MQTT SparkplugB Communication Protocol

Unified Namespace and Asset Model

Contents

1 Introduction

This document contains information on how to access elements from an MQTT Broker data model in any place of the Engineering Environment and how to access the entire, or just a part, of the Asset Model Structure.

2 System Requirements

In order to use this feature, you need to make sure some requirements are matched.

- **Factory Studio - FrameworX product version *fs-9.1.12***.
- **Devices → Channel and Node created for MQTTspB protocol**.

2.1 Device Configuration

At **Devices → Channels** Tab, create a new *Channel* for MQTTspB protocol. Configure the ProtocolOptions field with the necessary information. Make sure the **Type** is set for **Application Node**.

At **Devices → Nodes** Tab, create a node, assign it to our newly created channel, and fill the *Station* parameters in order to connect to your Broker.

There is no need to create communication Points.

3 Unified Namespace

3.1 Context

Unified Namespace is a software solution that acts as a centralized repository of data, information and context where any application or device can consume or publish data needed for a specific action.

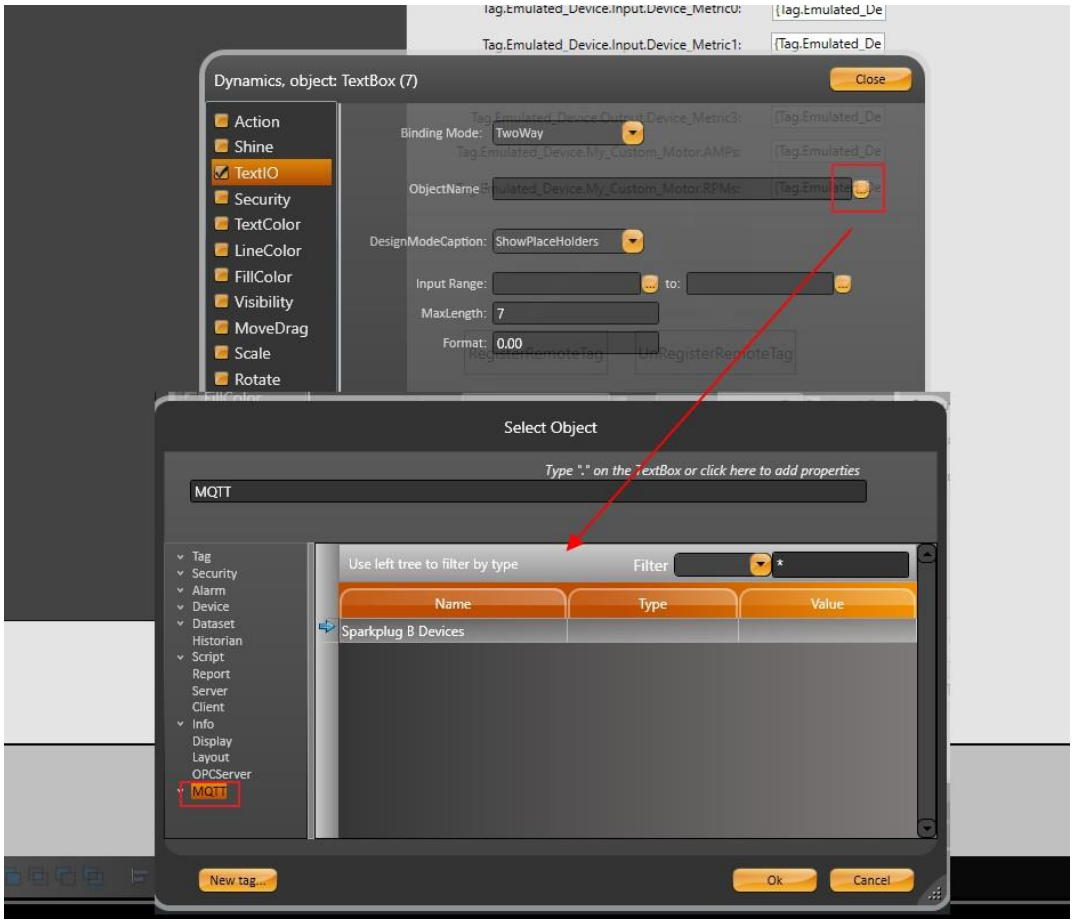
It allows users to collect data from various sources, and transform it to a format that other systems can understand. Without a centralized data repository, it could take months to deploy a new analytics application across the entire enterprise versus hours with a unified namespace.

3.2 Access Elements in Engineering

Once you have a successful connection established to an MQTT Broker, as detailed in the requirements section, you can start using the available variables from your Data Model in various places within your Project.

