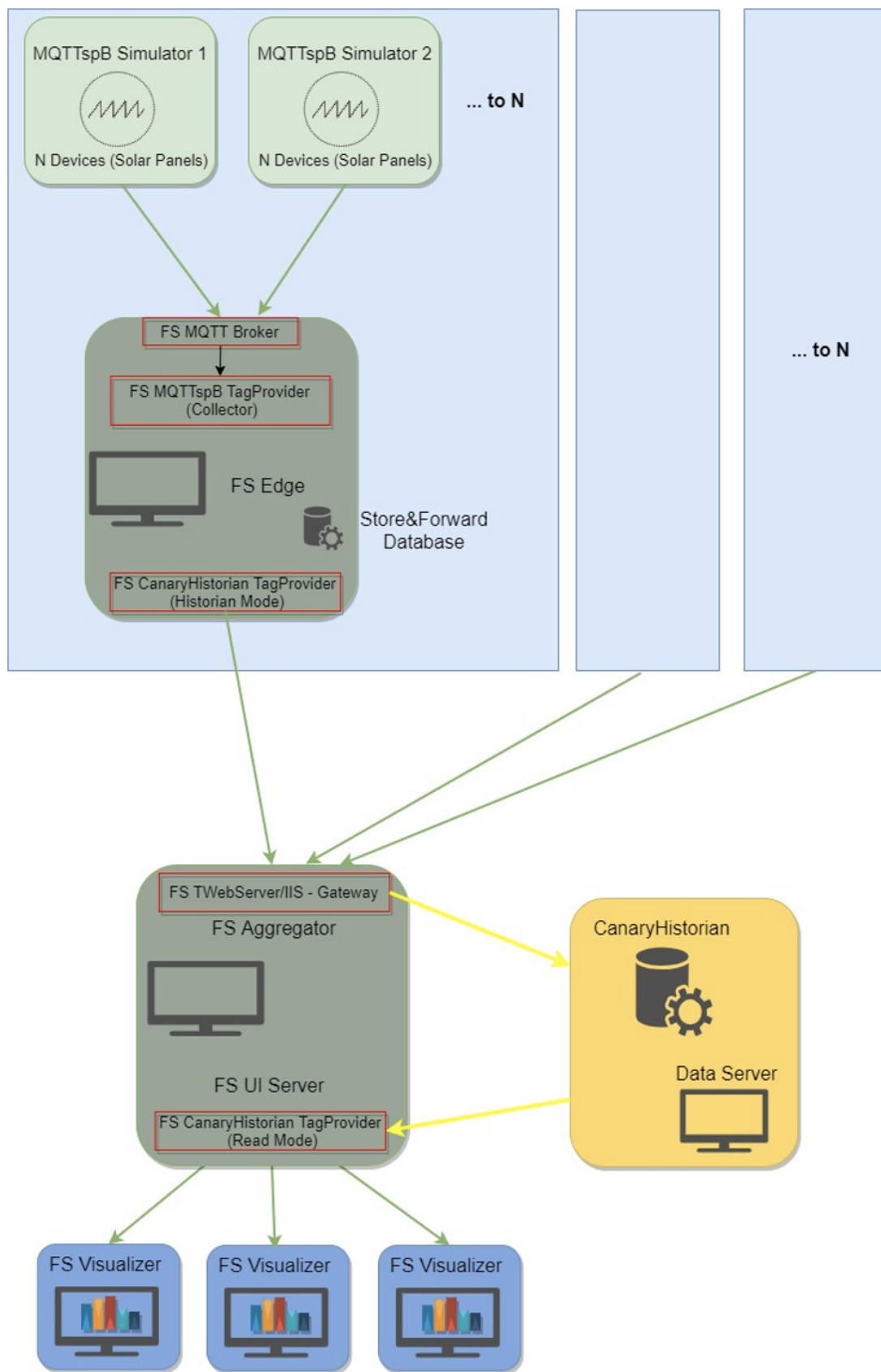


Demo Project for FactoryStudio 9.2

This document has information on the advanced Demo Project (Solar Panel) developed for FrameWorkX FactoryStudio's version 9.2 Release. [Download here.](#)

Overview

The Demo architecture is illustrated in the image below.



