# Remote Tags API

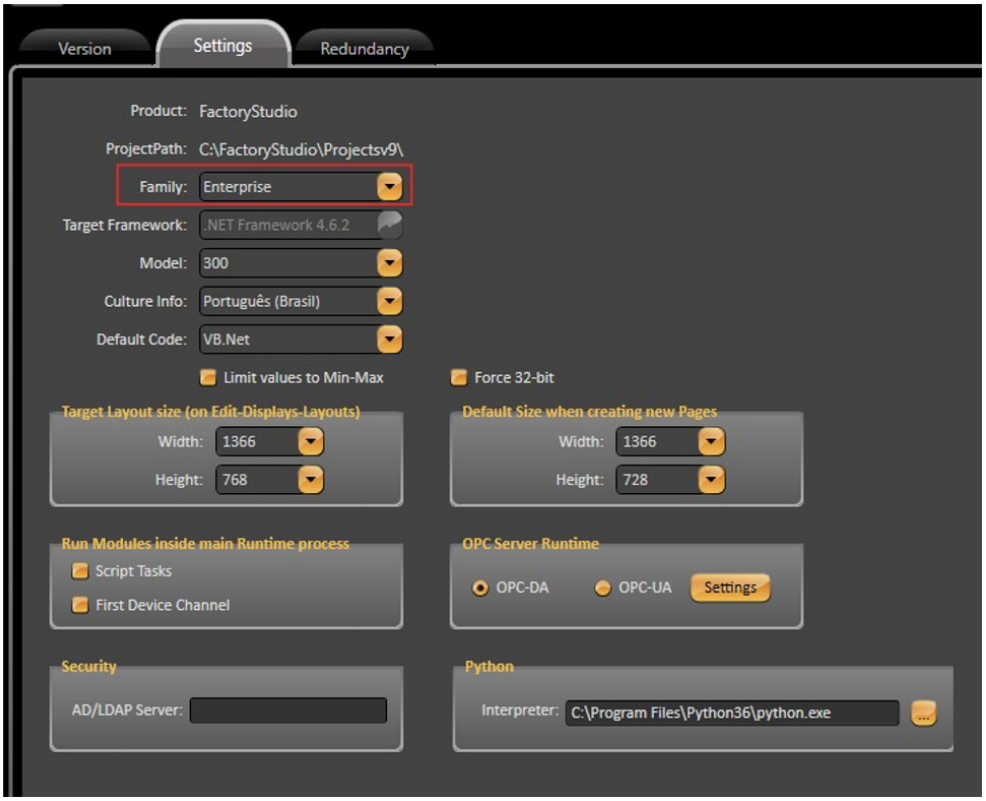
## How to Use

In order to use the Remote Tag API, you need to make sure some requirements are matched.

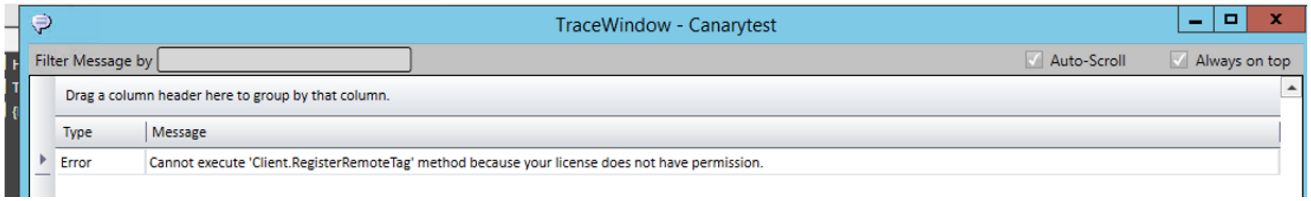
### Feature Requirements

#### Project Settings

To enable the RemoteAssets and RemoteTags features in your Project, you must be configured for the ***Enterprise*** Family. To do so, navigate to **Info > Settings**, and select the correct family.




