



EcoStruxure™

Geo SCADA Expert

Sparkplug B

Simple Driver Development Kit Sample Driver

www.schneider-electric.com

Issue Details

Issue	Date	Author	Comments
1	30.06.20	S. Beadle	New.

References

Sparkplug B protocol:

<https://www.eclipse.org/tahu/spec/Sparkplug%20Topic%20Namespace%20and%20State%20ManagementV2.2-with%20appendix%20B%20format%20-%20Eclipse.pdf>

Simple Driver Development Kit reference:

<https://tprojects.schneider-electric.com/telemetry/display/CS/Technical+Guides>

Sparkplug B payload libraries and samples:

<https://github.com/eclipse/tahu>

MIT License

Copyright (c) 2020 Schneider Electric and its subsidiaries. All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Table of Contents

Chap	Title	Page
1	Introduction	1
1.1	This Driver's Source Code	1
1.2	The Sparkplug B Protocol	2
1.2.1	The Broker	2
1.2.2	Message Payload	2
1.2.3	MQTT Client Library	2
1.2.4	Topics	2
2	Building the Driver	4
2.1	Sparkplug B	4
2.1.1	References.....	4
2.1.2	Files.....	4
2.1.3	Debugging.....	5
2.2	DriverSparkplug B.....	5
2.2.1	References.....	5
2.2.2	Post-Build Operations	6
2.3	DriverSparkplugBInstaller	6
3	Using the Driver.....	7
3.1	Installation	7
3.1.1	Automated Installation	7
3.1.2	Manual Installation	7
3.2	Configure Objects	8
3.3	Configure a Broker	8
3.4	Test Nodes and Devices.....	9
3.4.1	Python	9
3.4.2	Node-Red	10
3.5	Your First BIRTH	12
3.6	Created Configuration.....	14
3.6.1	Node	14
3.6.2	Device	15
3.6.3	Points	15
3.7	Receiving New Configuration	16
3.8	Making Templates of the Configuration	16
3.9	Automated Configuration of Instances.....	18
3.10	Property Configuration	18
3.11	Processing Data	19
3.12	How to Diagnose any Problems	19
4	The Code	21
4.1	Where Next?	21

1 Introduction

This document describes the code for a Sparkplug B driver for Geo SCADA Expert, written in C# for the Simple Driver Framework using the Driver Development Kit (DDK).

The driver is offered as source code which you can build with Visual Studio. It includes the two parts of the driver and an installer, enabling you to build a package to deploy to Geo SCADA Expert servers. It is not supported.

The source code is available for you to freely use, modify and extend to suit your requirements or that of your clients. It is not the most optimized, efficient or elegant code, and the functionality is not assured in the way the core product is, but we hope that its simplicity will encourage engagement with the Geo SCADA driver development process and explore the new ideas presented for Sparkplug B implementation with Geo SCADA. We encourage you to add to the code by submitting 'pull requests' on GitHub.

The functionality in the driver includes basic data processing, and the feature which we hope will stimulate innovation and conversation is the built-in automatic configuration facility. This integrates well with the Geo SCADA template and instance features.

To implement and deploy this example you will need to verify functionality and add appropriate security measures for your environment.

