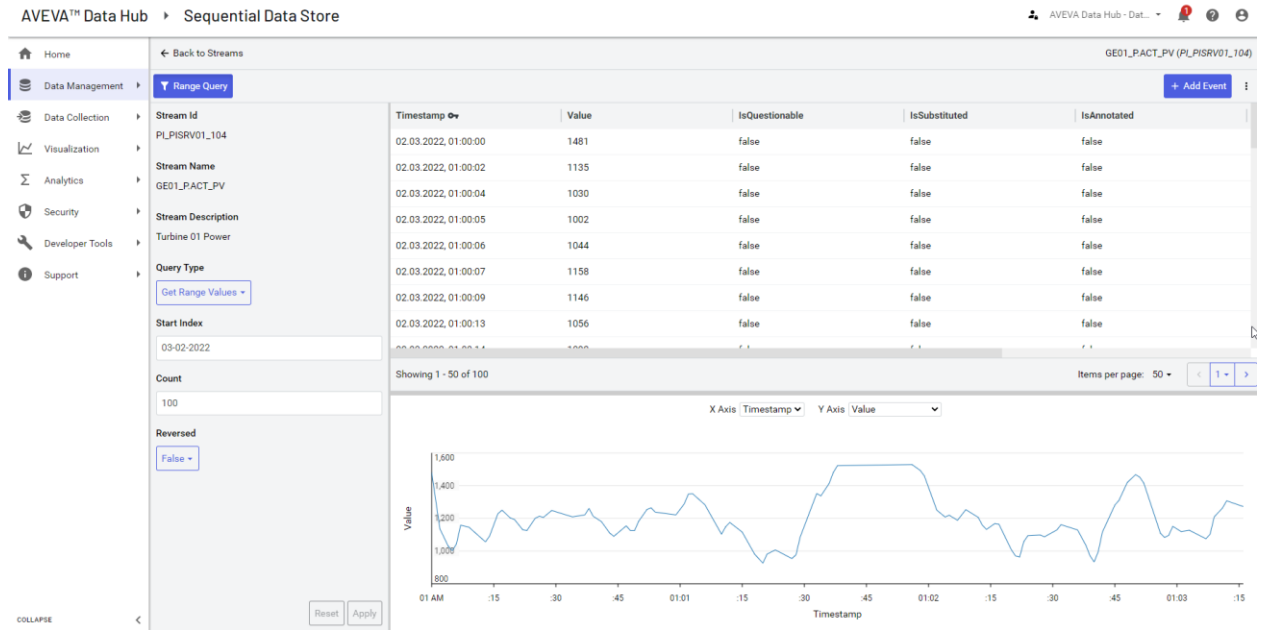
Navigate to Data Management / Sequential Data Store. On this page, you can search for all the Active Power streams. On the Search field on the Streams tab, enter \*P.ACT\*, which would result in only the Active Power data streams for all the wind farm turbines being displayed.

If one of the data streams is selected such as GE01.P.ACT.PV, a Manage Data Button appears under the Streams Detail tab.



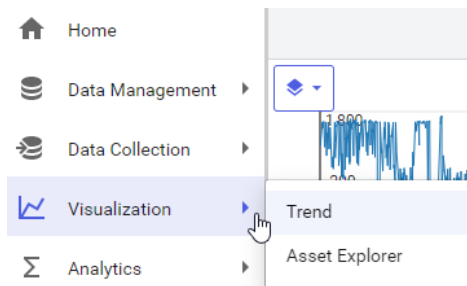
Press it to get to the next page. Set the Query type to Get Range Values. The Get Range Values query type is similar to viewing data via PI System Management Tools under Data>Archive Editor. If the Get First Value or Get Last Value query types are selected, this would display a single data point at either the beginning or the end of the data timeframe, respectively.

Enter a Start Index of 03-02-2022 and click on Apply. The results for the Active Power stream for Wind Turbine GE01 will be shown in tabular form with a 100-event count (modifiable via the Count field) with a trend shown on the on the bottom.

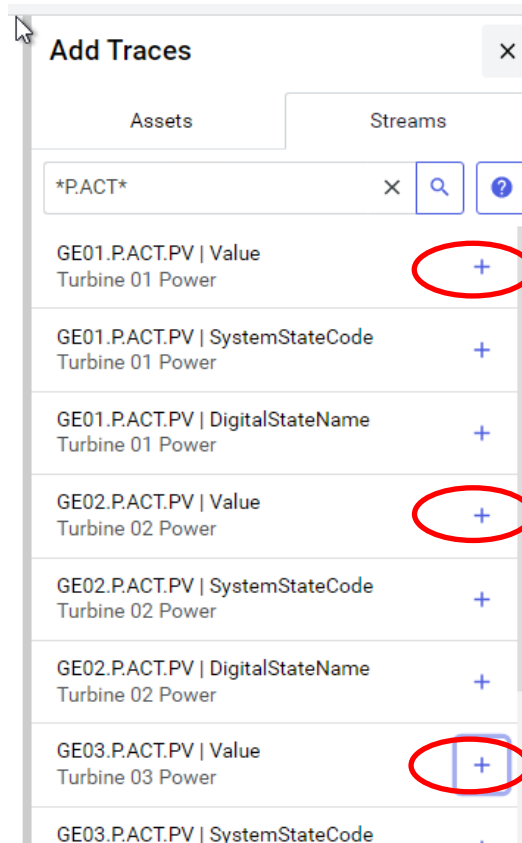


### Visualizing Wind Farm Data in ADH

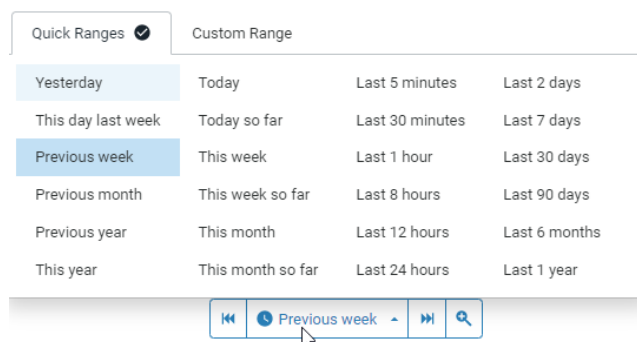
We can delve deeper into the visualization and trending of the wind farm data in the Visualization section of ADH. On the Trend page under Visualization, data streams in ADH can be trended, analyzed and overlaid for various time ranges of interest.



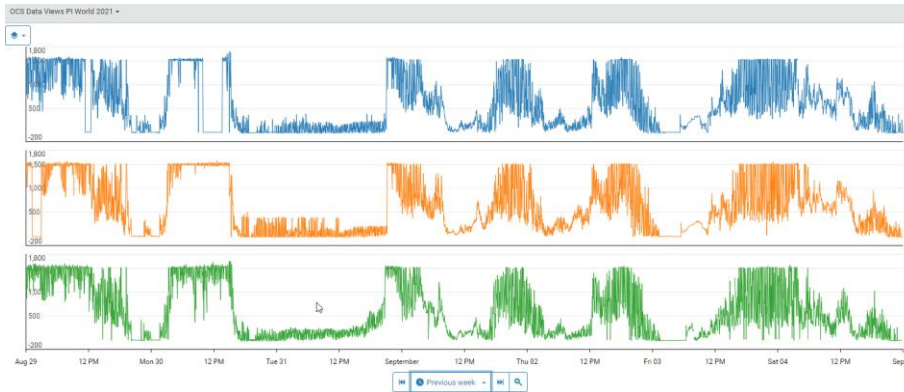
In this section, the Active Power for the wind turbines for the last week will be analyzed. In the Search query bar, enter \*P.ACT\*, and click the magnifying glass icon. This will return the Active Power for all the 10 turbines. Add the P.ACT.PV values for the first 3 turbines (GE01, GE02 & GE03) to the trend by clicking the + button to the right of the corresponding data stream.



In the Time Range field, select Previous Week from Quick Ranges.



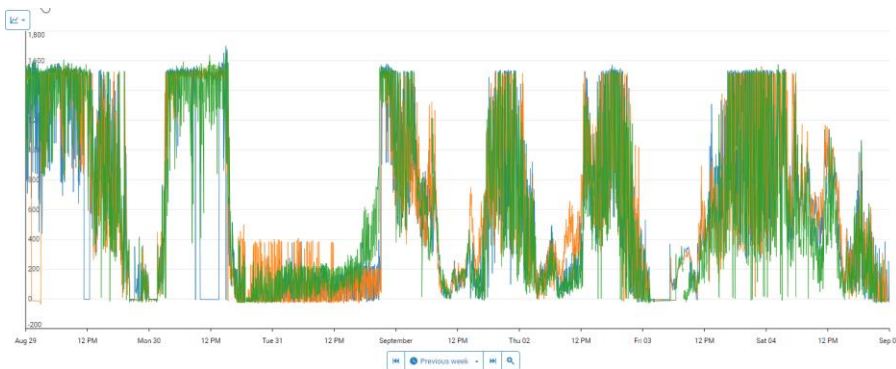
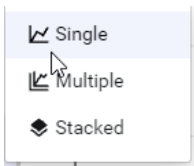
The corresponding trend should be as below:



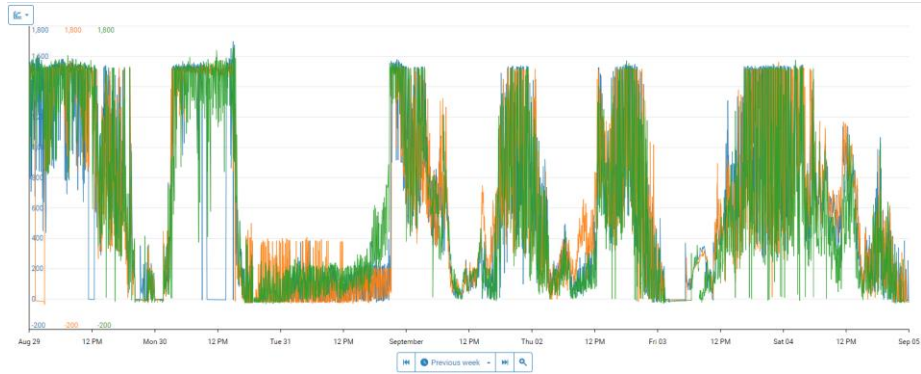
Each Active Power data stream is trended as a separate graph, but we can combine the 3 data streams into a single graph by clicking on the Layers icon at the top left corner:



Selecting Single converts the trends into a single graph with a single y-scale as below.



Selecting Multiple converts the trend into a single graph but with multiple y-scales now:

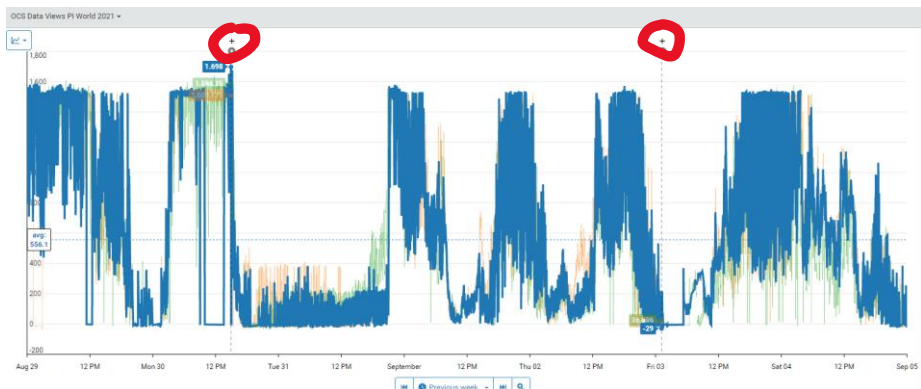


Clicking on a specific data stream in the table below the trend highlights the trend, and a trend can be added or removed by checking or unchecking the box to the left of the stream name. The last value, minimum, maximum, and average values are shown for each trace in the table, making it easy to analyze data over the entire time range in the trend.

<input checked="" type="checkbox"/>	Name	Timestamp	Value	UOM	Min	Max
GE01_PACT_PV						
<input checked="" type="checkbox"/>	Value	9/4/2021 11:56:03 PM	-9.000		-29.000	1,698.000
GE02_PACT_PV						
<input checked="" type="checkbox"/>	Value	9/4/2021 11:37:38 PM	-20.096		-37.000	1,576.000
GE03_PACT_PV						
<input checked="" type="checkbox"/>	Value	9/4/2021 11:49:22 PM	-5.351		-34.000	1,658.000


When you select (highlight) a trace, its minimum and maximum are automatically marked as well as the average for the time range.

When multiple cursors are added to the trend, the duration, delta (difference), average, minimum, and maximums between consecutive cursors are calculated and displayed, to analyze periods in-between the cursors. You need to click on the + on the cursors to activate the calculation



Timestamp	GE01_PACT_PV   Value	GE02_PACT_PV   Value	GE03_PACT_PV   Value
08/30/2021 02:56:30 PM	1,698.000	1,509.771	1,586.250
3d 10:12:02	Min: -29.000 Max: 1,698.000 Average: 400.476 Delta: -1,727.000	Min: -29.000 Max: 1,548.000 Average: 417.692 Delta: -1,495.327	Min: -34.000 Max: 1,658.000 Average: 424.663 Delta: -1,559.564
09/03/2021 01:08:32 AM	-29.000	14.445	26.686

## Managing Stream Metadata in ADH



Before reading this section, please refer to the following course YouTube video: <https://youtu.be/wijDu4hM8TA>

Stream metadata in ADH is essential in providing context and associated information for data streams. In our case, we need to provide metadata for our wind farm data streams, so that we can identify which streams are associated with a specific turbine, and what type of measurement that stream describes.

This is needed when we create an ADH Data View in the next section for the subsequent machine learning application, and is analogous to metadata inclusion for elements in PI Asset Framework in order to be used later in a PI Integrator for Business Analytics, for instance.

Under the Sequential Data Store -> Streams, after selecting at least one stream there is a Metadata and Tags tab on the right:

Details

Metadata and Tags

Stream Tags

No tags to display.




Stream Metadata

Metadata	Value
engunits	Deg. <span style="float: right;">ⓘ</span>
pointid	100 <span style="float: right;">ⓘ</span>

### Stream Tags and Stream Metadata

Stream tags in SDS are used to categorize or denote special attributes of streams and are represented as a list of strings. They are defined by specific rules.








Stream metadata is represented as a dictionary of string keys and associated string values (i.e. key-value pairs). It can be used to associate additional information with a stream. When streaming data from a PI System to ADH, all configured PI Point attributes and classes such as PointID, Point Source, EngUnits, etc. are automatically included as Stream Metadata. If a wind farm stream such as GE10.P.ACT.PV is selected from the list, its Stream Metadata is shown as below, as these values were already set when first configuring these tags in the PI Data Archive.

GE01.PACT.PV   



Details Metadata and Tags

**Stream Tags**  
No tags to display.

**Stream Metadata**

Metadata	Value	
engunits	kW	
pointid	104	
pointsource	Wind	
pointtype	Float32	
sourcetag		
step	0	
tag	GE01.PACT.PV	

Metadata for each stream can be added individually by switching to the edit mode (pencil icon)

GE01.PACT.PV  

Details Metadata and Tags

**Stream Tags**  
No tags to display

toggling the metadata pane then via the +Add Metadata button.

PI-Float32

Tags Metadata

[+ Add Metadata](#)

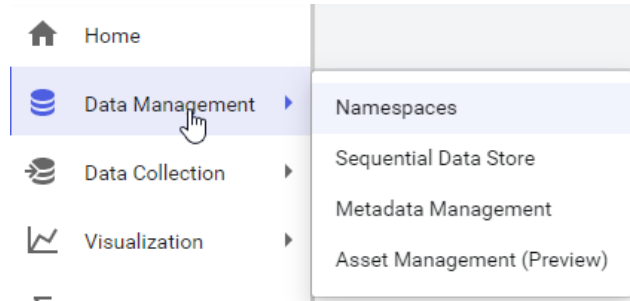
**Metadata Value**

engunits	kW	
pointid	104	
pointsource	Wind	
pointtype	Float32	
sourcetag	Metadata Value...	
step	0	
tag	GE01_PACT_PV	

If metadata for multiple streams need to be added in bulk, the Metadata Management page under the Data Management section of the Navigation menu should be used instead.