The MQTTSpB Simulators are responsible for generating data for various Solar Panel structures and publishing them to a Broker.

An Edge Project will connect to that Broker and read all data published by the Solar Panels (simulators) using the TagProvider for MQTTspB Protocol. This data will then be transferred to a Historian Database using the TagProvider for CanaryHistorian in Historian Mode.

Finally, there are User Interface Projects that connects to the Historian Database through the TagProvider for CanaryHistorian (in read mode) and display the data in dynamically.

Generating and Historizing Data

Setting Up Project for Data Storage

The Project *fxEdgeCollector*, will act as a Gateway for the data generated by an MQTTspB Simulator to the CanaryHistorian through an MQTTspB Broker.

The configuration is done without any FactoryStudio Tags, using only the TagProvider Unified Namespace (see more information on this topic on its [documentation](#)).

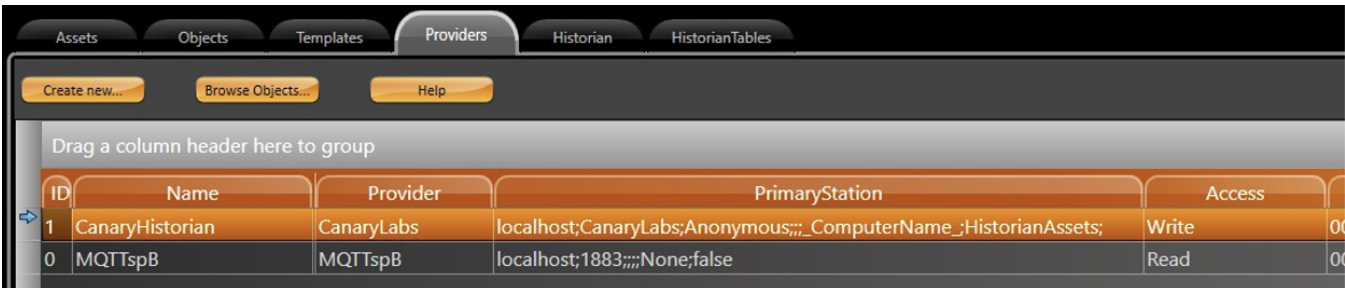
In order to ensure you have the correct behavior, check if the following settings are correct:

At **Edit Tags Providers MQTTspB**, change the connection parameters according to your Broker. You can use Tatsoft's default one (TMQTTBroker).

For more information on the MQTTspB Protocol and TMQTTBroker, please refer to their documentation. [MQTTspB](#) and [TMQTTBroker](#).

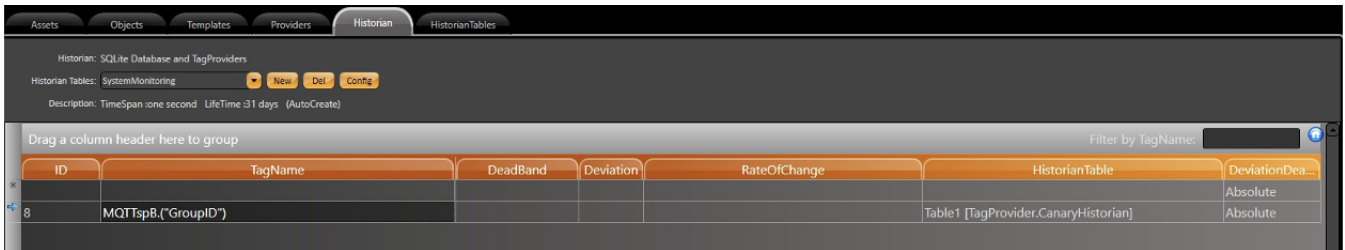
At **Edit Tags Providers CanaryHistorian**, map the Server Name and View to the Historian Server that will be used.