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




#### Device Configuration

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

In the ***Devices*** > ***Nodes*** Tab, create a node, assign it to our newly created channel, and fill in 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

On ***Scripts*** > ***Tasks,*** select the default *ServerStartup* task, and go to its Code Editor environment. Add the following call method:

```
/// <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 call method will enable some predefined methods available for Remote Tags and Assets in the chosen Node. These methods are described in the next sections.

---

## Runtime Methods

### RegisterElementToTag

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

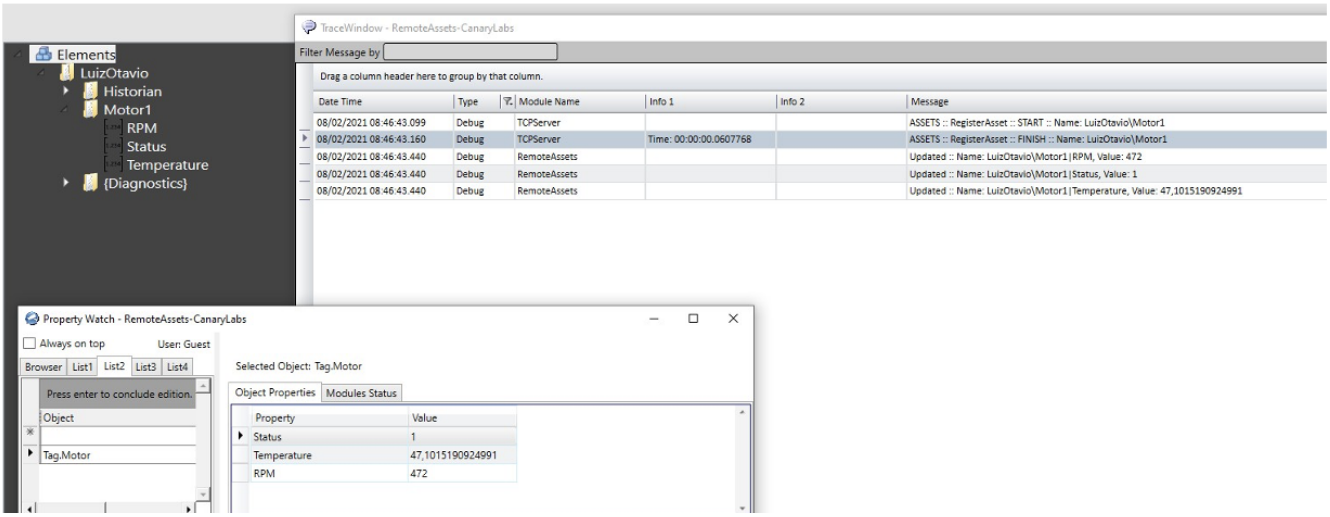
This means that you will have a Template that contains the same elements your Asset does. The method's syntax is:

```
/// <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 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

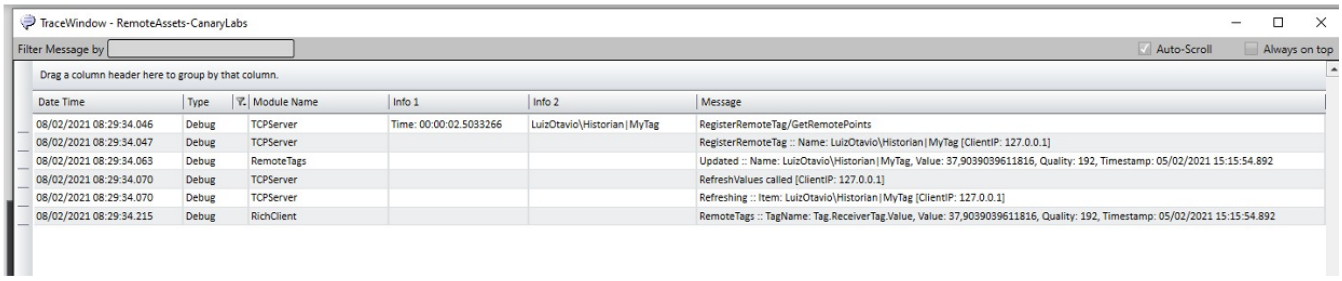
This method allows you to map a specific RemoteTag to a Project Tag.

```
/// <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 messages in the TraceWindow Logs stating that the remoteTagName was registered and its value was updated.




## RefreshRemoteTags

Performs a new reading on the registered elements.

The method's syntax is:

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

## UnregisterRemoteTag

Removes RemoteTags from the registered list.

The method's syntax is:

```
/// <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

Removes RemoteAssets from the registered list.

The method's syntax is:

```
/// <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

Removes All RemoteAssets from the registered list.

The method's syntax is:

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

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

## UnregisterAllRemoteTags

Removes All RemoteTags from the registered list.

The method's syntax is:

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

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