A second way is to use the Asset Management functionality which will include metadata and Asset context in one step




### *Metadata Management / Asset Management*

ADH Metadata Rules enable users to relate similar data streams for the purposes of analysis and display, and to feed into downstream applications. Metadata rules also create and assign metadata to the streams they identify in bulk. Metadata rules are created by selecting parsable metadata from stream name structures and applying the rule to identify all streams whose names match the defined pattern.

In this case, we need to create metadata rules for all the wind farm data streams in order to associate each stream with a specific turbine (i.e. GE01, GE02, etc.), as well as designate the measurement type (i.e. Active Power, Wind Speed, etc.).

Another way to get to the same result is to use an Asset Rule

 Note	<p><i>There is already an Asset Rule created for this course called <b>Turbine Enumeration</b>. Please do not create any additional Metadata / Asset Rules because every student uses the same SDS Streams, and any new Metadata Rule created will overwrite the existing rule on the shared SDS Streams, thus creating potential errors/conflicts. This is because Metadata Rules are intended to only be used by Account Administrators when initially configuring data in Namespaces.</i></p> <p><i>You can view this existing Metadata Rule configuration by selecting the Edit Metadata Rule option. The information stated below is just for reference in case this Metadata Rule needs to be recreated from scratch.</i></p>
---	---

The exact steps how to set up an asset rule are detailed in the appendix.

### Alternative approach


In the next version of the AVEVA™ PI to Data Hub Agent there will be a close link between the asset structure in AF and the way to create assets. All the steps to identify assets are significantly simplified by using this approach.

PREVIEW: Assets can be imported directly by creating a data transfer that relates to Asset Framework.

The screenshot shows a 'Transfer' search interface. At the top, there is a 'Transfer' header with a help icon. Below it is a 'Search' box with a close button. The search is divided into two tabs: 'AF Element Search' (selected) and 'PI Point Search'. Under 'Search Criteria', there are several fields: 'AF Database' (Wind Power), 'Root Asset' (\Wind Power Generation Fleet), 'Element Name' (Enter Element Name), 'Attribute Name' (Enter Attribute Name), 'Attribute Value' (Enter Attribute Name, =, Enter Value), 'Template' (Wind Turbine), and 'Category' (<All>). A 'Search' button is located at the bottom right of the criteria section. Below the search criteria, there is a checked checkbox for 'Search Results (10)'. The results are displayed in a blue box with a list of 10 items, each starting with a checkmark and the path 'Wind Power Generation Fleet\Big Buffalo Wind Farm\GE01' through 'GE10'.

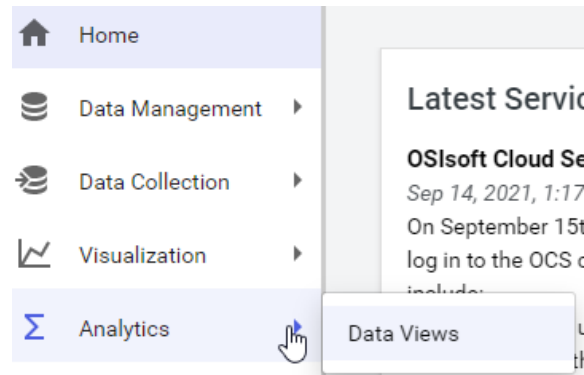
The assets of type (template) Wind Turbine are automatically selected including all PI Tags as streams and all Attributes as Metadata making the data import and association to assets much easier than using asset rules.

## Preparing Wind Farm Data for Machine Learning Using ADH Data Views

  
Video


Before reading this section, please refer to the following course YouTube video: <https://youtu.be/YRpU4xyZ69A>

Once the relevant stream metadata has been added, we now need to prepare our wind farm dataset to be used for machine learning, with the relevant associated metadata. For this purpose, we utilize the Data View feature in AVEVA™ Data Hub under the Analytics section of the Navigation menu:



A Data View is a user-selected subset of data from one or more streams, presented in a tabular format.

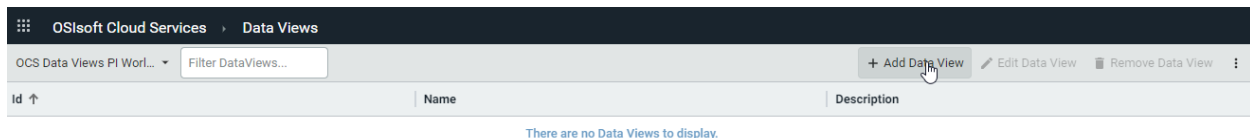
A data view is created by defining the target namespace, the source stream or streams, the desired data fields, time period, and interpolation interval. The ability to create data views in AVEVA™ Data Hub meshes directly with AVEVA's Data Science Enablement efforts, whereby users will be able to programmatically access Data View content for the purposes of advanced analytics. In this section, we will create a new Data View for the wind turbine data so that it is accessible externally in Microsoft Power BI and in a Jupyter Notebook.

  
Note

*Unlike Asset / Metadata Rules, each student can create their own Data View, which will not impact the shared SDS Streams. Thus, feel free to follow the instructions below to create your own Data View in ADH.*

To create a new data view, click Add Data View in the Data Views pane.

1)) Give the Data View a name as follows “Wind Turbine Data\_NNN\_XXXX-XX-XX”, where NNN are your initials and XXXX-XX-XX is today’s date in four digit year, two digit month and two digit day respectively.



2) The Query ID is “GE Query NNN”, where NNN are your initials

3) Select the Query Type to **assets**

Query Id  
GE Query TWS

Query Type  
 Streams  Assets

Query Value ?  
Search...

4) As Query Value enter "GE\*" press Search . This will return all the assets of the course.

Home | Data Management | Data Collection | Visualization | Analytics | Security | Developer Tools | Support

Name: Wind Turbine Data\_twe\_2021-09-16 | Description: Description... | Data View Shape: Standard | Narrow

Query Id: GE Query | Query Type:  Streams  Assets

Query Value ?  
GE\*

Asset Name	Asset Id
GE01	GE01
GE02	GE02
GE03	GE03
GE04	GE04
GE05	GE05
GE06	GE06
GE07	GE07
GE08	GE08
GE09	GE09
GE10	GE10

5) Press Save and Next

We then select the values. There should be nine.

**Add Data Fields** [X]

1 Select Data Fields ————— 2 Edit Data Fields  
Optional

Value [X] Filter by Summary Type... [X]

Filter by text...

Select All  Show Included Fields

**GE Query NNN (9)** ^

<input type="checkbox"/>	Active Power   IsQuestionable Property Id
<input type="checkbox"/>	Active Power   IsSubstituted Property Id
<input type="checkbox"/>	Active Power   SystemStateCode Property Id
<input checked="" type="checkbox"/>	Active Power   Value Property Id
<input type="checkbox"/>	Blade1 Actual Value   DigitalStateName Property Id
<input type="checkbox"/>	Blade1 Actual Value   IsAnnotated Property Id
<input type="checkbox"/>	Blade1 Actual Value   IsQuestionable Property Id
<input type="checkbox"/>	Blade1 Actual Value   IsSubstituted Property Id
<input type="checkbox"/>	Blade1 Actual Value   SystemStateCode Property Id
<input checked="" type="checkbox"/>	Blade1 Actual Value   Value Property Id
<input type="checkbox"/>	Blade2 Actual Value   DigitalStateName Property Id

Next Cancel Apply (9)

This is the result with the values in a tabular view:

On the bottom of the screen under by clicking on the time bar



A time selection window will appear

Index Configuration, keep the default settings for Start Index, End Index and Time Interval. These settings are not important now as they will be changed programmatically later. If the Save Defaults with Data View box is checked, the configuration set in these fields will be maintained the next time the Data View is accessed via the user interface.

The index configuration and retrieval method can be specified or overridden when making a data query. See [Get data view data](#).

Start Index (Local)

End Index (Local)


Time Interval (dd.hh:mm:ss)

Save Defaults with Data View ?

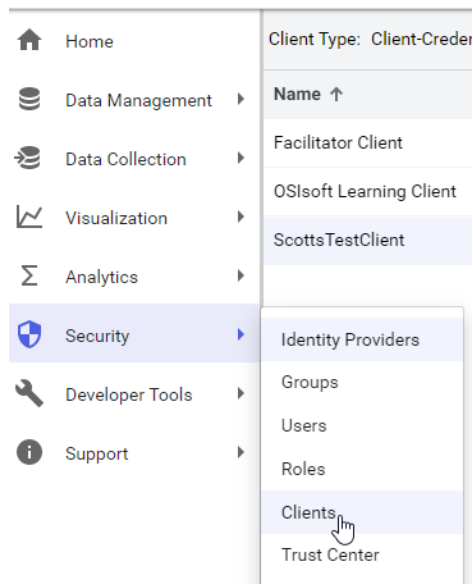
Click Save and Close

The Data View now appears to be in the correct format needed.

## Connecting to ADH Data View using an ADH Client

 Video	Before reading this section, please refer to the following course YouTube video: <a href="https://youtu.be/14Si8-cnHXU">https://youtu.be/14Si8-cnHXU</a>
--	--

For external programs to connect to ADH, an ADH Client needs to be set up first. This Client's details will then be used to connect to ADH and access this Data View. Navigate to the Clients page under Security in the Navigation menu.



There is already a Client defined for this course, called **PI World Client for Data View Lab**. Click on this client to view Details, Code Examples and more Information. The Client Type for this case is Client Credentials.

Client-credentials type clients are used for server to server communication where no user interaction is required. The client typically must authenticate with the token endpoint using its Client Id and Client Secret, provided they are valid. The default (and maximum) token lifetime is 3600 seconds, which means that authentication must be performed again if this time period elapses. Please make note of the Client Id and Client Secret.

The Client Id can be obtained at any time for an ADH Client in the Client Details section under the Details tab:

PI World Client for Data View Lab

Details Roles Code Examples Information

**Client Details**

**Client Id**  
41268d67-d633-4273-a771-8dccc281e3a5

**Tenant Id**  
c5da943f-2712-4dfa-a09a-791d949ee23b

**Name**  
PI World Client for Data View Lab

**Token Lifetime**  
3600 seconds

**Status**  
Enabled


However, the Client Secret is only available upon first creation of an ADH Client, thus needs to be copied immediately and stored in a safe place. However, if a user forgets or misplaces the Client Secret, another one can be added by clicking on +Add Secret in the Client Secrets section of the Clients page.

Client Secret Details

+ Add Secret Edit Secret Remove Secret

Status ↑	Description	Expiration Date
Active	Actual Secret	Never Expires

In this section, a user can also edit or remove an existing Client Secret. On the Edit Secret page, the Description or Expiration Date (including setting it to never expire) can be modified.



Note

*A Client Id and Client Secret are used to authenticate and retrieve a token that provides access to ADH. Any client that presents the Client Id and Client Secret will be able to authenticate and access data. A Client Id is not confidential, but a Client Secret is. Hence, it is imperative that the Client Secret be stored in a secure location that is accessible only to the client that needs to use it to authenticate and access ADH.*

For detailed information regarding adding a client-credentials type client, please refer to Appendix B.

Example code illustrating common connection syntax in different programming languages is provided on the Code Example tab on the right. These examples are important if a user does not want to use the ADH Python library but instead wants to connect to ADH via generic API calls instead.



## 3. Utilizing AVEVA™ Data Hub Data in Machine Learning Model

### a. Objective

The objective of this section is to connect to the wind farm ADH Data View created in the previous section, filter the data set to remove outliers and use this cleansed dataset to train, test and evaluate a machine learning model. This is all accomplished programmatically in a Jupyter Notebook via Python code.

### b. Tasks

All the tasks below are to be done in a Jupyter Notebook with Python code:

1. Connect to ADH and access the Wind Farm Data View
2. Use Power BI to get a first impression of the data
3. Determine variables that affect Wind Farm Active Power
4. Cleanse and prepare a Wind Farm dataset for machine learning
5. Train & evaluate a selected machine learning algorithm
6. Test trained machine learning model with sample input

### c. Approach & Details

We will connect the data using AVEVA Data Hub Power BI Connector and have a first look at it

We will connect to the created ADH Data View in a Jupyter Notebook locally on your Virtual Machine.

**This will be done on the instructors VM to avoid placing personal credentials into the training VMs**

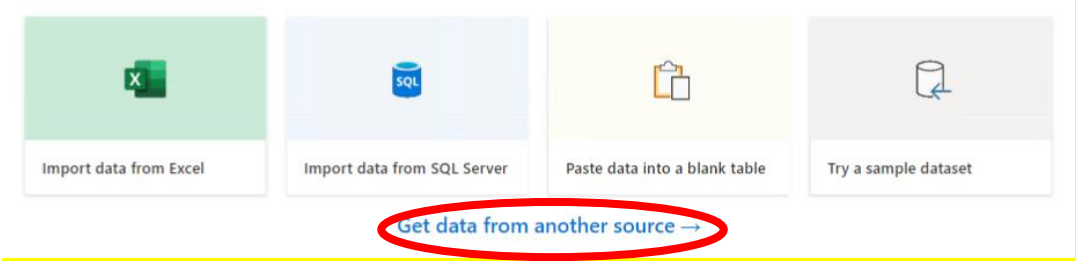
#### i. Connecting with AVEVA Data Hub Power BI Connector

The Connector has the following capabilities:

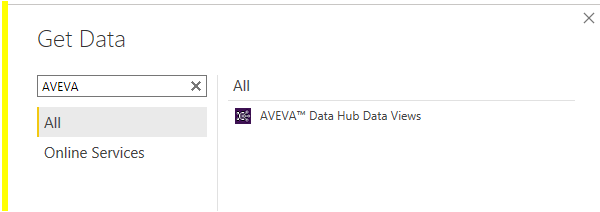
1. Import Data Views into Microsoft Power BI
2. Choose from Interpolated or Stored data retrieval modes
3. Configure dynamic time ranges
4. Supported in Power BI Desktop and the Power BI Service (through an On-Premises Data Gateway)

The connector takes care of all connection and data transfer issues, so analysis of the data can start immediately. The connector is preinstalled on your virtual machines.

Once Power BI launched, skip all the windows were an e-mail is requested. Click get data from another source

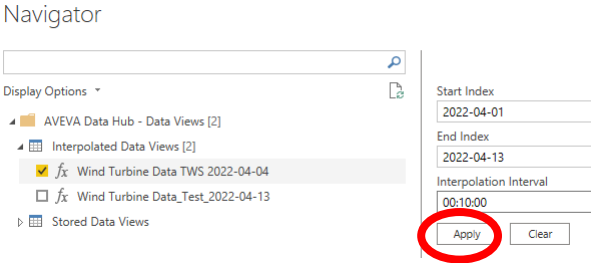


Create a new data connection. Search for AVEVA and pick the AVEVA™ Data Hub Data Views.