The Dataset mapped don't need to pre-exist on your Server, as it can be auto-created by this application. For more information on the CanaryHistorian Protocol, please refer to its documentation ([CanaryHistorian](#)).



ID	Name	Provider	PrimaryStation	Access	
1	CanaryHistorian	CanaryLabs	localhost;CanaryLabs;Anonymous;;;_ComputerName_;HistorianAssets;	Write	00
0	MQTTspB	MQTTspB	localhost;1883;;;None;false	Read	00

At **Edit Tags Historian**, you will see an entry from the MQTTspB Provider, mapping a Dynamic Tag to be By mapping the GroupID with the syntax MQTTspB.("GroupID") all child elements (NodeIds, DeviceIds and attributes) will be automatically historized as well.



ID	TagName	DeadBand	Deviation	RateOfChange	HistorianTable	DeviationDea...
8	MQTTspB.("GroupID")				Table1 [TagProvider.CanaryHistorian]	Absolute



There is a setting that determines how quickly the CanaryHistorian service rebuild their views cache when new tags or metadata arrive. That setting is in the [CanaryViews.exe.admin](#) file located in the installation location under the Views subfolder.

The parameter is called **lockoutTimeForRecaching** and the default is 5 minutes. The Views service only reads this setting when the service is starting. So, any manipulation to the value would require a restart of the Views service. The installation will try to modify this configuration to 0.

MQTTspB Simulator for Data Generation

This section details the necessary configuration regarding the MQTTspB Simulator and configuration files. Make sure you have a file called *MQTTspBSimulator-Demo9.2.exe.config* inside the same directory as the sample projects. This file contains the information required for customizing the simulator to create Tags in a specific format.



Important

DO NOT change the configuration file name.

The syntax for the configuration file is described below:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
<configSections>
<section name="Barcelona" type="System.Configuration.AppSettingsSection" />
<section name="Bilbao" type="System.Configuration.AppSettingsSection" />
<section name="Madrid" type="System.Configuration.AppSettingsSection" />
<section name="Sevilha" type="System.Configuration.AppSettingsSection" />
</configSections>

