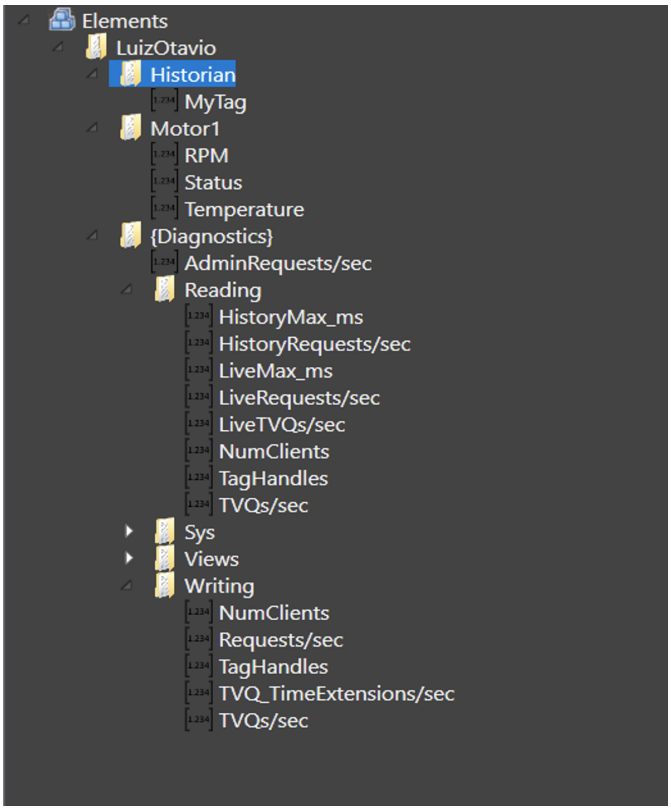
Remote Tags and Assets

Introduction

Assets let you configure additional metadata for your project when you have the Enterprise version of FactoryStudio. For example, you can organize the objects in your project into a hierarchy.

This lets you group tags that are related to each other. The hierarchy may reflect such things as the area of your manufacturing floor or the location of your machinery.

The image below describes an example of a Project hierarchy.