Press Connect In the following window the namespace “AVEVA Data Hub - Data Views”

Sign in with your credentials and chose the following dataset “Wind Turbine Dataview TWS 2022-04-04” Start Index 10 days ago, Format YYYY-MM-DD. End Index today. Interpolation Interval 00:10:00



press apply to get a preview

Then press load to get the data

Apply Clear

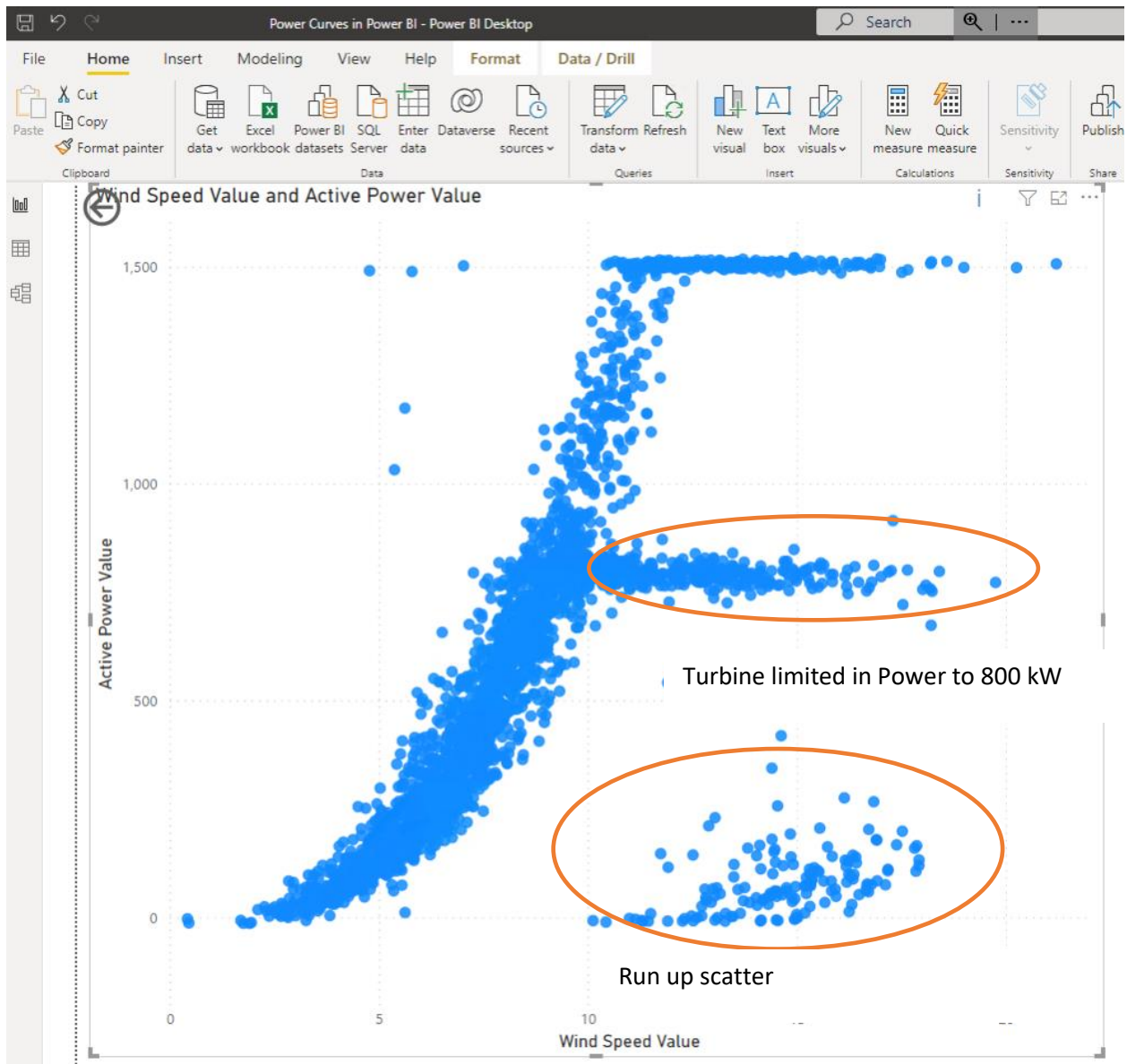
Wind Turbine Data TWS 2022-04-04

Timestamp	Name	Active Power Value	Blade1
04/01/2022 12:00:00 AM	GE01	-2.4512196	
04/01/2022 12:10:00 AM	GE01	90	
04/01/2022 12:20:00 AM	GE01	84.90909	
04/01/2022 12:30:00 AM	GE01	488	
04/01/2022 12:40:00 AM	GE01	23.866032	
04/01/2022 12:50:00 AM	GE01	55.339046	
04/01/2022 1:00:00 AM	GE01	63.63333	
04/01/2022 1:10:00 AM	GE01	-8.486033	
04/01/2022 1:20:00 AM	GE01	13.59931	
04/01/2022 1:30:00 AM	GE01	-2.8919492	
04/01/2022 1:40:00 AM	GE01	-2.1292372	
04/01/2022 1:50:00 AM	GE01	-1.3665254	
04/01/2022 2:00:00 AM	GE01	-0.6038136	

Load Transform Data Cancel

We will not go into the details of Power BI. There is a simple drill through Power BI Report built that takes a first look at the power curves by turbine.

You may open it, it is in the folder Lab Material “Power Curves in Power BI”




The turbine GE05 is limited in power, there is scattered data probably due to start ups the will need to be cleaned up before we apply machine learning.

**Close Power BI before continuing as it is consuming a lot of your VM resources!**

For more detail on the connector see:

<https://docs.osisoft.com/bundle/data-hub/page/visualize-data/power-bi-connector.html>

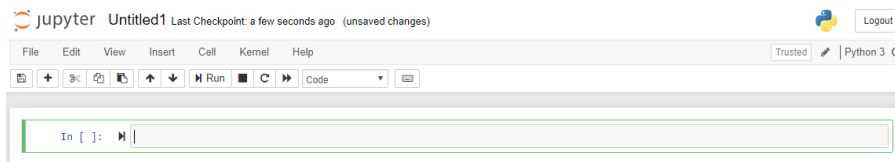
## ii.Using Jupyter Notebook Installed Locally on Virtual Machine

 Video	<p>Before reading this section, please refer to the following course YouTube video: <a href="https://youtu.be/npZYLFUvNGO">https://youtu.be/npZYLFUvNGO</a></p>
--	---

In order to connect to the Jupyter Notebook locally on your Virtual Machine, open the Lab Material folder on the Desktop, then double-click on the Start Jupyter Notebook.bat file. The screen below should appear after a few seconds, be patient:




Click on the New dropdown menu in the upper right corner, then select Notebook: Python 3 to create a new notebook with Python 3:



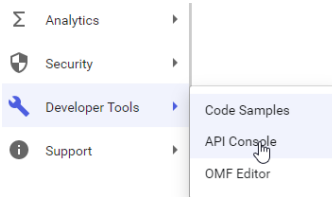
In the Course Material folder, the completed Jupyter Notebook is available for review titled **Wind Turbine ADH Data\_ADH Python Library**. It is suggested that you either follow along with this finished Notebook or copy and paste code sections into a new Notebook.

## iii.Connect to ADH and access the Wind Farm Data View

In this course, we are going to use the ADH Python library, which has built-in functions to perform the various ADH actions required. For documentation regarding the ADH Python library, as well as more help with Python, please refer to Appendix D.

 Video	<p>Before reading the next section regarding the API Console, please refer to the following course YouTube videos:</p> <ol style="list-style-type: none"><li>1. <a href="https://youtu.be/BCPZOOJc16I">https://youtu.be/BCPZOOJc16I</a></li><li>2. <a href="https://youtu.be/zBJ2Fz-EUsA">https://youtu.be/zBJ2Fz-EUsA</a></li><li>3. <a href="https://youtu.be/5QS_haLiu-0">https://youtu.be/5QS_haLiu-0</a></li></ol>
--	---


An alternative would be to access ADH via generic REST API calls. There is another Jupyter Notebook in the Course Materials folder titled **Wind Turbine ADH Data\_RESTAPI & OMF**, which shows you how to do the various ADH calls via REST API without the ADH Python library. Please note that the material refers to the Heritage OSisoft API! Adaptions need to be made. There is an API Console under the Developer Tools section of the Navigation menu which helps users verify their generic API calls:



The API Console provides a graphical interface to utilize the SDS REST API, which enables you to read and write SDS data.



For more information regarding API calls, please refer to Appendix D.

 Video	<p><i>Before reading the next section, please refer to the following course YouTube videos:</i></p> <ol style="list-style-type: none"> <li>1. <a href="https://youtu.be/uo7aCMtaaOc">https://youtu.be/uo7aCMtaaOc</a></li> <li>2. <a href="https://youtu.be/atwGVfEDzBE">https://youtu.be/atwGVfEDzBE</a></li> </ol>
--	--

If we utilize the Data Hub Python library connection methodology in the Jupyter Notebook, we need to import the necessary Python libraries, modify the config.ini file, connect to ADH, then authenticate and pass the identity token.

The **config.ini** file is available for review in the Course Material folder. The strings referenced in the ADH Python library functions below correspond to the sections and content of this file, which needs to be modified with the correct namespace, tenant ID, Client Id, Client Secret, etc. for this course:

```

IMPORTANT: replace these values with those provided by AVEVA
[Configurations]
Namespace = OCS Data Views PI World 2021
[Access]
Resource = https://datahub.connect.aveva.com
Tenant = c5da943f-2712-4dfa-a09a-791d949ee23b
ApiVersion = v1
[Credentials]
ClientId = 41268d67-d633-4273-a771-8dccc281e3a5
ClientSecret = ████████████████████


```

```

; IMPORTANT: replace these values with those provided by OSISOFT
[Configurations]
Namespace = OCS Data Views PI World 2021
[Access]
Resource = https://dat-b.osisoft.com
Tenant = 95e3e8c9-b327-4b36-92de-d61044f25f76
ApiVersion = v1
[Credentials]
ClientId = ████████████████████
ClientSecret = ████████████████████

```

Note the Client Id and Client Secret under the Credentials section of the config.ini file for connecting to the ADH Client we created, i.e. “OSISOFT Learning Client” Client.

 Video	<p>Before reading the next section, please refer to the following course          YouTube video: <a href="https://youtu.be/wUDY5wpjYwg">https://youtu.be/wUDY5wpjYwg</a></p>
--	--

We first import the necessary Python packages in our Jupyter Notebook and then connect to AVEVA™ Data Hub, where the code for this is as below:

```

In [1]: ! get_ipython().magic(u'config IPCompleter.greedy=True')

In [2]: ! import requests
import configparser
import json
import pandas as pd
from datetime import date, timedelta
from ocs_sample_library_preview import *

In [3]: ! config = configparser.ConfigParser()
config.read('config.ini')

ocsClient = OCSClient(config.get('Access', 'ApiVersion'), config.get('Access', 'Tenant'), config.get('Access', 'Resource'),
config.get('Credentials', 'ClientId'), config.get('Credentials', 'ClientSecret'))

namespaceId = config.get('Configurations', 'Namespace')

```

The next step is to get the Data View and pass it to a pandas Data Frame. The Start Index is set to 10 days prior to today’s date, the End Index is set to today’s date and the Time Interval is set to be 1 minute. The count for the maximum number of rows of the Data View passed to the Data Frame needs to be increased from the default of 1,000 to 150,000, so that all data contained is retrieved. Please ensure that the dataviewId corresponds to the Data View name entered previously i.e. **Wind Turbine Data\_NNN\_XXXX-**

**XX-XX**, where NNN are your initials and XXXX-XX-XX is today's date in four digit year, two digit month and two digit day respectively (e.g. Wind Turbine Data\_APN\_2020-08-28 in the example code below).


```
In [4]: #dateFrom = '2020-01-12'
#dateTo = '2020-01-23'
dateFrom = str(date.today()-timedelta(days=10))
dateTo = str(date.today()) #10 days of data
timeinterval = '00:01:00' #interpolate every minute

#dataviewId = Wind Turbine Data_NNN_XXXX-XX-XX, where NNN are your initials and XXXX-XX-XX is today's date in four
# digit year, two digit month and two digit day respectively (Wind Turbine Data_APN_2020-03-23 in this example)
dataviewId = "Wind Turbine Data_twe_2021-09-16"

data, nextPage, firstPage = ocsClient.DataViews.getDataInterpolated(namespaceId, dataviewId, startIndex=dateFrom,
                                                                    endIndex=dateTo, interval=timeinterval, count=150000)

df = pd.DataFrame(data)
df["Timestamp"] = pd.to_datetime(df["Timestamp"])
df
```

#### iv. Determine variables that affect Wind Farm Active Power

 Video	<p>Before reading this section &amp; Section 3.3.4.1, please refer to the following course YouTube video: <a href="https://youtu.be/e_5msGD5EME">https://youtu.be/e_5msGD5EME</a></p>
--	---

As a data scientist creating a machine learning model, one needs to be aware of the features that affect the label prediction. This would need to be discussed with the Subject Matter Expert (SME). Thus, in order to determine the features that affect the Active Power for a wind turbine, a correlation plot is added via the matplotlib Python library. However, first, the Data Frame column names must be abbreviated, in order to be displayed clearly in the plot. The code section for this, and the corresponding plot, are below:

```
In [5]: import matplotlib.pyplot as plt
import numpy as np

In [6]: #Renaming DataFrame column names to abbreviations, in order to display these column names clearly in a correlation plot

df.rename(columns = {'Blade1 Actual Value Value': 'BL1', 'Blade2 Actual Value Value': 'BL2',
                    'Blade3 Actual Value Value': 'BL3', 'Rotor Speed Value': 'RS', 'Turbine State Value': 'TS',
                    'Active Power Value': 'AP', 'Nacelle Position Value': 'NP', 'Generator Cooling Air Temperature Value': 'AT',
                    'Wind Speed Value': 'WS'}, inplace = True)

In [7]: #Check the correlation between Active Power and the rest of the variables

#retrieve the correlation table
df_corr = df.corr()

#increase the size of the figure
fig = plt.figure(figsize=(50,10))
ax = fig.add_subplot(111)

#set the color pallete (Red, yellow, green)
cax = ax.matshow(df_corr, cmap=plt.cm.RdYlGn)
fig.colorbar(cax)

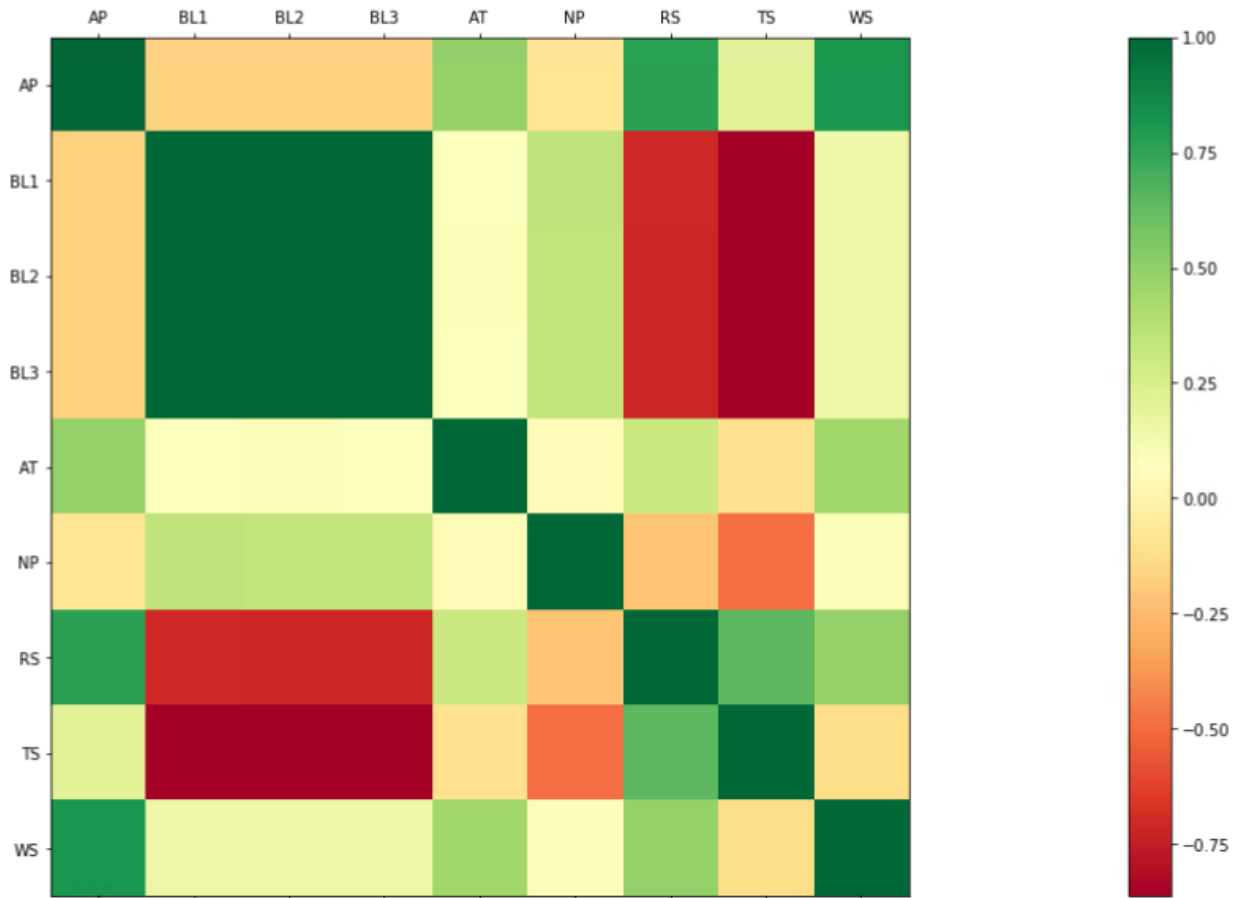
#configure the labels
labels = [c for c in df_corr.columns]

#make sure to show all the labels
ax.set_xticks(np.arange(len(labels)))
ax.set_yticks(np.arange(len(labels)))

#Setting labels for the x and y axes of the correlation plot
ax.set_xticklabels(labels)
ax.set_yticklabels(labels)

plt.show()
```





**Figure 4: Correlation plot for Wind Turbine Attributes**

As per the plot above, the Active Power is strongly correlated with Air Temperature, Rotor Speed, Turbine State and Wind Speed. Since Rotor Speed is directly affected by Wind Speed and the turbine only generates power when it is operating (the Turbine State), the two weather variables, Wind Speed and Air Temperature, are selected as the features of interest for our machine learning model, with Active Power as the desired label.