<Bilbao>
<add key="NumberOfDevices" value="2" />
<add key="Name" value="Type=string;Sampling=constant;range=;defaultValue=Bilbao" />
<add key="State" value="Type=int;Sampling=variable;range=0,10;defaultValue=" />
<add key="Longitude" value="Type=double;Sampling=constant;range=-3.23,-2.456;defaultValue=" />
<add key="Latitude" value="Type=double;Sampling=constant;range=43.30, 43.341;defaultValue=" />
<add key="PanelPower" value="Type=double;Sampling=variable;range=200,500;defaultValue=" />
<add key="PanelVoltage" value="Type=double;Sampling=variable;range=-220,220;defaultValue=" />
<add key="PanelCurrent" value="Type=double;Sampling=variable;range=-10,10;defaultValue=" />
<add key="TemperaturePort" value="Type=double;Sampling=variable;range=15,35;defaultValue=" />
</Bilbao>
<Barcelona>
...
</Barcelona>
<Madrid>
...
</Madrid>
<Sevilha>
...
</Sevilha>
</configuration>
```

Each **section** defined in **configSections** will be a **NodeId** that is created in the Broker. Their attributes are defined in their own section.

The **NumberOfDevices** key represents the amount of DeviceIds created for that specific NodeId. You can change that quantity by changing this value.