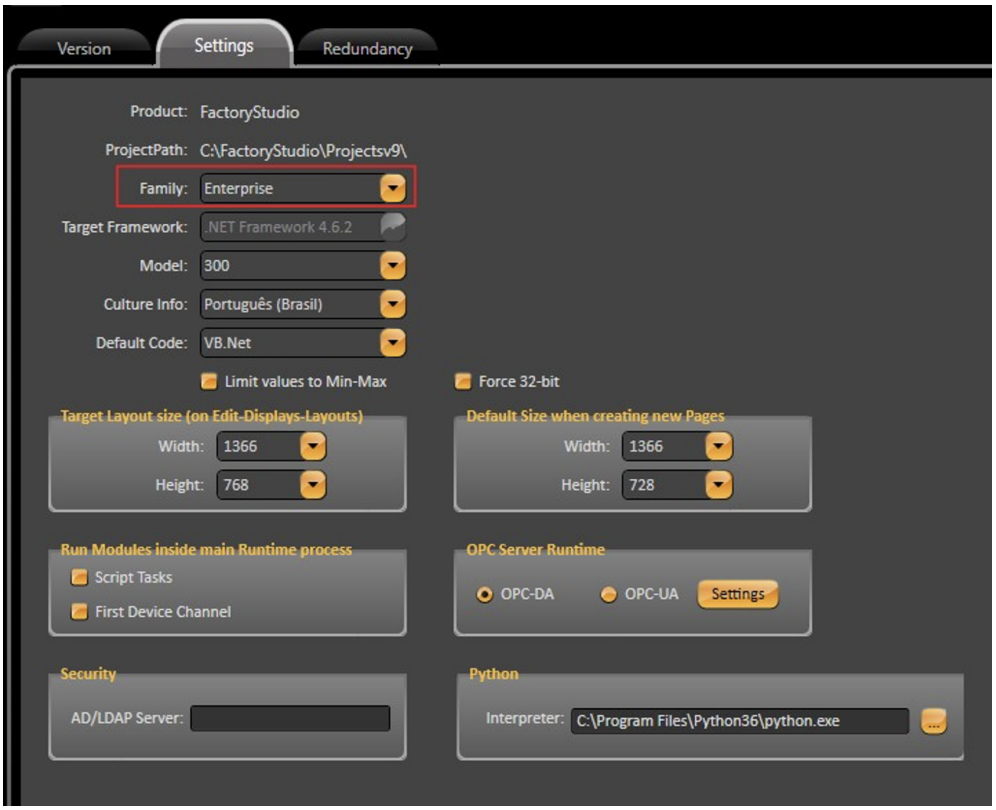
How to Us

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

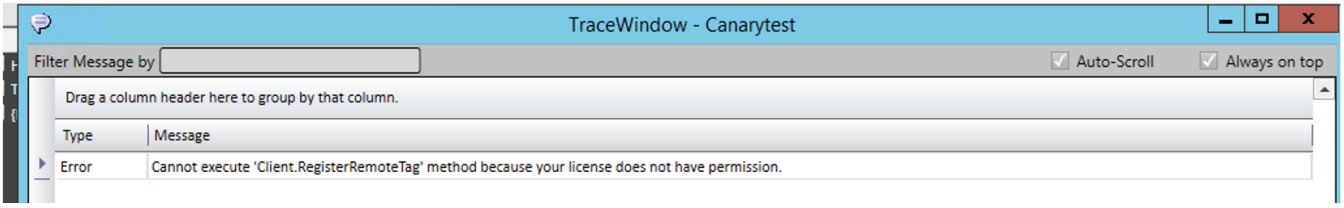
Feature Requirements

Project Settings

For the RemoteAssets and RemoteTags features to be enabled in your Project, it must be configured for ***Enterprise*** Family. To do so, navigate to *Info* > *Se ttings* and select the correct family.



Otherwise you may see an error message on your TraceWindow Logs as described in the image.



Device Configuration

At ***Devices*** > ***Channels*** Tab, create a new *Channel* for the desired communication protocol. Configure the ProtocolOptions field with the necessary information, if required by the driver.

At ***Devices*** > ***Nodes*** Tab, create a node, assign it to our newly created channel, and fill the *Station* parameters as required by the driver.

There is no need to create communication Points when using the Remote Tags and Assets feature.

Server Startup Task

At ***Scripts*** > ***Tasks*** select the default *ServerStartup* task and go to its Code Editor environment.

Add the following method call:

```
/// <summary>
/// Set device node for RemoteTags and RemoteAssets services.
/// It should be called on ServerStartup task only
/// </summary>
/// <param name="deviceNodeNameTags">Device Node Name for RemoteTags</param>
/// <param name="deviceNodeNameAssets">Device Node Name for RemoteAssets</param>
/// <returns></returns>
@Server.SetRemoteDeviceNode(string deviceNodeNameTags, string deviceNodeNameAssets)

E.g.:
string nodeRemoteTags =  @Device.Node.<NodeForRemoteTags>.GetName(); string nodeRemoteAssets = @Device.Node.
<NodeForRemoteAssets>.GetName(); @Server.SetRemoteDeviceNode(nodeRemoteTags, nodeRemoteAssets);
```

This method call will enable some pre defined methods available for Remote Tags and Assets into the chosen Node. Said methods are described in the next sections.

Runtime Methods

RegisterElementToTag

This method allows you to map a whole RemoteAsset into a Tag. In order to do that, you need to create a Template with the same structure of your asset.

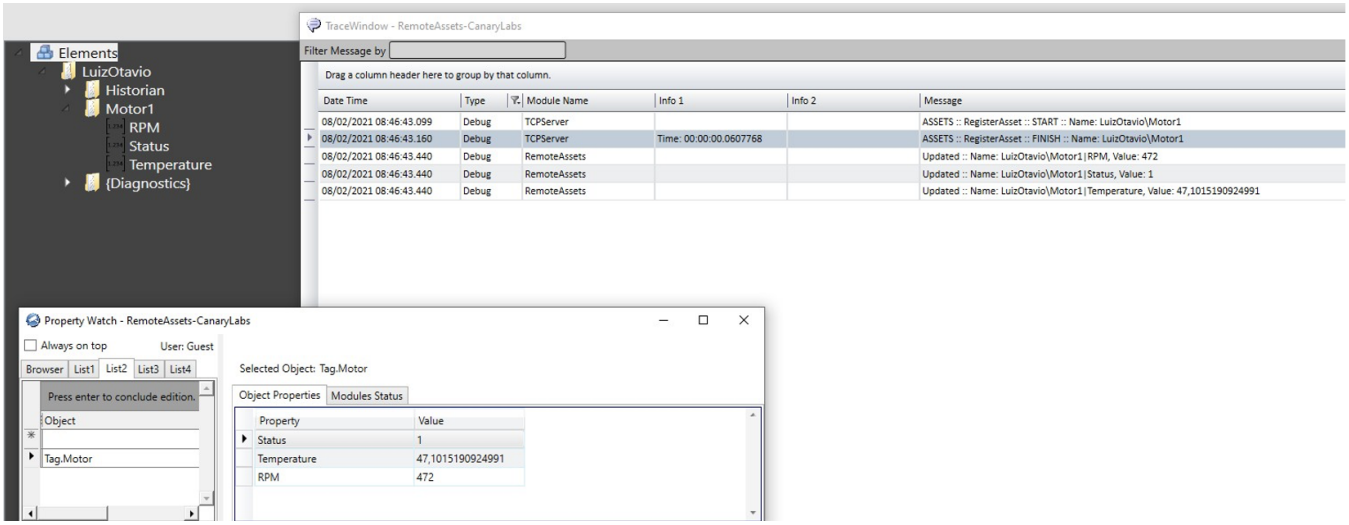
This means that you will have a Template that contains the same elements your Asset does. Method Syntax:

```
/// <summary>
/// Register element name to tag on RemoteAssets service.
/// </summary>
/// <param name="assetName">Asset Name in Remote Device</param>
/// <param name="tagName">Tag Name in your Project</param>
/// <param name="readOnly">Flag indicating whether asset is read-only [Optional]</param>
/// <returns>Flag indicating success (true) or fail (false) </returns> @Client.RegisterElementToTag(string
assetName, string tagName, [bool readOnly=false])

E.g.:
string assetName = "LuizOtavio\Motor1" string tagName = @Tag.Motor.GetName();
@Client.RegisterElementToTag(assetName, tagName);

// For Async method use @Client.RegisterElementToTagAsync(assetName, tagName);
```

When executing this method, you should see some messages in the TraceWindow Logs stating that the Remote Asset was registered to a Parent Tag and its Template elements have their values updated.