## v.Cleanse and prepare a Wind Farm dataset for machine learning

### Generating Wind Turbine Power Curve & Identifying Outlier Data Regions

In order to be consistent with the original Data View column names, we rename the Data Frame columns back to their original titles. Next, a scatter plot of Active Power vs. Wind Speed is added and visualized:

```
In [8]: #Renaming DataFrame column names from abbreviations back to their original full names
df.rename(columns = {'BL1':'Blade1, Actual Value Value', 'BL2':'Blade2, Actual Value Value',
                    'BL3':'Blade3, Actual Value Value', 'RS':'Rotor Speed Value', 'TS':'Turbine State Value',
                    'AP':'Active Power Value', 'NP':'Nacelle Position Value', 'AT':'Air Temperature Value',
                    'WS':'Wind Speed Value'}, inplace = True)

In [9]: #Plotting Active Power versus Wind Speed
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(111)
ax.scatter(df['Wind Speed Value'], df['Active Power Value'])
ax.set_xlabel('Wind Speed (m/s)')
ax.set_ylabel('Active Power (kW)')
ax.set_title('Active Power vs Wind Speed')
plt.show()
```

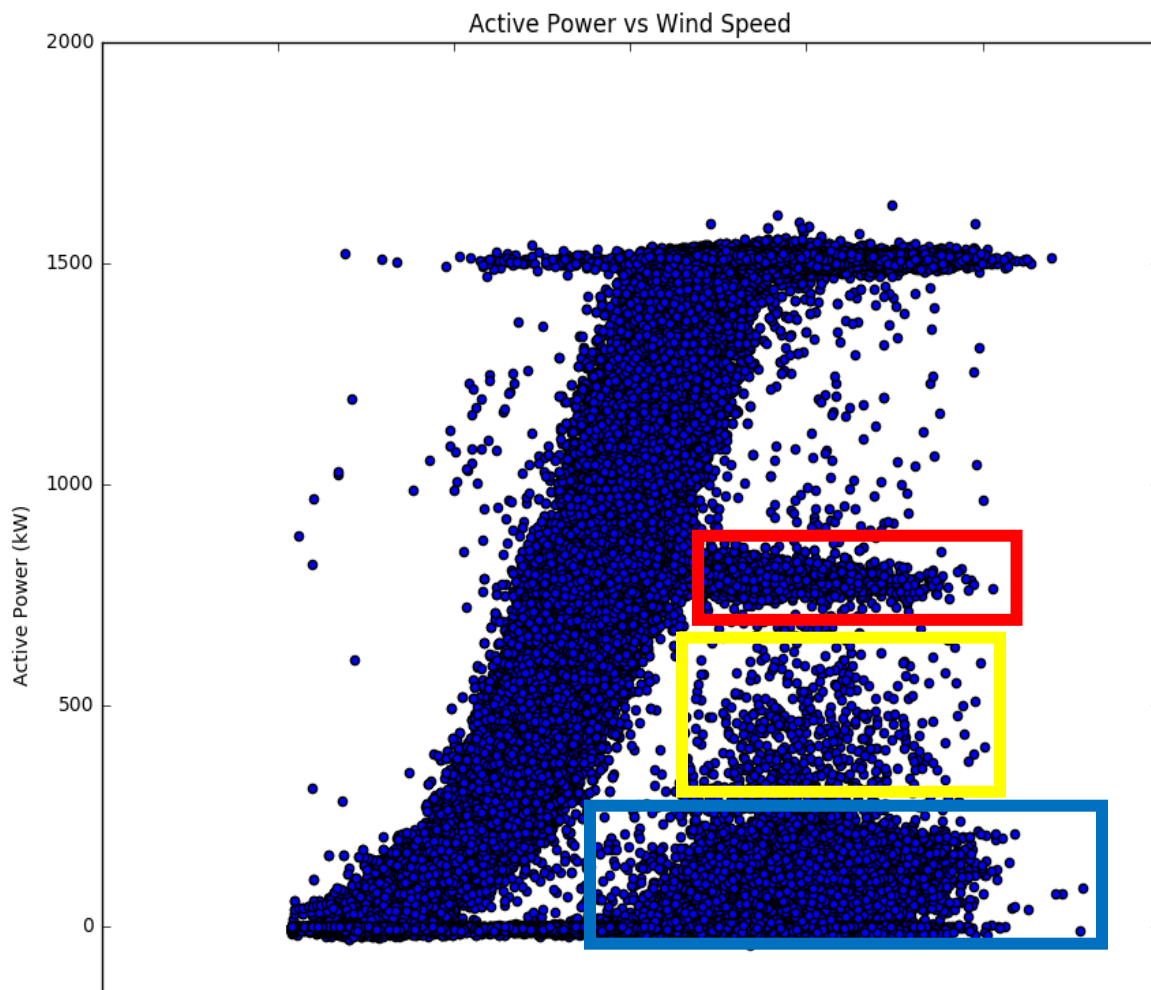



Figure 5: Plot of Wind Turbine Active Power vs. Wind Speed for Unfiltered Dataset

The data seems to describe something like the typical power curve for a turbine. This is good news for our modeling efforts. However, the data set needs to be cleaned up before using it to train a machine learning model. Upon consultation with the subject matter expert (SME) for wind turbines, 3 outlier regions are identified:

1. The red region is due to one of the turbines (GE05) having a different Rated Power than the others.
2. The blue region is due to the turbines not producing power even though the wind is blowing; this could be due to maintenance, shutdowns, or curtailments (e.g., the grid will not accept power).
3. The yellow region is due partially to some short periods of low production between stops and mostly to an unexplained period of low production during high winds.

These outliers need to be filtered out before we send the data to a machine learning model.

 Video	<i>Before reading this section, please refer to the following course YouTube video: <a href="https://youtu.be/QYNGtMKqhOc">https://youtu.be/QYNGtMKqhOc</a></i>
--	---

### Removing Turbine GE 05

First, for the red box in the scatter plot, GE05 is also excluded from the data set, because it has a much lower Rated Power than the other turbines. Also, high wind speeds (above 13 m/s) that do not alter the Active Power are also removed

```
In [11]: #Remove the GE05 turbine rows from the data frame because it has a Lower rating relative to all the other turbines
filterOutGE05 = df['Name'] != "GE05"
df_Filter = df[filterOutGE05]
```

### Removing negative Values

Then, any negative values of Active Power are removed (turbine is not turning and consuming some power for its internal control systems)

```
In [11]: #Filter out negative & excessive Active Power Values
filterNegativeActivePower = (df_Filter['Active Power Value'] >= 0)
df_Filter = df_Filter[filterNegativeActivePower]
```

### Exclude bad performing areas

. Next, in order to exclude operating periods corresponding to high wind with lower than expected power generation (yellow box in scatter plot), a filter criterion is used when 'Wind Speed' > 10

And 'Active Power' < 600. Typically, the power is expected to be above 1000 kW for the turbines of interest at these high wind speeds.

```
In [12]: #Remove the rows where we have a high wind speed and Low active power in order to keep only the normal operating conditions
filterOutLowPowerHighWindSpeedData = (~((df_Filter['Wind Speed Value'] > 10) & (df_Filter['Active Power Value'] < 600)))
df_Filter = df_Filter[filterOutLowPowerHighWindSpeedData]
```

### Use normal operating State only

For the blue box in the scatter plot (i.e. to eliminate times of little or no production due to curtailments, maintenance, etc.), we need to know the Turbine State of the wind turbines. We have a Turbine State Value column in our Data View, but these are integer values which correspond to specific operating states. In the df Data Frame, we need to use a filter criterion when the 'Turbine State Value' = 16.

```
In [14]: #Keep only the rows which correspond to the "Load Operation" state
filterLoadOperationState = df_Filter['Turbine State Value'] == 16
df_Filter = df_Filter[filterLoadOperationState]
```

### Filter out high wind speed

At high wind were the turbine dies no longer increase its power the blades pitch (turn away from the wind) offering less efficient power capture. The generator is at its limited power the turbine uses just the wind energy needed to keep up.

```
In [15]: #Filter out high Wind Speeds (> 13 m/s) that do not change the Active Power results
filterOutHighWind = df_Filter['Wind Speed Value'] < 13
df_Filter = df_Filter[filterOutHighWind]
```

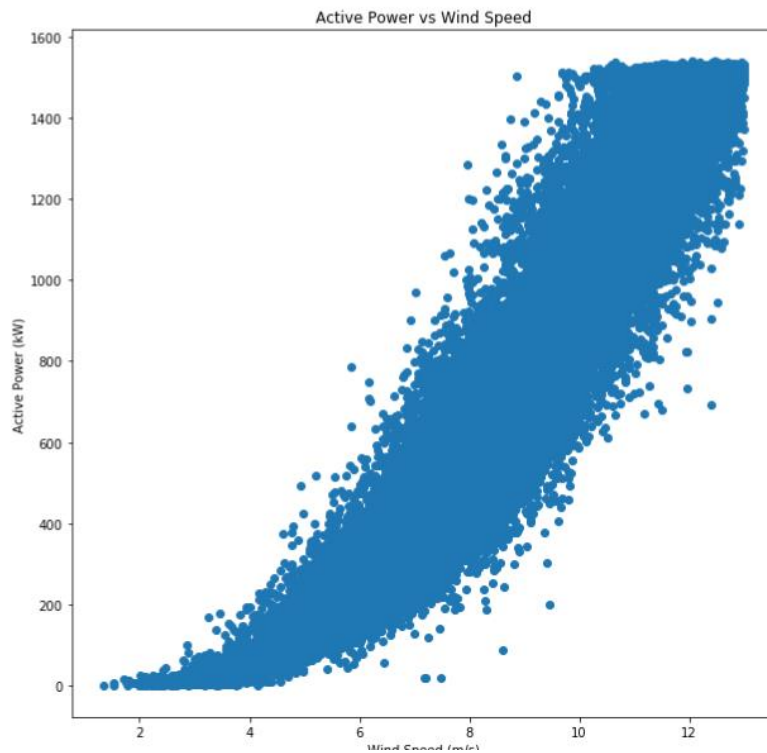
## Plot Results

The updated scatter plot now more closely resembles a typical Power vs. Wind Speed curve for a wind turbine (minus the section above the rated speed) as per below:

```
In [16]: #Plotting Active Power versus Wind Speed - filtered data frame representing Normal Operating Conditions
```


```
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(111)
ax.scatter(df_Filter['Wind Speed Value'], df_Filter['Active Power Value'])
ax.set_xlabel('Wind Speed (m/s)')
ax.set_ylabel('Active Power (kW)')
ax.set_title('Active Power vs Wind Speed')

plt.show()
```



**Figure 6: Plot of Wind Turbine Active Power vs. Wind Speed for Filtered Dataset**

## vi. Train & evaluate a selected machine learning algorithm

 Video	<p>Before reading this section, please refer to the following course YouTube videos:</p> <ol style="list-style-type: none"><li>1. <a href="https://youtu.be/-Xu099ubv9w">https://youtu.be/-Xu099ubv9w</a></li><li>2. <a href="https://youtu.be/jqQcfEnIzaA">https://youtu.be/jqQcfEnIzaA</a></li><li>3. <a href="https://youtu.be/9pwUHx60Tus">https://youtu.be/9pwUHx60Tus</a></li><li>4. <a href="https://youtu.be/WkneG5Hd6cl">https://youtu.be/WkneG5Hd6cl</a></li></ol>
--	--

The next step would be to randomly split the overall data into a training set and a testing set using a function from the scikit-learn Python package. 75% of the data set is used as training data, while 25% is used as scoring/testing data. The label data used are the Active Power values while the feature data used are the Air Temperature and Wind Speed values:

```
In [20]: #Prepare the training & testing/scoring data sets, and split them randomly
from sklearn.model_selection import train_test_split
#define the target variable to be predicted
y = np.nan_to_num(df_Filter['Active Power Value'].values)
#split the dataset randomly into test and train sets
X_train, X_test, y_train, y_test = train_test_split(np.nan_to_num(df_Filter[['Air Temperature Value', 'Wind Speed Value']]
y, test_size=0.25, random_state=42)
```

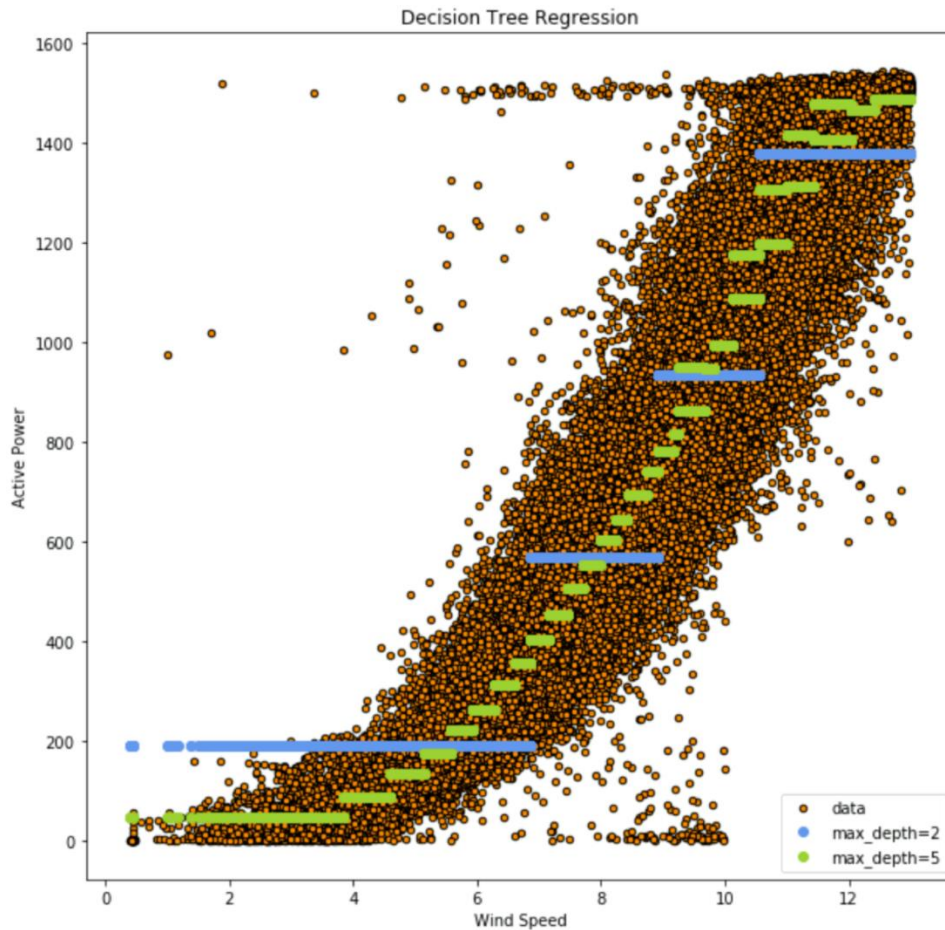
The machine learning model that we are selecting to use for this wind turbine case will be the Decision Tree Regressor (DTR) model, which is available in the scikit-learn package. For more information regarding the DTR algorithm, please refer to Appendix D.

```
In [23]: #Use the Decision Tree Regression Machine Learning model from scikit-Learn
from sklearn.tree import DecisionTreeRegressor
regr_1 = DecisionTreeRegressor(max_depth=2)
regr_2 = DecisionTreeRegressor(max_depth=5)
regr_1.fit(X_train, y_train)
regr_2.fit(X_train, y_train)

# Predict
y_1 = regr_1.predict(X_test)
y_2 = regr_2.predict(X_test)
```

The resulting fit to the data by the machine learning model can be plotted as per the code below:

```
In [87]: # Plot the results
plt.figure(figsize=(10,10))
plt.scatter(X_train[:,1], y_train, s=20, edgecolor="black", c="darkorange", label="data")
plt.scatter(X_test[:,1], y_1, color="cornflowerblue", label="max_depth=2")
plt.scatter(X_test[:,1], y_2, color="yellowgreen", label="max_depth=5")
plt.xlabel("Wind Speed")
plt.ylabel("Active Power")
plt.title("Decision Tree Regression")
plt.legend()
plt.show()
```



**Figure 7.A: Wind Turbine Power Curve with DTR Algorithm Fit to Filtered Dataset**

As can be seen from the plot above, the Decision Tree Regressor model with a maximum depth of 5 decision trees fits the training dataset well.