To launch the simulator with this custom configuration file, run the executable file with the path for the config file as a parameter.

The Simulator can easily be executed, through the ProjectAdmin Page in *fxAssetsMonitor* Project.

MQTTspB Simulator

Connection String Parameters

Broker URL:

127.0.0.1

Port:

1883

Username:

Password:

Disconnect

Connect

Simulation Settings

Base GroupID Name:

GroupID

Base NodeID Name:

NodeID

Number Of Nodes:

1

Base DeviceID Name:

Panel

Number Of Devices per Node:

1

Total Tags: 32

Apply

Runtime Simulation Settings

Send one Data Sample

Send Continuous Data

Show Log

Status: Connected with Success -- Birth Messages sent

You can now simulate new values, for the Variable sampling Type of Attributes by clicking on the Send buttons. and Add/Remove all NodeIds from the Broker by forcing a Connection/Disconnection.

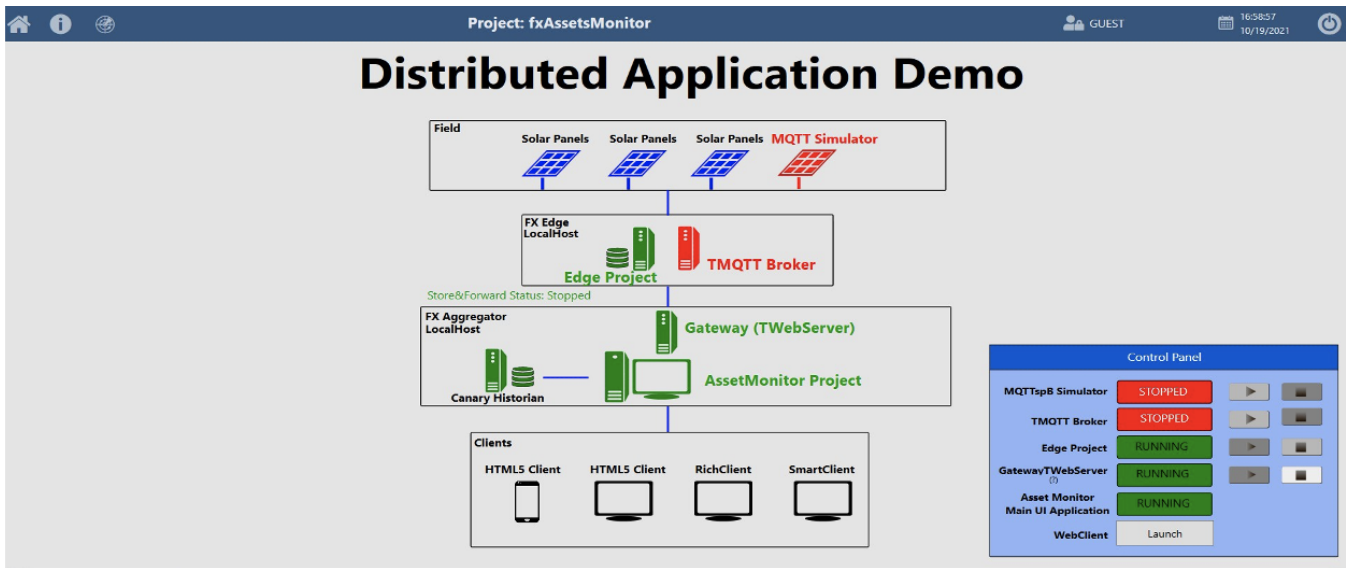
UI and Data Visualization

The User Interface and Data Visualization portion of this demo consists in a Project, [fxAssetsMonitor](#), with a .Net Display to administrate the project applications and a HTML5 and WPF Displays for the SolarPanel assets, loaded from CanaryHistorian, in the AssetsControl and Map component.

Project Administrator Display in .Net