In the **Draw** Environment, pick up a TextBox element and insert in your display. Double-click on it and add a TextIO Dynamic as it was done for showing Tag values.

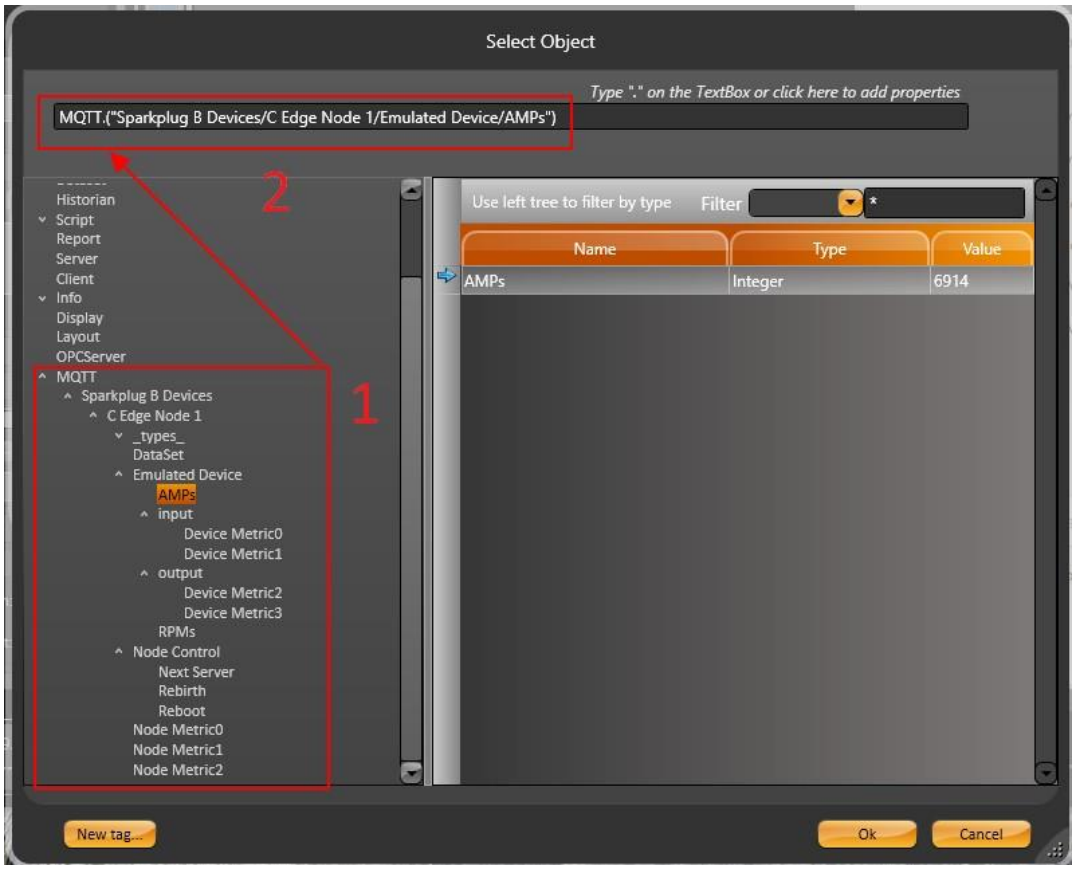
On the **ObjectName** field, select the Browse button. At the dialog window, you should see a list of objects you can choose from. At the bottom of the list there will be **MQTT**



Object Selection.

There is an expand option for the **MQTT** object. By clicking on it, you can browse through all existing elements that are connected to your Broker (1).

The expression field will be filled with a syntax: MQTT.(”⟨ *Address In MQTT* ⟩”) (2).



MQTT Data Model.

By doing that, you can display the information from this Communication Protocol directly in you Display, without the need to create a Tag and Communication Point.

This feature usage is not limited to Displays. You can create AlarmItems, store in Historian Tables, and even use this MQTT variable as a Communication Point to Write Data into a different Protocol.

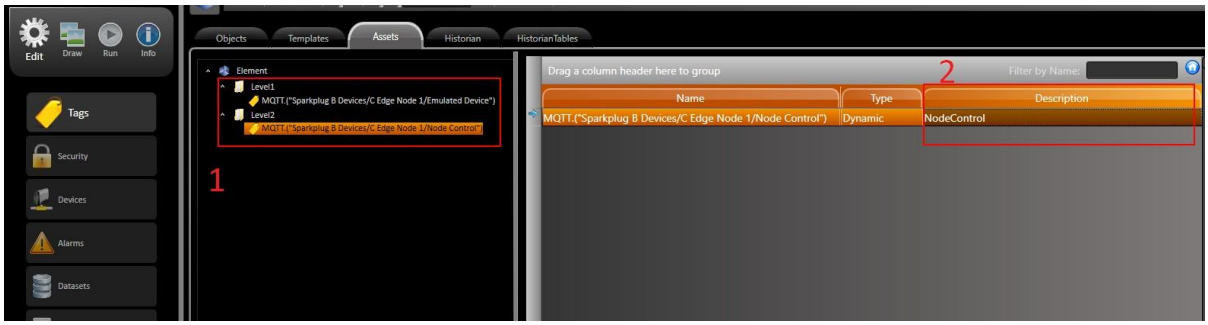
Using external variable.