We can visualize the binary nodes of the max\_depth=2 decision tree to better understand how it works:

```
In [19]: # Visualize Decision Tree (Depth 2)
from sklearn import tree
plt.figure(figsize=(12,12)) # set plot size (denoted in inches)
tree.plot_tree(regr_1, feature_names = FEATURE_NAMES, filled = True, fontsize=10);
plt.show()
```

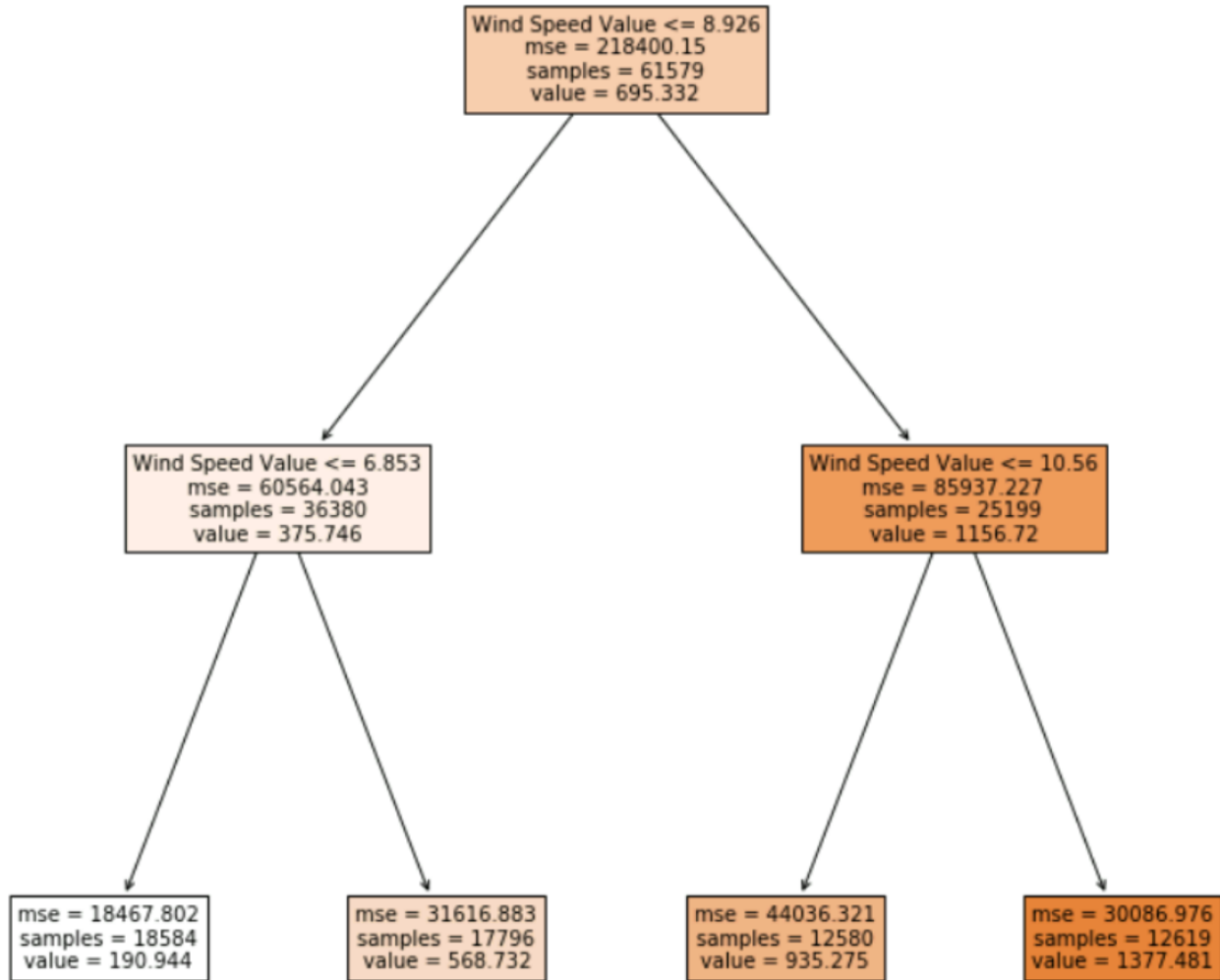



Figure 7.B: Wind Turbine Power Curve with DTR Algorithm Fit to Filtered Dataset

vii. Test trained machine learning model with sample input

 Video	<p>Before reading this section, please refer to the following course YouTube video: <a href="https://youtu.be/F3JhYkHKvZE">https://youtu.be/F3JhYkHKvZE</a></p>
--	---

Next, this machine learning model is saved to a file via the pickle module, so that we can access it later and feed live weather data to it. This model is then opened from the file, tested/scored and a sample prediction generated from a given set of air temperature and wind speed values, as per below:



```

In [28]: M #save the machine Learning model to disk
import pickle
filename = 'WT_ActivePower_model.sav'
pickle.dump(regr_2, open(filename, 'wb'))

In [29]: M #Test the model with the scoring/testing data set
loaded_model = pickle.load(open(filename, 'rb'))
result = loaded_model.score(X_test, y_test)
#print the model score
print(result)

0.9489498011964014

In [30]: M #Sample prediction
# define input
new_input = [[45, 6.6]] #Temp=45 F, Wind Speed = 6.6 m/s
# get prediction for new input
new_output = regr_2.predict(new_input)
print(new_output)

[279.67192816]

```

The coefficient of determination ( $R^2$ ) of the testing data set is approximately 0.9, which indicates that the algorithm picked can generate sufficiently accurate active power predictions from a given set of wind speed and air temperature data. The sample input of 45 F and 6.6 m/s generated an active power result of 302 kW.

Congratulations! You have successfully created a predictive machine learning model using data from ADH in a Jupyter Notebook!

## 4. Getting Forecasted Weather Data into ML Model

### a. Objective


The objective of this section is to obtain live weather forecasts for Amarillo, TX (the location where the wind farm operates) and predict Active Power using the previously created Machine Learning model, given Wind Speed and Air Temperature forecast data.

### b. Tasks

1. Connect and retrieve forecasted weather data via the Open Weather API
2. Utilize retrieved weather data to predict Active Power from machine learning model

### c. Approach & Details

#### i. Connect and retrieve forecasted weather data via the Open Weather API

 Video	<p>Before reading this section, please refer to the following course YouTube videos:</p> <ol style="list-style-type: none"><li>1. <a href="https://youtu.be/X50WOWhIGGw">https://youtu.be/X50WOWhIGGw</a></li><li>2. <a href="https://youtu.be/M-rLSyngYHc">https://youtu.be/M-rLSyngYHc</a></li></ol>
---	--

Live weather forecast data is available via the Open Weather API: <https://openweathermap.org/api>. A free subscription can be obtained, which allows programmatic access via an API key to obtain weather data for many cities for 5 days into the future, updated every 3 hours, including for Amarillo, TX. This weather data can be brought into the Jupyter Notebook via the code section below, which returns the result in a JavaScript Object Notation (JSON) format.

```
In [117]: #Call the OpenWeather API to retrieve the forecasted air temperature and wind speed for Amarillo, TX for the next 5 days
import requests
url="https://api.openweathermap.org/data/2.5/forecast?q=Amarillo,US&APPID=5dac981ce33f41f61d8d1ea06ee89798"
responseWeatherForecast=requests.get(url)

In [118]: responseWeatherForecast.json()

Out[118]: {'city': {'coord': {'lat': 35.222, 'lon': -101.8313},
  'country': 'US',
  'id': 5516233,
  'name': 'Amarillo',
  'population': 190695,
  'sunrise': 1581687314,
  'sunset': 1581726481,
  'timezone': -21600},
  'cnt': 40,
  'cod': '200',
  'list': [{'clouds': {'all': 99},
  'dt': 1581714000,
  'dt_txt': '2020-02-14 21:00:00',
  'main': {'feels_like': 270.84,
  'grnd_level': 890,
  'humidity': 67,
  'pressure': 1020,
  'sea_level': 1020,
  'temp': 279.54,
```



Note

If an error is received while accessing the Open Weather API, please try again later, as this is most probably due to the limit of 60 calls per minute offered by the free subscription being exceeded.



Video

Before reading the next section, please refer to the following course YouTube video: <https://youtu.be/sy3ofJ7sVlw>

This JSON response is stored in a pandas DataFrame, with the necessary unit and format conversions needed. For instance, the air temperature from Open Weather is sampled in degrees Kelvin, thus needs to be converted to degrees Fahrenheit. The wind speed is sampled in meters/sec, so no conversion is needed. In addition, the timestamp in the JSON response is a string, thus needs to be converted to a Date/Time format. The Timestamp, Air Temperature and Wind Speed values are all stored as NumPy arrays. The code for this section is as below:

```
In [26]: #Use the machine Learning model developed previously to predict the Active Power and add the values to the existing Data

import pickle
filename = 'WT_ActivePower_model.sav'
loaded_model = pickle.load(open(filename, 'rb'))

from decimal import Decimal
PredictedPowerArray=[]

for index, row in dfWeatherForecast.iterrows():
    new_input = [[row['Temp (F)'], row['Wind Speed (m/s)']]
    result = loaded_model.predict(new_input)
    np.array(PredictedPowerArray.append(result))

dfWeatherForecast['Predicted Active Power (kW)']=pd.DataFrame(PredictedPowerArray)

dfWeatherForecast.round(2)
```

Out[26]:

	Timestamp	Temp (F)	Wind Speed (m/s)	Predicted Active Power (kW)
0	2021-09-17 15:00:00	66.45	4.64	116.67
1	2021-09-17 18:00:00	76.23	6.87	401.34
2	2021-09-17 21:00:00	88.99	2.95	41.77
3	2021-09-18 00:00:00	85.82	2.48	41.77
4	2021-09-18 03:00:00	72.81	4.08	41.77
5	2021-09-18 06:00:00	69.37	2.03	41.77
6	2021-09-18 09:00:00	66.16	1.90	41.77
7	2021-09-18 12:00:00	65.26	3.06	41.77
8	2021-09-18 15:00:00	74.57	4.40	116.67
9	2021-09-18 18:00:00	85.86	5.15	175.51
10	2021-09-18 21:00:00	90.37	5.59	213.23
11	2021-09-19 00:00:00	86.76	7.10	401.34

## ii.Utilize retrieved weather data to predict Active Power from ML model



Video

Before reading this section, please refer to the following course YouTube video: <https://youtu.be/uexAC58YiwY>

This Air Temperature and Wind Speed data from Open Weather are then passed to the machine learning model saved in a file previously, in order to predict the Active Power values.

These predicted Active Power values (in units of kW) are then stored in the same Data Frame as a new column, as per the code section below:

```
In [34]: #Use the machine learning model developed previously to predict the Active Power and add the values to the existing Data
import pickle
filename = 'WT_ActivePower_model.sav'
loaded_model = pickle.load(open(filename, 'rb'))

from decimal import Decimal
PredictedPowerArray=[]

for index, row in dfWeatherForecast.iterrows():
    new_input = [[row['Temp (F)'], row['Wind Speed (m/s)']]
    result = loaded_model.predict(new_input)
    np.array(PredictedPowerArray.append(result))

dfWeatherForecast['Predicted Active Power (kW)']=pd.DataFrame(PredictedPowerArray)

dfWeatherForecast.round(2)
```

Out[34]:

	Timestamp	Temp (F)	Wind Speed (m/s)	Predicted Active Power (kW)
0	2021-09-17 15:00:00	66.29	4.64	116.67
1	2021-09-17 18:00:00	71.87	6.87	401.34
2	2021-09-17 21:00:00	81.43	2.95	41.77
3	2021-09-18 00:00:00	85.82	2.48	41.77
4	2021-09-18 03:00:00	72.81	4.08	41.77
5	2021-09-18 06:00:00	69.37	2.03	41.77
6	2021-09-18 09:00:00	66.16	1.90	41.77
7	2021-09-18 12:00:00	65.26	3.06	41.77
8	2021-09-18 15:00:00	74.57	4.40	116.67
9	2021-09-18 18:00:00	85.86	5.15	175.51
10	2021-09-18 21:00:00	90.37	5.59	213.23
11	2021-09-19 00:00:00	86.76	7.10	401.34
12	2021-09-19 03:00:00	77.43	7.05	401.34

## 5. Sending External Data Back to ADH


### a. Objective

The objective of this section is to send the forecasted data back to ADH as a new stream in the Sequential Data Store (SDS) of the Namespace we are using for this course, then graph this data in the Visualization section.

### b. Tasks


1. Create a new complex SDS Type for the forecasted data
2. Create a new SDS Stream
3. Send forecasted data back to ADH via this new SDS Stream
4. Analyze & visualize forecasted data in ADH

### c. Approach & Details

 Video	<p><i>Before starting this section, please refer to the following course YouTube video: <a href="https://youtu.be/4pRcV4jS668">https://youtu.be/4pRcV4jS668</a></i></p>
--	---

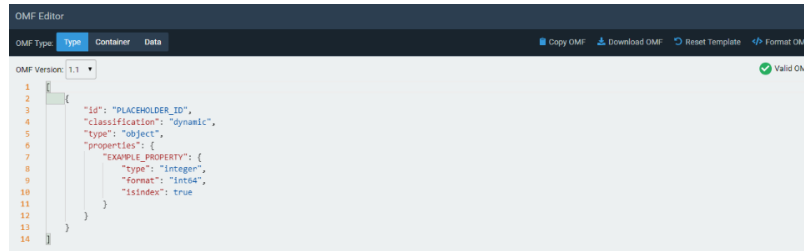
In order to send data back to ADH in this course, we utilize the ADH Python Library once again.

However, we can also do this via the OSisoft Message Format or OMF protocol. This method is shown in the other Jupyter Notebook in the Course Materials folder titled ***Wind Turbine ADH Data\_RESTAPI & OMF.***

 Video	<p><i>Before reading the next section regarding OMF, please refer to the following course YouTube videos:</i></p> <ol style="list-style-type: none"><li>1. <a href="https://youtu.be/5bwtHI9NS7U">https://youtu.be/5bwtHI9NS7U</a></li><li>2. <a href="https://youtu.be/L43UB8lae7U">https://youtu.be/L43UB8lae7U</a></li></ol>
--	---

OMF defines a set of message headers and message bodies, which are written in JSON and are used to generate compliant messages for data ingress. For ADH, OMF can be used to create types, create streams, and populate streams with data. There are three message formats that can be used to accomplish these tasks: type messages, container messages, and data messages, respectively. The OMF Editor in the

Developer Tools section of the Navigation menu helps build and validate OMF messages to be ingested by the ADH Data Store.




```
OMF Editor
OMF Type: Type Container Data
Copy OMF Download OMF Reset Template Format OMF
OMF Version: 1.1 Valid OMF
1 {
2   "id": "PLACEHOLDER_ID",
3   "classification": "dynamic",
4   "type": "object",
5   "properties": {
6     "EXAMPLE_PROPERTY": {
7       "type": "integer",
8       "format": "int64",
9       "indexed": true
10    }
11  }
12 }
13
14 }
```


For more information regarding OMF, please refer to Appendix D.