RegisterRemoteTag

This method allows you to map a specific RemoteTag into a Project Tag.

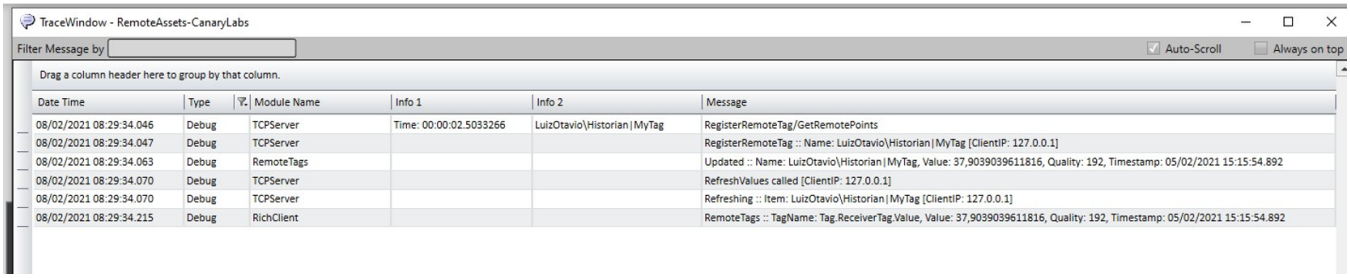
Method Syntax:

```
/// <summary>
/// Register remote tag on RemoteTags service.
/// </summary>
/// <param name="remoteTagName">Tag Name in Remote Device</param>
/// <param name="tagName">Tag Name in your Project</param>
/// <param name="readOnly">Flag indicating whether asset is read-only [Optional]</param>
/// <param name="onlyValueProperty"> Flag indicating whether only value property is used (not  Min,  Max,
EngUnits  and  Description)  [Optional]</param>
/// <returns>Flag indicating success (true) or fail (false) </returns> @Client.RegisterRemoteTag(string
remoteTagName, string tagName, [bool readOnly=false],
[bool onlyValueProperty=false])

E.g.:
string  remoteTagName  =  "LuiOtavio\Historian|MyTag"; string tagName = @Tag.DoubleTag.GetName(); @Client.
RegisterRemoteTag(remoteTagName, tagName);

// For Async method use @Client.RegisterRemoteTagAsync(remoteTagName, tagName);
```

When executing this method, you should see some messages in the TraceWindow Logs stating that the remoteTagName was registered and its value was updated.



RefreshRemoteTags

Perform a new reading on the registered elements.

Method Syntax:

```
/// <summary>
/// Refresh Tags configured in RemoteTags service.
/// </summary>
/// <returns></returns> @Client.RefreshRemoteTags()
```

UnregisterRemoteTag

Remove RemoteTag from registered list.

Method Syntax:

```
/// <summary>
/// Unregister remote tag on RemoteTags service.
/// </summary>
/// <param name="remoteTagName">Tag Name in Remote Device</param> @Client.UnregisterRemoteTag(string
remoteTagName)

E.g.:
string  remoteTagName  =  "LuiOtavio\Historian|MyTag"; string tagName = @Tag.DoubleTag.GetName(); @Client.
UnregisterRemoteTag(remoteTagName);

// For Async method use @Client.UnregisterRemoteTagAsync(remoteTagName, tagName);
```

UnregisterElementToTag

Remove RemoteAsset from registered list.

Method Syntax:

```
/// <summary>
/// Unregister element name to tag on RemoteAssets service.
/// </summary>
/// <param name="assetName">Asset Name in Remote Device</param>
/// <returns>0 if Success</returns> @Client.RegisterElementToTag(string assetName)

E.g.:
string assetName = "LuizOtavio\Motor1" @Client.UnregisterElementToTag(assetName);

// For Async method use @Client.UnregisterElementToTagAsync(assetName);
```

UnregisterAllAssets

Remove All RemoteAssets from registered list.

Method Syntax:

```
/// <summary>
/// Unregister all Assets from RemoteAssets service.
/// </summary>
/// <returns>0 if Success</returns> @Client.UnregisterAllAssets()

E.g.:
@Client.UnregisterAllAssets();
```

UnregisterAllRemoteTags

Remove All RemoteTags from registered list.

Method Syntax:

```
/// <summary>
/// Unregister all Remote Tags from RemoteAssets service.
/// </summary>
/// <returns>0 if Success</returns> @Client.UnregisterAllRemoteTags()

E.g.:
@Client.UnregisterAllRemoteTags();
```