4 Asset Modeling

It is possible to have a full or partial view of the MQTTspB Data Model inside your Project. In **Edit → Assets**, you can create your own Levels and assign MQTT nodes to it, from the Namespace discussed in the previous section (1).

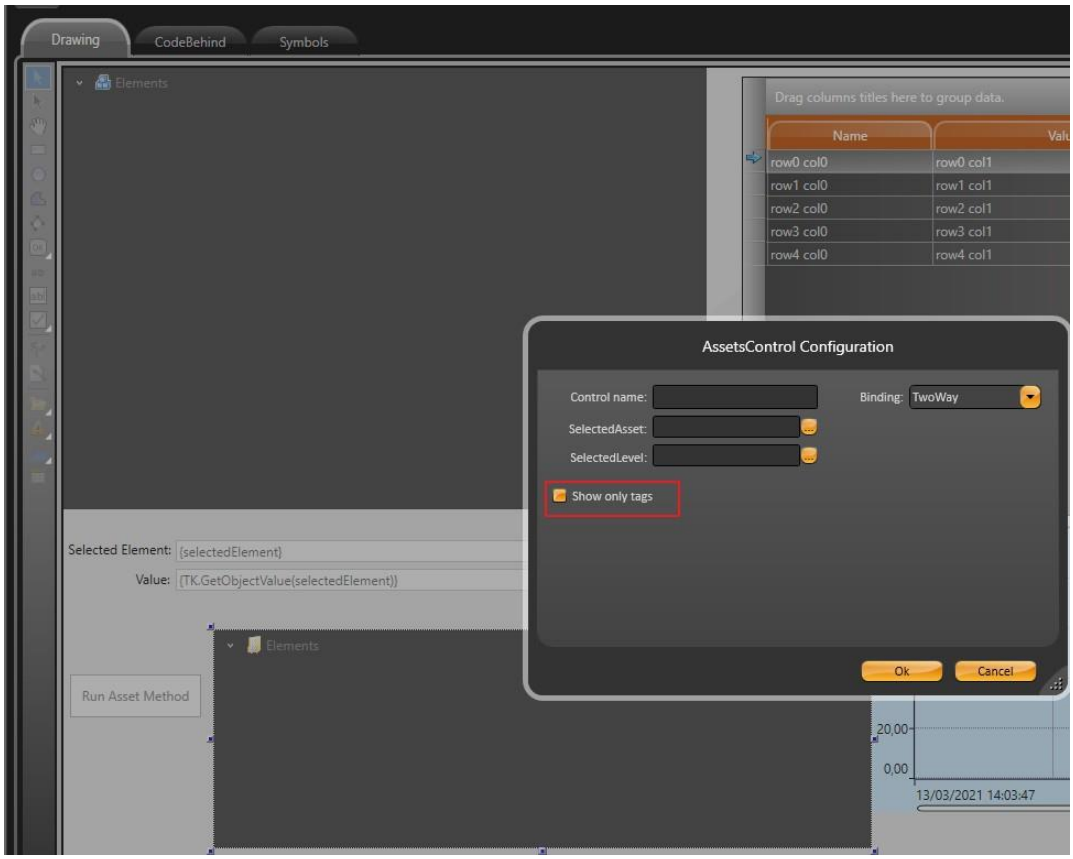
This method allows you to import just a piece of the model, from the selected node down.

The name of the Level in the Asset Tree (in Runtime) can be edited at the **Description** column (2).