But in order to use the OMF methodology, a new OMF Connection needs to be created in ADH first, so that we can use this channel to send data streams to the SDS. For more information on configuring a new OMF Connection, please refer to Appendix C.

### i. Creating a new SDS Type


 Video	<p>Before reading this section, please refer to the following course YouTube video: <a href="https://youtu.be/8w748p0sl54">https://youtu.be/8w748p0sl54</a></p>
--	---

In order to send our forecasted data (which includes Air Temperature, Wind Speed and Active Power values) back to ADH, a new SDS Type needs to be created first. This new type will be complex, i.e. a collection of simple atomic types. This time-series indexed complex type with 3 values will be utilizing two simple types, DateTime and Double, in order to define the data shape/structure.

 Note	<p>The alternative <b>Wind Turbine ADH Data_RESTAPI &amp; OMF</b> Notebook in the Course Materials folder does not create a new SDS Type for the new data stream but instead uses the pre-existing simple PI-Float32 type.</p>
---	--

In order to create a new SDS Type, there are 2 methods:

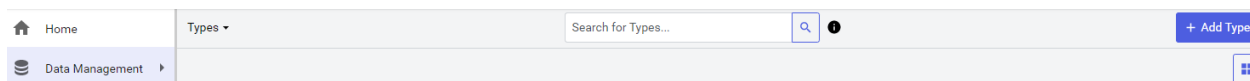
1. Create the new type via the user interface
2. Create the new type programmatically

 Note	<p>Unlike Metadata Rules, each student can create their own SDS Types. Thus, feel free to follow the instructions below to create your own SDS Types in ADH.</p>
---	--

## Creating New SDS Type Using the User Interface

### **Example only do not execute it will be done programmatically!!!**

The first step would be to navigate to the Sequential Data Store under Data Management in the Navigation menu.



The next step would be to choose on Types, then Add Type. Enter **Wind Turbine Predictions Time\_NNN\_XXXX-XX-XX** for the Id and Name, where NNN are your initials and XXXX-XX-XX is today's date in four digit year, two digit month and two digit day respectively (i.e. Wind Turbine Predictions Time\_APN\_2020-08-28 in the example below). Under the Description field, enter "A Time-Series indexed type with 3 values".

Click the + button to the right of the Properties field, so that there are 4 total Properties. These are for Air Temperature, Wind Speed, Predicted Active Power and Timestamp. Enter the following for Id, Name and Type, then check the Key box for Timestamp:

Wind Turbine Predictions Time\_twe\_2021-09-15 ⋮

Types are read only and cannot be edited.

<b>Id</b>	<b>Name</b>
Wind Turbine Predictions Time_twe_2021-09-15	Wind Turbine Predictions Time_twe_2021-09-15
<b>Description</b>	<b>Base Type</b>
A Time-Series indexed type with 3 values	

**Properties (4)**

Key Id	Name	Type	UOM
<input checked="" type="checkbox"/> Timestamp	Timestamp	DateTime	
<input type="checkbox"/> Air_Temperature	Air Temperature	Double	
<input type="checkbox"/> Wind_Speed	Wind Speed	Double	
<input type="checkbox"/> Predicted_Active_Power	Predicted Active Power	Double	

Click Save to store this new complex SDS Type.

## Creating New SDS Type Programmatically using ADH Python Library

In order to create the new SDS Type programmatically, we first need to reconnect to ADH and authenticate our token using the config.ini file and the ADH Python library functions, then create this new SDS type titled **Wind Turbine Predictions Time\_NNN\_XXXX-XX-XX**, where NNN are your initials and XXXX-XX-XX is today's date in four digit year, two digit month and two digit day respectively ((i.e. Wind Turbine Predictions Time\_REB\_2020-08-28 in the example below).

```
In [26]: > config = configparser.ConfigParser()
> config.read('config.ini')

> ocsClient = OCSClient(config.get('Access', 'ApiVersion2'), config.get('Access', 'Tenant'), config.get('Access', 'Resource'),
> config.get('Credentials', 'ClientId'), config.get('Credentials', 'ClientSecret'))

> namespaceId = config.get('Configurations', 'Namespace')
```

```
In [27]: > #SDS Type Name = Wind Turbine Predictions Time_NNN_XXXX-XX-XX, where NNN are your initials and XXXX-XX-XX is today's date in
> #four digit year, two digit month and two digit day respectively (Wind Turbine Predictions Time_REB_2020-08-28 below)

> typeWindTurbinePredTimeName = "Wind Turbine Predictions Time_REB_2020-08-28"

> typeWindTurbinePredTime = SdsType(id=typeWindTurbinePredTimeName, description="A Time-Series indexed type with 3 values",
> sdsTypeCode=SdsTypeCode.Object)

> doubleType = SdsType()
> doubleType.Id = "doubleType"
> doubleType.SdsTypeCode = SdsTypeCode.Double

> timeType = SdsType()
> timeType.Id = "DateTimeType"
> timeType.SdsTypeCode = SdsTypeCode.DateTime

> airtemperature = SdsTypeProperty()
> airtemperature.Id = "Air_Temperature"
> airtemperature.Name = "Air Temperature"
> airtemperature.SdsType = doubleType

> windspeed = SdsTypeProperty()
> windspeed.Id = "Wind_Speed"
> windspeed.Name = "Wind Speed"
> windspeed.SdsType = doubleType

> activepower = SdsTypeProperty()
> activepower.Id = "Predicted_Active_Power"
> activepower.Name = "Predicted Active Power"
> activepower.SdsType = doubleType

> time = SdsTypeProperty()
> time.Id = "Timestamp"
> time.Name = "Timestamp"
> time.SdsType = timeType
> time.IsKey = True

> typeWindTurbinePredTime.Properties = [airtemperature, windspeed, activepower, time]

> ocsClient.Types.getOrCreateType(namespaceId, typeWindTurbinePredTime)
```


If we go to the ADH portal, we should see this new SDS Type under Data Management> Sequential Data Store> Types in the Navigation menu:

OSisoft Cloud Services > Sequential Data Store

Id	Name
PI-Digital	PI Digital
PI-Float32	PI Float32
PI-Float64	PI Float64
PI-Int16	PI Int16
PI-Int32	PI Int32
PI-String	PI String
PI-Timestamp	PI Timestamp
Wind Turbine Predictions Time_CBD_2021-09-10	Wind Turbine Predictions Time_CBD_2021-09-10
Wind Turbine Predictions Time_twa_2021-09-10	Wind Turbine Predictions Time_twa_2021-09-10




## ii. Creating a New SDS Stream



Video

Before reading this section, please refer to the following course YouTube video: <https://youtu.be/h3lq1XL36pk>

After the creation of a new SDS Type, the forecasted air temperature, wind speed and active power values will be stored in a new SDS Stream based on this new type.




Note

The alternative **Wind Turbine ADH Data\_RESTAPI & OMF** Notebook in the Course Materials folder creates a new SDS Stream via OMF to only contain the active power values with a PI-Float32 type.

There are also 2 methods to create a new SDS Stream:

1. Create the new stream via the user interface
2. Create the new stream programmatically



Note

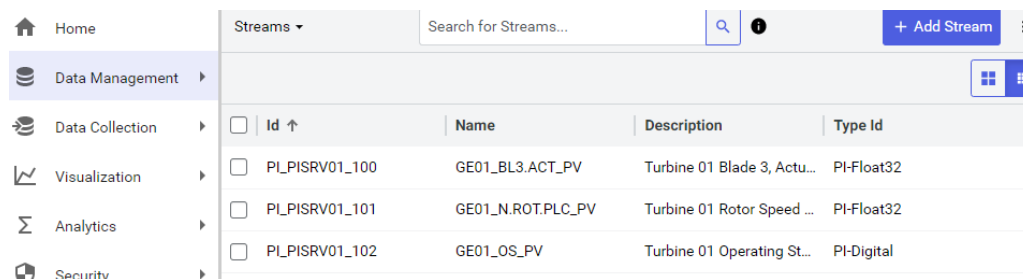
Unlike Metadata Rules, each student can create their own SDS Streams. Thus, feel free to follow the instructions below to create your own SDS Streams in ADH.

### Creating the New Stream Using the User Interface

The first step would be to navigate to the Sequential Data Store under Data Management in the Navigation menu.

The next step would be to click on Streams, then Add Stream. In the Stream Id and Name fields, enter **WT\_5dForecast\_NNN\_XXXX-XX-XX** and **Wind Turbine 5 Day Forecast Data\_NNN\_XXXX-XX-XX** respectively, where NNN are your initials and XXXX-XX-XX is today's date in four digit year, two digit month and two digit day respectively.

In the Description field, enter, and in the Type Id field, select the previously created type **Wind Turbine Predictions Time\_NNN\_XXXX-XX-XX** from the dropdown menu:



<input type="checkbox"/>	Id ↑	Name	Description	Type Id
<input type="checkbox"/>	PL_PISR01_100	GE01_BL3.ACT_PV	Turbine 01 Blade 3, Actu...	PI-Float32
<input type="checkbox"/>	PL_PISR01_101	GE01_N.ROT.PLC_PV	Turbine 01 Rotor Speed ...	PI-Float32
<input type="checkbox"/>	PL_PISR01_102	GE01_OS_PV	Turbine 01 Operating St...	PI-Digital

## Creating the New Stream Programmatically using ADH Python Library

We create the new SDS Stream programmatically via the ADH Python library functions as below, assuming we already connected to ADH and authenticated our token as per the previous section. This new stream Id is **WT\_5dForecast\_NNN\_XXXX-XX-XX** and the Stream Name is **Wind Turbine 5 Day Forecast Data\_NNN\_XXXX-XX-XX**, where NNN are your initials and XXXX-XX-XX is today's date in four digit year, two digit month and two digit day respectively.

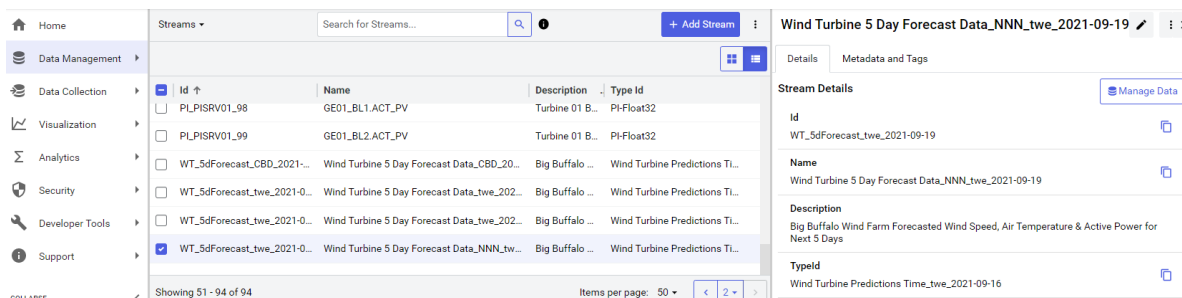
```
In [28]: #SDS Stream Id = WT_5dForecast_NNN_XXXX-XX-XX, SDS Stream Name = Wind Turbine 5 Day Forecast Data_NNN_XXXX-XX-XX,
#where NNN are your initials and XXXX-XX-XX is today's date in four digit year, two digit month and two digit day
#respectively

StreamId = "WT_5dForecast_twe_2021-xx-xx"
StreamName = "Wind Turbine 5 Day Forecast Data_twe_2021-xx-xx"
StreamDescr = "Big Buffalo Wind Farm Forecasted Wind Speed, Air Temperature & Active Power for Next 5 Days"

In [29]: stream = SdsStream()
stream.Id = StreamId
stream.Name = StreamName
stream.Description = StreamDescr
stream.TypeId = typeWindTurbinePredTime.Id

ocsClient.Streams.createOrUpdateStream(namespaceId, stream)
```

If we go to the ADH portal, we should see this new SDS Stream under Data Management> Sequential Data Store> Streams in the Navigation menu:



The screenshot shows the ADH portal interface. On the left is a navigation menu with 'Data Management' selected. The main area displays a table of streams. The table has columns for 'Id', 'Name', 'Description', and 'Type Id'. The stream 'WT\_5dForecast\_twe\_2021-09-19' is selected. On the right, the 'Stream Details' panel shows the following information:

Field	Value
Id	WT_5dForecast_twe_2021-09-19
Name	Wind Turbine 5 Day Forecast Data_NNN_twe_2021-09-19
Description	Big Buffalo Wind Farm Forecasted Wind Speed, Air Temperature & Active Power for Next 5 Days
TypeId	Wind Turbine Predictions Time_2021-09-16

### iii. Sending Data to New SDS Stream



Before reading this section & Section 5.3.4, please refer to the following course YouTube video: <https://youtu.be/Wf8z4fNBPb8>


Once this new data stream is created, either using the ADH User Interface or programmatically, the data values can then be sent to this new stream. This is done by looping through the dfWeatherForecast Data Frame's rows and using an ADH Python library function as per below:

```
In [30]: #Prepare the newly created OCS data stream values by looping through the dfWeatherForecast Data Frame's rows

values = []

for index, row in dfWeatherForecast.iterrows():
    values.append({"Air_Temperature": row["Temp (F)"], "Wind_Speed": row["Wind Speed (m/s)"],
                  "Predicted_Active_Power": row["Predicted Active Power (kW)"], "Timestamp":
                  row["Timestamp"].strftime('%Y-%m-%d %H:%M:%S') })

ocsClient.Streams.updateValues(namespaceId, stream.Id, json.dumps(values))
```



Note

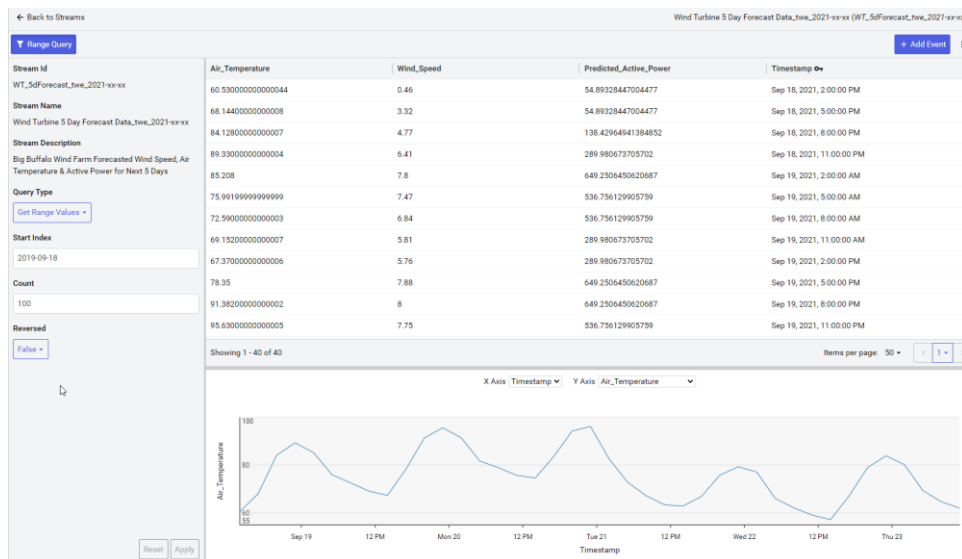
The alternative **Wind Turbine ADH Data\_RESTAPI & OMF** Notebook in the Course Materials folder sends data via OMF. Since OMF needs all the fields and values as strings, the data contained in the Data Frame as DateTime and Float types need to be converted to strings and must be comma-separated.

Then, a new values message would need to be created and Predicted Active Power data containing timestamps and values posted using the requests.post method

#### iv. Analyzing the New SDS Stream Data in ADH

The newly created SDS stream and data is now available to be viewed in ADH. Navigate to the Sequential Data Store under Data Management>Streams and select the newly created SDS Stream:

On the Data page under Streams, the values can be viewed by selecting Get Range Values under Query Type.