In this Display, you can monitor the status of your applications (Visualizer and Collector Projects, TMQTTBroker, MQTTspBSimulator and TWebServer) and Start/Stop them.

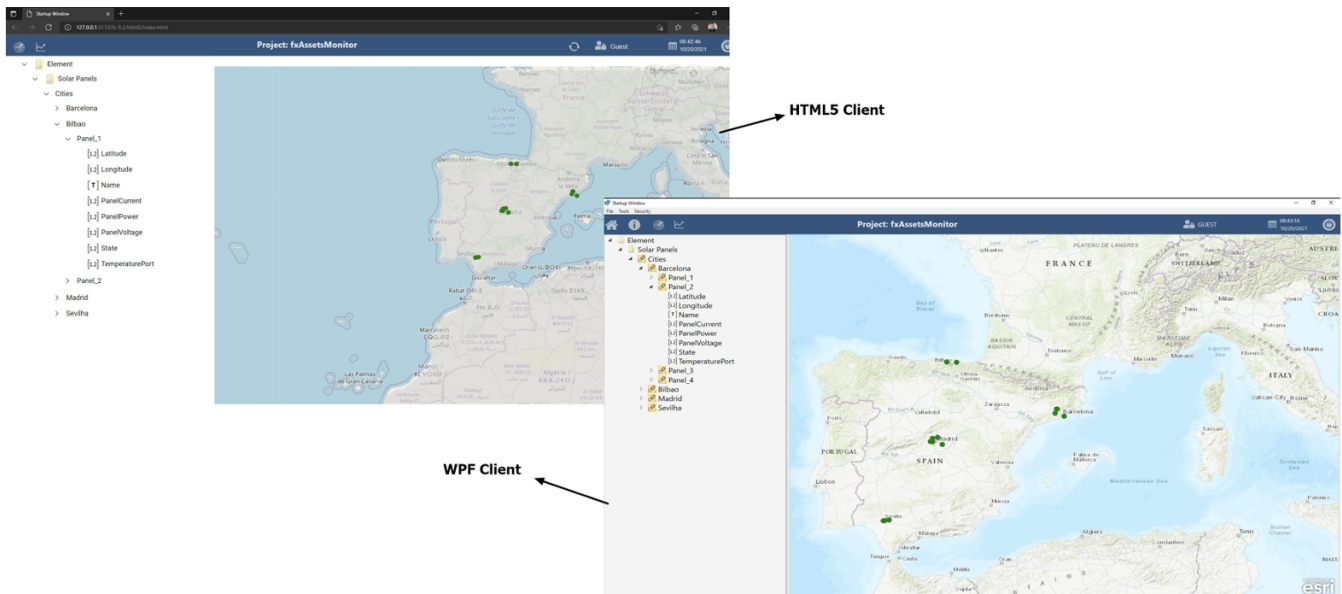
There is also a button to launch the HTML5 WebClient in [MsEdge](#) Browser.



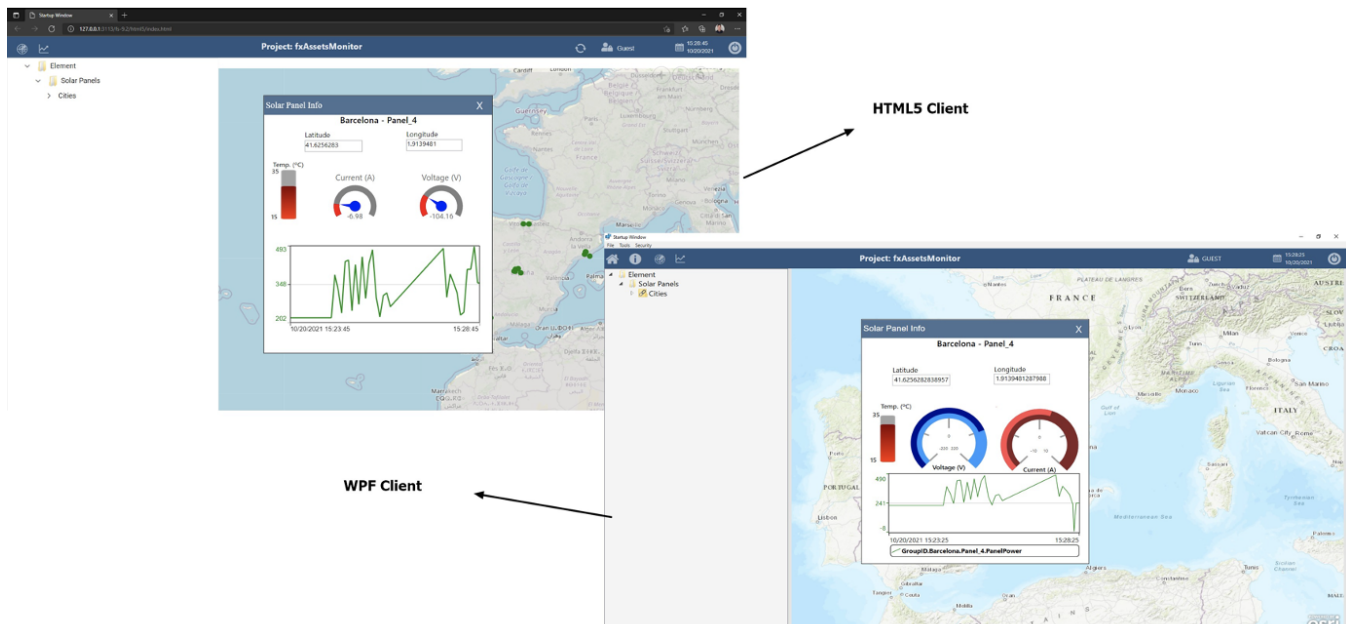
Data Visualization

The visualization portion of this project is available both in WPF and HTML5 clients. Despite using different technologies (.Net Framework and Javascript), their behaviors and components are quite similar. Below you will find some images showing both clients.

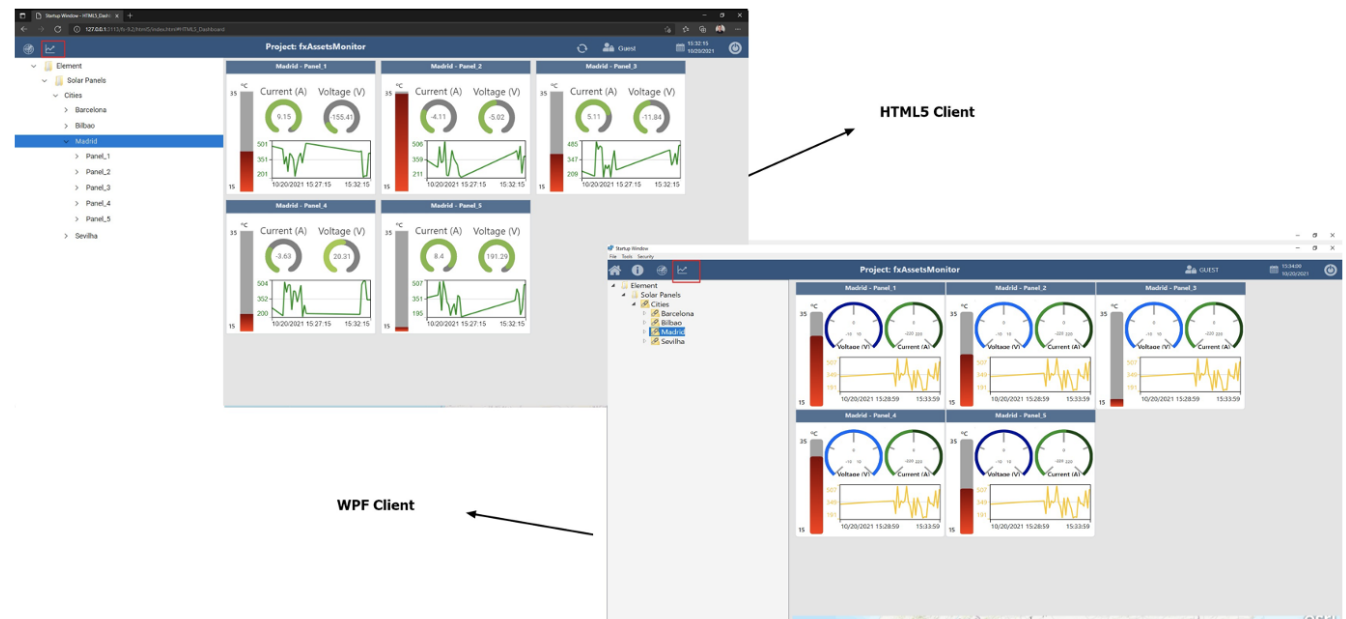
If your *fxEdgeCollector* configuration was done correctly and you have all necessary applications running, you should see some markers added to the Map components. They are loaded from the CanaryHistorian Dataset (Latitude and Longitude coordinates defined in the MQTTSpBSimulator).



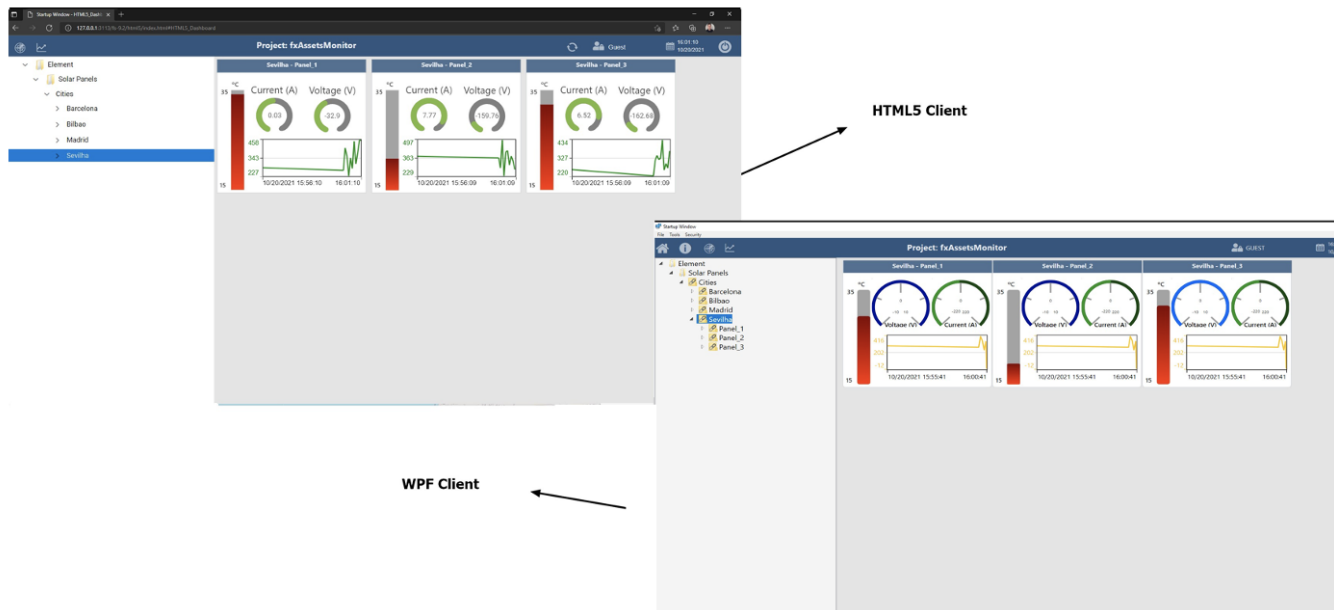
You can expand the AssetControl Tree to see the inner elements and Click on the Marker to reveal some detailed information on that specific SolarPanel, as illustrated below.



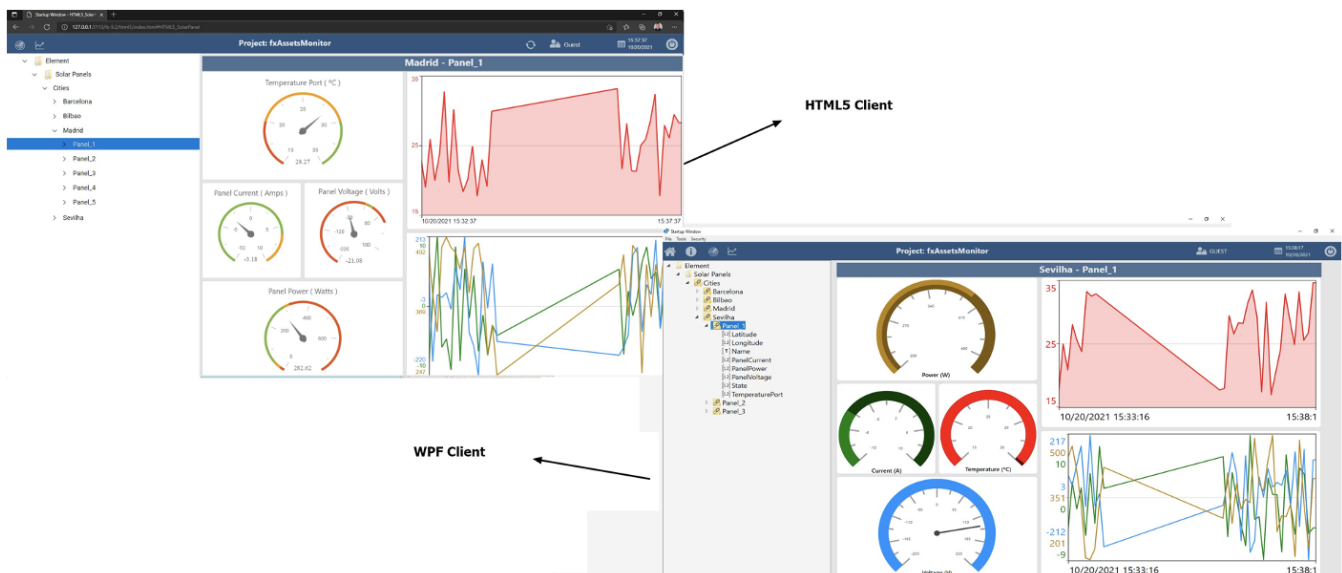