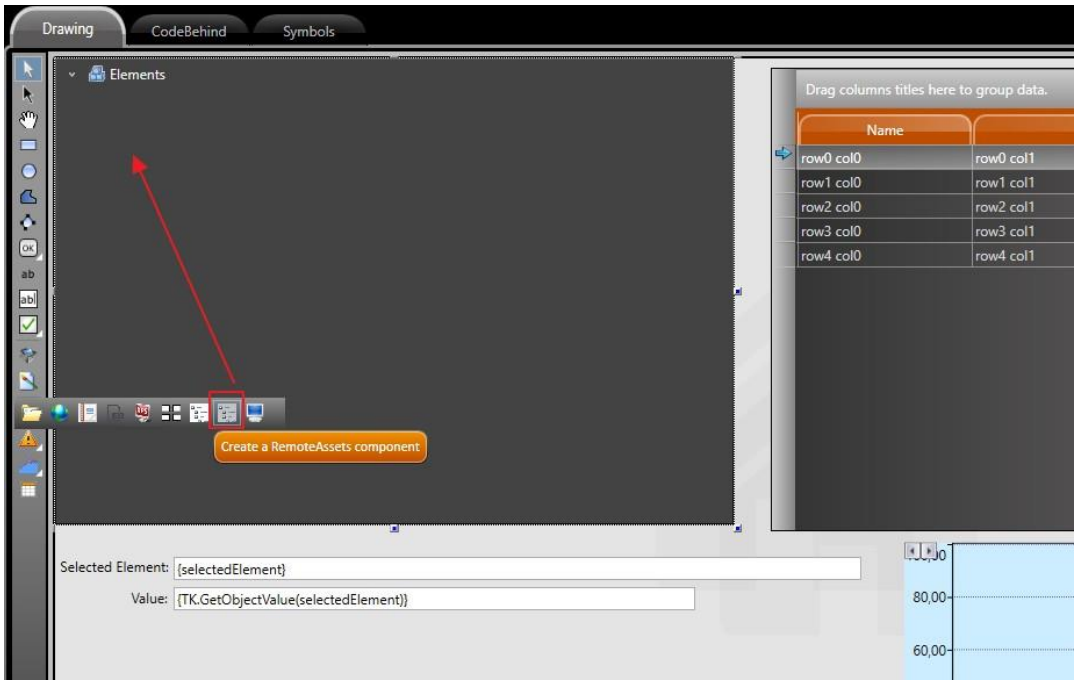
Importing Partial Asset Model.

At your **Draw** Environment, add a component called **AssetControl**. Open the element configuration and uncheck the **Show Only Tags** CheckBox.



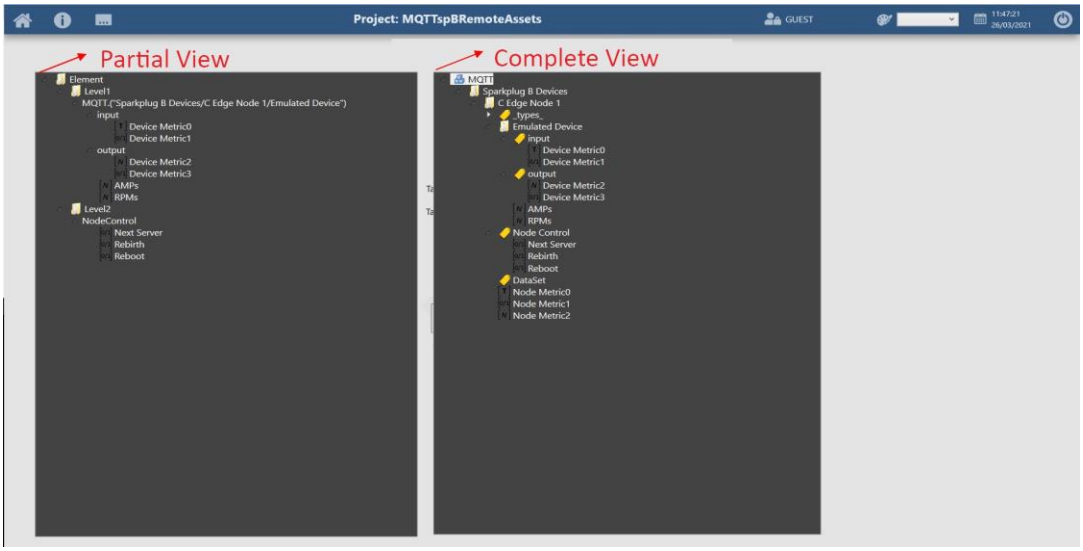
AssetControl configuration.

Alternatively, you can import the entire Data Structure by using the **RemoteAssets** component.



Component to import the entire structure.

If you have done everything correctly, in Runtime your Asset View should look something like this.



Data Structure in Runtime.

It is very important to reassure that the information displayed both in Runtime as in the Engineering Environment are Dynamic. The asset tree displayed depends on what information is being sent to the Broker.

5 Revision History

Revision A			
	Created By:	Luiz Otávio Santos	March/26/2021
	Reviewed By:	Eric Vigiani	March/26/2021
	Approved By:	Roberto Vigiani	March/26/2021