From the data stream tables and graphs, we can confirm that the forecasted data from our machine learning model has been successfully sent and stored in ADH.

We can also plot the historical Active Power values for one of the turbines, i.e. GE02 in this case, with the forecasted Active Power values:



As the real-time Active Power data is streamed to ADH, we can later compare how the actual values match up with the forecasted data.

**Congratulations! With this, you have successfully completed this course!**

 Video	<p>To conclude this course, please refer to the following course YouTube video: <a href="https://youtu.be/WBnygON2IBA">https://youtu.be/WBnygON2IBA</a></p>
--	---

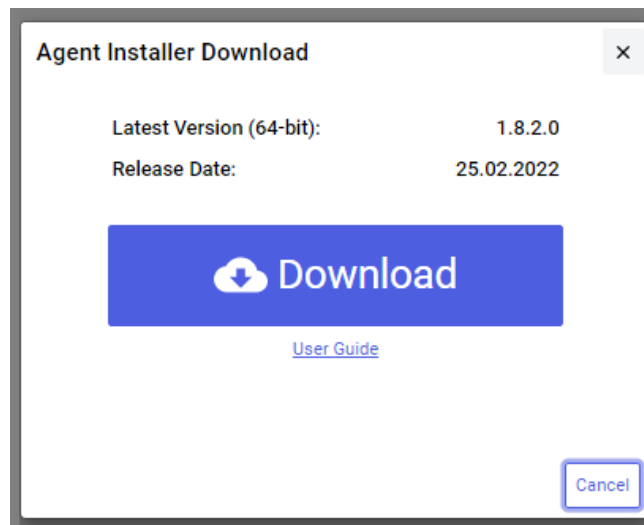
## 6. Appendix A: Streaming Data to ADH from PI Server

### a. AVEVA™ PI System Connections

You can establish high-throughput data connections from a AVEVA™ PI System into ADH with the Data Collection Feature

AVEVA™ Data Hub > PI to Data Hub Agents

Filter Agents...	Description ↑	Status	Version	Data Archive	Region	Namespace
	Data from feed environment 434	Good	1.8.2.0	PISRV01	westus	AVEVA Data Hub - D...



1. Download the PI to AVEVA™ Data Hub Agent Installation Kit displays.
2. Click the Download Install Kit link. Refer to the Getting Started Guide link in the Details tab for installation instructions.

## b. Installing the PI to AVEVA™ Data Hub Agent

PI to AVEVA™ Data Hub enables data transfer from your on-premises PI Server(s) to AVEVA™ Data Hub (ADH). The actual release of PI to AVEVA Data Hub supports the following features:

1. Transfer of a selection of AVEVA™ PI Data Archive PI Points from PI Servers to ADH Sequential Data Store (SDS) streams. Some of the PI Point attributes information is transferred as SDS stream metadata and properties.
2. Simultaneous transfer of both historical and streaming data from AVEVA™ PI Data Archive(s) to SDS for a selection of PI Points
3. Configuration of a PI Points selection and data transfer management via the ADH Customer Portal.

AVEVA™ PI to ADH Agent has two major components:

1. An on-premises component called the “PI to AVEVA Data Hub Agent”, which is installed on a computer used as a bridge between the source PI Server and the ADH destination. It runs as a service and performs fast, secure data transfer.
2. A cloud component called a “PI System Connection” or “PI System Connection Data Source”, residing within ADH that receives the data from the on-premises PI to ADH Agent and stores it in SDS. Note that storage in SDS is partitioned by ADH namespace.

The deployment of a AVEVA™ PI to ADH Agent establishes a 1 to 1 connection from a on-premises source AVEVA™ PI Server to an ADH PI System Connection.

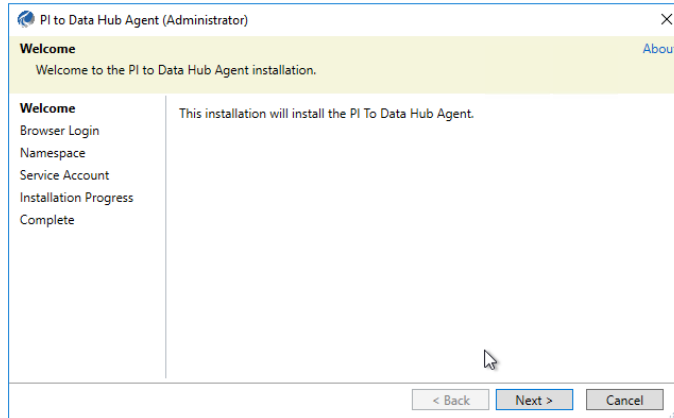
The AVEVA™ PI to ADH Agent must be installed by a user who has administrative privileges on the local computer as well as in your ADH account.

AVEVA™ PI to ADH Agent requires the following permissions on PI Data Archive:

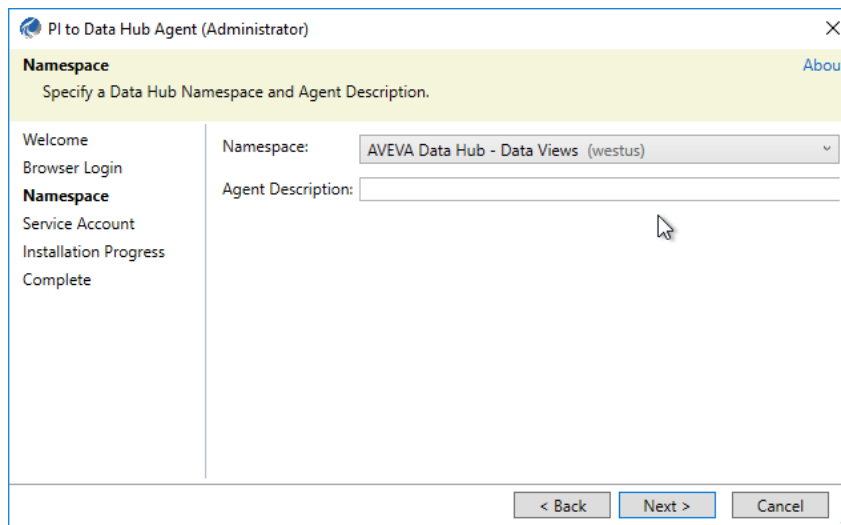
1. Read access to Archive Data (the PIARCDATA Security Table)
2. Read access to the PI Points configuration (the PIPOINT Security Table)
3. Read access to all the PI Points that you are interested in transferring
4. Read access to all the Data of the PI Points that you are interested in transferring

### i. Step-by Step Installation Process

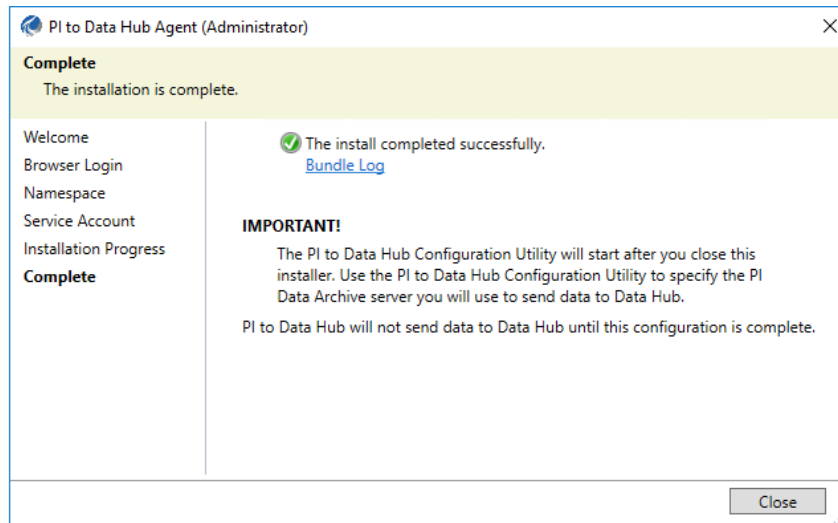
1. Once downloaded, double-click or “Run as Administrator the installation kit to install the AVEVA™ PI to ADH Agent:



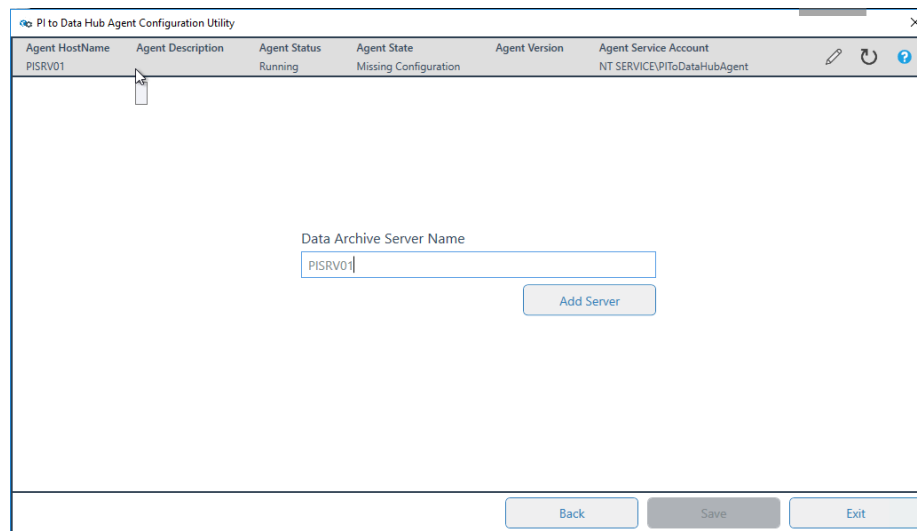
2. Click Next and enter the Account Id or Company Alias. In this case, it is AVEVATraining. Click Next.
3. A login prompt opens for you to sign in your ADH account. The user you choose must have Administrator privileges in your ADH account.
4. Once logged in, from the top dropdown menu, select the Namespace in which you created the PI System Connection and select the corresponding Connection. In this case, the Namespace is **AVEVA Data Hub – Data Views** and the Connection is **Data from feed environment 434**.



Enter the Service Account for the PI to ADH Agent. This account must have access to the PI Data Archive with the privileges previously mentioned. For this course, we will leave it as the default NT Service\PtoDataHubAgent account. Then click on the Install button.

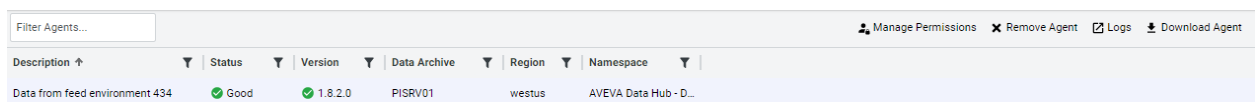


The Agent will then open a Configuration Agent were to specify the PI System



Next, enter the name of the PI Data Archive from which you want to transfer data from. In this case, it is **PISRVO1**. Then click Next.

The installation should complete successfully and the PI to ADH Agent should register to the PI System Data Source in ADH. The registration process may take several minutes.



open Windows Services (services.msc) and verify that the service has been installed, started and is running with the service account provided:



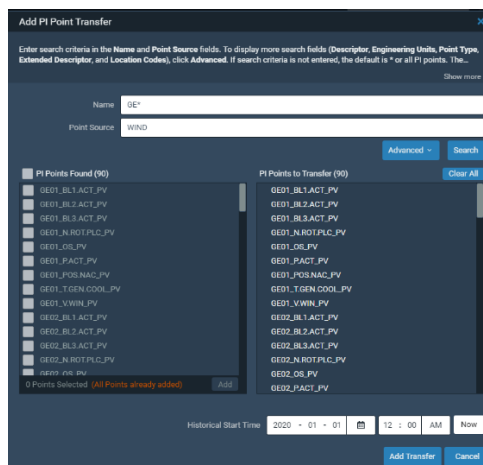
## c. Creating a Data Transfer

Once you have installed and registered the AVEVA™ PI to Data Hub Agent application, you can create a Transfer of data from the AVEVA™ PI System to the namespace you selected when creating the connection. It can take several minutes after installing the AVEVA™ PI to ADH Agent application for the on-premises agent to become available in the connections page.

*Note: A single instance of the PI to ADH Agent application can only support a single connection, which can only be the connection you create prior to installing the application. To create another connection (from the selected PI system to a different namespace, for example), you would have to create that connection, and then install another instance of the PI to ADH Agent application.*

To transfer data with the new connection, click the Add PI Point Transfer button. The Add PI Point Transfer dialog displays. In the Add PI Point Transfer dialog, enter the Point Name Mask and Point Source Mask fields. In this case the Point Name Mask is GE\* and the Point Source Mask is WIND. Click on Search. Check the PI Points Found box to select all 90 PI Points, then click the Add button.

Set the Historical Start Time to 2022-01-01 at 12:00 AM. Click Add Transfer to transfer the PI Points returned in the query.



The progress of the transfer displays in the Data Transfer dialog in the Details tab.

*Note: To transfer a set of PI Points different from those returned by the query, you must create a new query that groups in the desired PI Points.*

Click Start to start the Data Transfer or Stop to stop the Data Transfer. After historical data transfer has completed, current data will continue to be transferred. To create a different Data Transfer, you must first stop and remove the current Data Transfer. Note that this will not remove any data already transferred.

**Data Transfer** View Stop

Description	PI Points: 90
Status	Started
Current Activity	Sending Streaming Data
Events Per Second	Historical 0 Streaming 7.6
Last Streaming Read	Aug 28, 2020, 10:56:09 AM
Historical Transfer	<div style="width: 100.0%;"><div style="width: 100.0%;"></div></div> 100.0%
Historical Start	Jan 1, 2020, 12:00:00 AM
Historical End	Aug 28, 2020, 10:35:39 AM

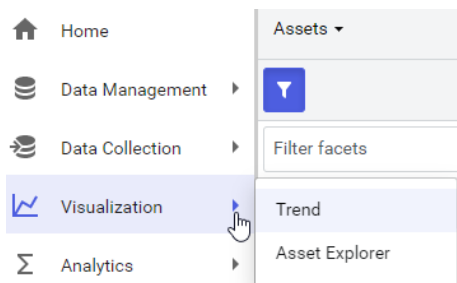
Once the historical transfer is at 100%, we can go to the Sequential Data Store under the Data Management section to view the transferred data.

## 7. Appendix B: Creating an asset rule

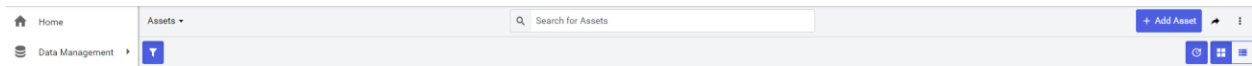
### a. Creating a single asset

The first step is to define the asset context. An asset type needs to be defined. The easiest way is to create a single asset manually. Add data and then use this asset as a template for an asset type.