Navigating further down the AssetControl Tree under the Cities node, you will see the Cities that were created (Barcelona, Bilbao, Madrid and Sevilha, by default). Select one of the Cities and click on the Trend Icon (1) to open a Dynamic Window containing information on all Panels from that specific location.



The image above is showing all 5 (five) panels from Madrid. The same Dynamic Display is automatically adjusted for a different quantity when Sevilha is selected, for example.



Going one level down, inside the Panels from a City, we can monitor the attribute values from that specific SolarPanel.



Gateway/TbWebServer

The TWebServer, our built-in WebServer application, not only is responsible for allowing Client connections through HTTP requests but also enabling your CanaryHistorian to forward Data into a different server.

When using this feature, if the connection to the Server you are pointing to is down, the application can store the data locally to be synchronized once the connection is established again.

At the *fxEdgeCollector* Project, under **Edit Tags Providers**, you will find a column named **ServerIP**. In this field you can enter an IP Address pointing to a server that contains a CanaryHistorian running. This Server **must** have a TWebServer running in order to be accessible.

In this demo, you can test this feature by entering the local loopback IP Address **127.0.0.1**. You can force the application to store data to be synchronized by stopping the TWebServer at the Project Admin Display. The data should be forwarded once the TWebServer is running again.

Modifying between Canary Historian and SQLite Historian

On fxEdgeCollector project:

- Modify the MQTT.(GroupID) to be linked with the SQLite Historian or Canary Historian using the column HistorianTable.

On fxAssetsMonitor project:

- Configure the startValue of Tag.ProjectAdmin.IsCanaryUsed to 0 (SQLite) or 1 (CanaryHistorian).
- Modify the Asset configuration to use the SQLiteHistorianByMQTT or CanaryHistorian Inside the "Element/Solar Panels/" clicking with the right button insert the TagProvider SQLiteHistorianByMQTT.("GroupID") or CanaryHistorian.("GroupID"). Important: only one tag provider is expected to be inside the "Element/Solar Panels/" level, so delete the TagProvider not desired from the asset configuration.
- Modify the DisplayText of SQLiteHistorianByMQTT.("GroupID") or CanaryHistorian.("GroupID") to be "Cities"