We create an asset by navigating to Visualization / Assets Explorer



Then click Add Asset



Choose Type <None>

It will be called Turbine

A fixed metadata referring to the Wind farm will be added

## Turbine

⋮ Cancel

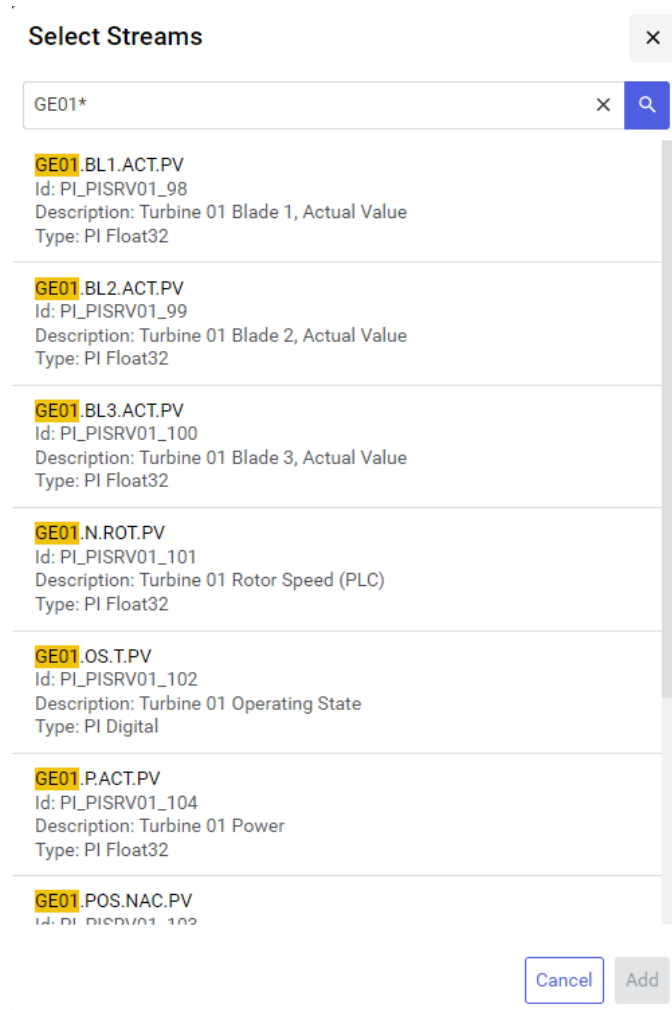
Asset	<input type="text" value="Turbine"/>
Id	9d0abfbe-a1dd-4cf8-be2c-443941cd8775
Asset Type	<None>
Description	<input type="text" value="Asset Description"/>

Metadata Properties Status

Add Metadata

Metadata	Value	Type	UOM
<input type="text" value="Farm"/>	<input type="text" value="Big Buffalo"/>	<input type="text" value="String"/>	<input type="text" value="none"/>

Next we will add properties by navigating to the properties pane. By querying GE01\* we will get all data streams associated to turbine GE01 which are representative of all GE turbines in the park as in PI on Prem they all are based on the same template and use a consistent naming pattern. Select all streams and press Add



The result will use the stream names a reference which are for us not clear enough.

We will change the names according to this list:

GE01.BL1.ACT.PV	Blade1 Actual Value
GE01.BL2.ACT.PV	Blade2 Actual Value
GE01.BL3.ACT.PV	Blade3 Actual Value
GE01.N.ROT.PV	Rotor Speed
GE01.OS.T.PV	Turbine State
GE01.P.ACT.PV	Active Power
GE01.POS.NAC.PV	Nacelle Position
GE01.TGEN.COOL.PV	Generator Cooling Air Temperature
GE01.V.WIN.PV	Wind Speed

## Prior

Metadata Properties Status

**Stream References**

GE01_BL1.ACT_PV	
GE01_BL2.ACT_PV	
GE01_BL3.ACT_PV	
GE01_N.ROT.PLC_PV	
GE01_OS_PV	
GE01_P.ACT_PV	
GE01_POS.NAC_PV	
GE01_T.GEN.COOL_PV	
GE01_V.WIN_PV	

## After

Blade1 Actual Value	
Blade2 Actual Value	
Blade3 Actual Value	
Rotor Speed	
Turbine State	
Active Power	
Nacelle Position	
Generator Cooling Air Temperature	
Wind Speed	

After saving we have created a single asset

OCS Data Views PI Worl... Assets Search for Assets + Add Asset

































































Filter facets

**Status**

- Good
- Warning
- Bad
- Unknown

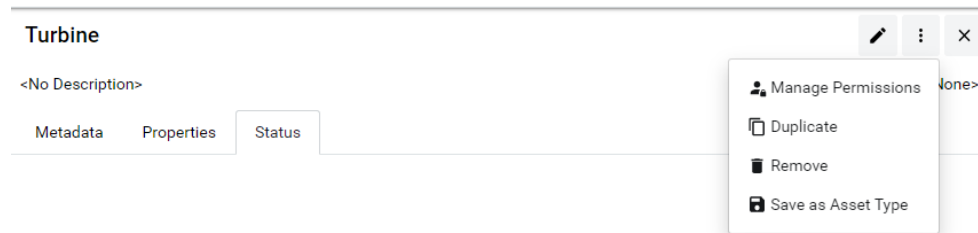
Turbine

The operating conditions above are defined by mapping rules which can be in the Status Tab

Value		Status	
Load Operation	→	  	
Idling Mode	→	  	
Feathering Position	→	  	
Idle Position	→	  	
Running Up	→	  	
Shutdown	→	  	
No Data	→	  	
Bladeangle Set	→	  	
Bladeangle Plus	→	  	
Bladeangle Minus	→	  	
Bladeangle Stop	→	  	
Bladeangle Float	→	  	
Man. Speed	→	  	
Man. Torque	→	  	
Load Shutdown	→	  	
Grid Disconnection	→	  	

## b. Elevating it to Asset Type

After selecting the new asset can be transformed to an Asset Type by clicking on the tree dot menu and saving it as asset type and pushing create



**Create an Asset Type for Turbine** [X]

Name:

Description:

### c. Applying the asset type to all turbines

In this step we will use the type to automatically derive all assets existing by analyzing the data streams provided. An Asset rule will do that for us.

Navigate to Data Management / Asset Management and click new Asset rule. We will use the asset type defined previously

**Create New Asset Rule** [X]

Name:

Description:

Asset Type:

In the next steps we will extract information out of the data streams which help us to define the assets needed. This information is stored in Tokens which can then be linked to data fields.

We will select any stream as we do have a good naming pattern for our streams as example

GE01.BL1.ACT.PV

The yellow part of the stream name is the Asset Name, while the green part is the measurement.

The goal is now to tell the rule to extract this information

We chose all letters preceding "." and allocate it to token "name"



GE01.BL1.ACT.PV

---

**1. Match:**  letters or numbers preceding the delimiter "."  
 the next 4 characters of any type (including letters, numbers, and symbols)  
 the next 4 letters or numbers  
 the next group of letters or numbers  
 the string literal "GE01"

**and name it:**  ×

[Capture](#)

Similar to for the measurement. You may click on the stream name to get more options

GE01.BL1.ACT.PV

---

**3. Match:**  letters or numbers preceding the delimiter "."  
 the next 3 characters of any type (including letters, numbers, and symbols)  
 the next 3 letters or numbers  
 the next group of letters or numbers  
 the string literal "BL1"

**and name it:**  ×

[Undo Capture](#) [Capture](#)

What remains is of no use an will be defined as a token3 and literal

Stream Name

GE01.BL1.ACT.PV

---

1. Match letters or numbers preceding the delimiter "." (name) - GE01

2. Match the delimiter "."

3. Match letters or numbers preceding the delimiter "." (measurement) - BL1

4. Match the delimiter "."

5. Match:  letters preceding the delimiter "."

- the next 3 characters of any type (including letters, numbers, and symbols)
- the next 3 letters
- the next group of letters
- the string literal "ACT"

and name it:

Undo Capture Capture

GE01.BL1.ACT.PV

---

r"." (name) - GE01

r"." (measurement) - BL1

{token03} - ACT

7. Match:  letters until the end of the stream name

- the next 2 characters of any type (including letters, numbers, and symbols)
- the next 2 letters
- the next group of letters
- the string literal "PV"

String literal matches are not available as tokens

Undo Capture Capture



We then need to map the values to the measurement

1 Extract Tokens

2 Map Tokens to Values

### Configure Stream Reference Name Token

Select the token which contains the measurement ?

{Measurement} - BL1.ACT  

### Token Mappings Status

Specify the token values for each of the tokens listed below.


- {Name} - GE01 ✓
- {Measurement} - BL1.ACT ✗

### Token Value Mappings

Token Value of BL1.ACT for Token Measurement from Stream Name - GE01.BL1.AC

? For an Asset Rule linked to an Asset Type, the 'Use Existing Token Values' of Reference Names from the Asset Type. Instead, use the 'Rename Token Val

- Use Existing Token Values
- Rename Token Values

 Generate Mappings  + Add Mapping  Remove All Mappings

No mappings added.

The system will search all streams and propose

### Token Value Mappings

Token Value of BL1 for Token measurement from Stream Name - GE01.BL1.ACT.PV.

- Use Existing Token Values
- Rename Token Values

 Generate Mappings  + Add Mapping  Remove All Mappings

#### Mappings

BL1	→	Map To...
BL2	→	Map To...
BL3	→	Map To...
N	→	Map To...
OS	→	Map To...
P	→	Map To...
POS	→	Map To...
TGEN	→	Map To...
V	→	Map To...

Using the mapping list to fill out above we get clearer names.:

BL1	Blade1 Actual Value
BL2	Blade2 Actual Value
BL3	Blade3 Actual Value
N	Rotor Speed
OS	Turbine State
P	Active Power
POS	Nacelle Position
TGEN	Generator Cooling Air Temperature
V	Wind Speed

The tokens are then mapped to the asset Metadata

### Configure Asset

Type Name <None>

Id {name}

Name {name}

Description Type '{' to display tokens

Stream Reference Name {measurement} ✎

Metadata [Add Metadata](#)

And we will get a preview of the results to expect

Id	Type	Name	Farm	Active Pow	Rotor Spee	Blade1 Act	Blade2 Act	Turbine Sta
GE01	GE Turbine	GE01		GE01.P.ACT...	GE01.N.RO...	GE01.BL1.A...	GE01.BL2.A...	GE01.OS.T...
GE02	GE Turbine	GE02		GE02.P.ACT...	GE02.N.RO...	GE02.BL1.A...	GE02.BL2.A...	GE02.OS.T...
GE03	GE Turbine	GE03		GE03.P.ACT...	GE03.N.RO...	GE03.BL1.A...	GE03.BL2.A...	GE03.OS.T...
GE04	GE Turbine	GE04		GE04.P.ACT...	GE04.N.RO...	GE04.BL1.A...	GE04.BL2.A...	GE04.OS.T...
GE05	GE Turbine	GE05		GE05.P.ACT...	GE05.N.RO...	GE05.BL1.A...	GE05.BL2.A...	GE05.OS.T...
GE06	GE Turbine	GE06		GE06.P.ACT...	GE06.N.RO...	GE06.BL1.A...	GE06.BL2.A...	GE06.OS.T...
GE07	GE Turbine	GE07		GE07.P.ACT...	GE07.N.RO...	GE07.BL1.A...	GE07.BL2.A...	GE07.OS.T...
GE08	GE Turbine	GE08		GE08.P.ACT...	GE08.N.RO...	GE08.BL1.A...	GE08.BL2.A...	GE08.OS.T...
GE09	GE Turbine	GE09		GE09.P.ACT...	GE09.N.RO...	GE09.BL1.A...	GE09.BL2.A...	GE09.OS.T...
GE10	GE Turbine	GE10		GE10.P.ACT...	GE10.N.RO...	GE10.BL1.A...	GE10.BL2.A...	GE10.OS.T...

At the end all asset are found and the values associated

Assets ▾  + Add Asset


Filter facets


Status  Good  Warning  Bad  Unknown

Asset Type  Turbine

Farm  Big Buffalo

GE01	✓	GE02	✓	GE03	✓	GE04	✓	GE05	✓
GE06	✓	GE07	✓	GE08	✓	GE09	✓	GE10	✓

 <b>Best Practice</b>	<p><i>Stream names which follow a standard pattern are more applicable to the creation and use of asset / metadata rules. The flexibility and complexity of metadata pattern definition will be further extended in future ADH versions.</i></p>
---	--

 <b>Tip</b>	<p><i>The value of metadata itself lies in its capacity to enrich sequential data, and to facilitate logical segregation and contextualization. Other services and applications, such as ADH data views, leverage stream metadata to simplify finding data and to providing context.</i></p> <p><i>Please reach out to OSIsoft if there are any questions, comments or concerns regarding metadata management in ADH.</i></p>
---	---

## 8. Appendix C: Configuring a New ADH Client

On the Clients page under the Security section of the Navigation menu, click on the +Add Client button. Set the Client Type as Client-Credentials and Account. The Client Name entered is OSIssoft Learning Client, select the Account Administrator checkbox, then click Continue.

**Add Client** [X]

Configure a new client credential client that will allow authentication as a headless client application using the **Client Id** and **Client Secret** provided on client creation. The **Token Lifetime** represents how long the access token will function before it expires. The default (and max) value is 3600 seconds (1 hour). The minimum value is 60 seconds (1 minute).

Name  
Test

Roles

- Tenant Administrator
- Tenant Contributor
- Tenant Data Steward
- Tenant Member
- Tenant Viewer

Token Lifetime  
3600

Cancel Continue

On the Add Secret window, enter the Description as below, then check the Never Expires box, then click Add:

### Add Secret ✕

To add a secret, specify description and select an expiration date for the secret and click **Add**. Note that the expiration date must be in the future.

Description (Optional)

Expiration Date

  :    
 Never Expires

Cancel Add

Make note of the Client Id and the Client Secret, as they are needed for accessing ADH programmatically from an external application.

### Client Successfully Created ✕

Client Id and Client Secret displayed below can be used to authenticate.

**⚠ Client Secret will no longer be accessible once you close this dialog. Please make a record of it.**

**Client Id** 📄  
60734f56-c167-4416-a5fe-4ae268033cda

---

**Client Secret** 📄



## 9. Appendix D: Creating an OMF Connection in ADH

Another method to send data to ADH is via the OSIsoft Message Format or OMF protocol. But first, a new OMF Connection needs to be created in ADH, so that we can use this channel to send data streams to the SDS via OMF.

Click on Connections in the Navigation menu>Data Management:



On the Connections page, select OMF as the Type and click on +Add Connection. Enter **Wind Turbine Data** for both the Name and if needed a Description.

Click Next and select the PI World Client for Data View Lab from the list.

Click Save to complete the process of creating a new OMF Connection.

## Add OMF Connection

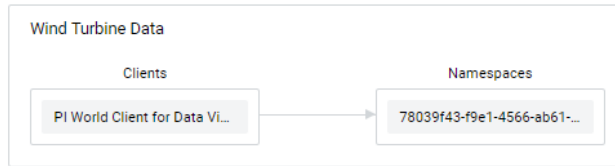
Review the connection summary information below before saving.

1. Details

2. Clients

3. Namespaces

4. Review



Cancel Back Save

## The OMF Endpoint will appear in the list

Connection/Clients	Description	Namespaces
> Wind Turbine Data		78039f43-f9e1-4566-ab61-808a20556c23

Overview

Id: d6beef7-789d-4009-acc8-02a5b06037ac  
Name: Wind Turbine Data  
Topic Namespace: 78039f43-f9e1-4566-ab61-808a20556c23  
Type: OMF

Data Flow

For step by step details regarding creating an OMF Connection, please refer to the following OSisoft YouTube video: <https://www.youtube.com/watch?v=52lAnkGC1IM>

## 10. Appendix E: References

For more information regarding the resources utilized in this course, please see the links below:

1. Wind Turbine Background:
  1. <https://www.awea.org/wind-101/basics-of-wind-energy> N/A
  2. Figure 2: [http://www.wind-power-program.com/turbine\\_characteristics.htm](http://www.wind-power-program.com/turbine_characteristics.htm)
  3. Figure 3: Badran, Omar & Abdulhadi, Emad & Mamlook, Rustom. (1992). Evaluation of parameters affecting wind turbine power generation.
2. AVEVA™ Data Hub: <https://datahub.connect.aveva.com>
3. OCS API Documentation: <https://docs.osisoft.com/bundle/ocs-docs-main/page/ocs-content-portal-overview.html>
4. Jupyter Notebook: <https://jupyter.org/>
5. Python Help: <https://www.python.org/>
6. OCS Python Library Documentation: [https://github.com/osisoft/OSI-Samples-OCS/tree/master/basic\\_samples/SDS/Python/SDSPy/Python3](https://github.com/osisoft/OSI-Samples-OCS/tree/master/basic_samples/SDS/Python/SDSPy/Python3)
7. Scikit-learn Python Package for Machine Learning: <https://scikit-learn.org/>
8. Decision-Tree Regressor Machine Learning Algorithm:  
<https://gdcoder.com/decision-tree-regressor-explained-in-depth/>  
<https://scikit-learn.org/stable/modules/tree.html#regression>
9. OpenWeather API: <https://openweathermap.org/api>
10. OMF Documentation: <https://docs.osisoft.com/bundle/omf/page/index.html>

Your opinion is important to us, please help us improve by openly providing feedback on your experience in the labs at AVEVA World.