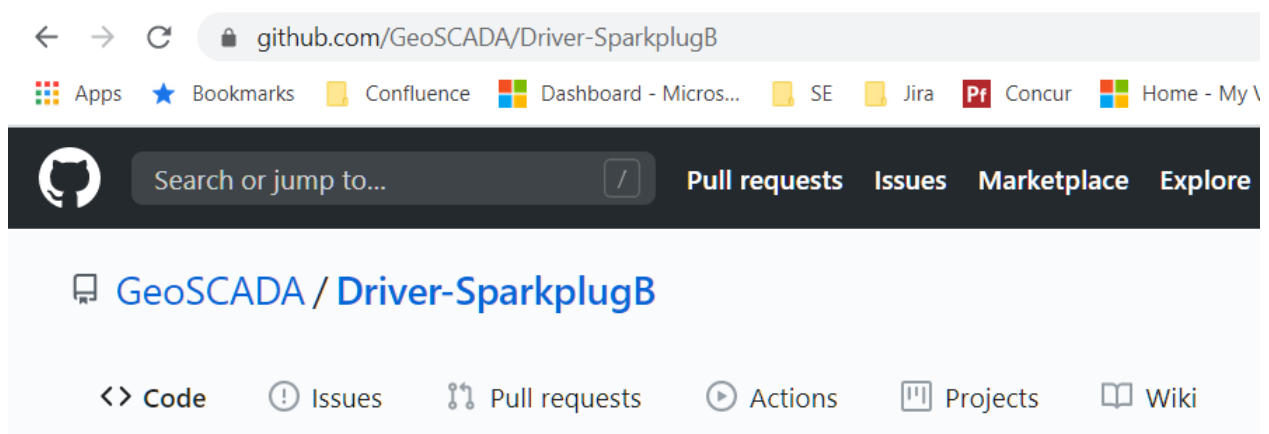
We have developed this technology preview independently of the MQTT protocol framework built in to the Geo SCADA Expert product. Feedback about this from customers will help us decide how to add a Sparkplug B driver to the core product.

You can discuss these features, driver development, MQTT and Sparkplug B in the SE Exchange forums:

<https://community.exchange.se.com/t5/Geo-SCADA-Expert-Forum/bd-p/ecostruxure-geo-scada-expert-forum>

1.1 This Driver's Source Code

You will find source code for this driver within the GitHub system. The project name is Geo SCADA / Driver-Sparkplug B:



URL: <https://github.com/GeoSCADA/Driver-SparkplugB>

1.2 The Sparkplug B Protocol

The MQTT protocol specifies how computers connect and exchange messages with a communications hub, named a Broker. MQTT does not specify the payload (content) of those messages. Support of MQTT does not give any interoperability between devices and systems.

Sparkplug B is a specification for MQTT enabled devices and applications to send and receive messages with a standard payload format oriented to SCADA applications. It is an Eclipse open source project. Sparkplug B specification specifies how various MQTT devices must connect and disconnect from the MQTT broker. This includes device lifecycle messages such as the required birth and last will & testament messages that must be sent to ensure the device lifecycle state and data integrity. The specification describes how point data ('metrics') are defined and communicated.

Sparkplug B adds timestamping of data to the previous A version of Sparkplug.

For protocol detail, this document will just point users to the protocol payload documentation in the References section. However, here are some points of note regarding how the protocol maps into Geo SCADA Expert features.

1.2.1 The Broker

The Sparkplug B protocol uses MQTT, which uses a communications 'Broker' or Server acting as a go-between for devices and master. The broker is not part of Geo SCADA Expert. Brokers are typically open-source and may be cloud hosted, though some cloud hosted services may not provide all MQTT features such as retained messages.

To test the driver you will need a broker on your network. Geo SCADA does not include a broker. Available brokers include:

Eclipse MosquittoTM - <https://mosquitto.org/>

RabbitMQ - <https://www.rabbitmq.com/>

Mosca - <https://github.com/mcollina/mosca>

1.2.2 Message Payload

The Sparkplug B protocol generally defines message Topics, Payload and sequencing of messages. The payloads are – to MQTT – transparent blocks of binary data, and Sparkplug B uses a message encoding format called Protobuf, which is an open source format initially created by Google.

Protobuf messages are defined by a schema, and the schema is compiled into the code of the driver.

1.2.3 MQTT Client Library

The Geo SCADA implementation uses a .Net library called M2MQTT (<https://github.com/eclipse/paho.mqtt.m2mqtt>) to provide the MQTT client protocol functionality. If protocol feature changes are required, then either the M2MQTT library must be extended, or another library used to provide the features needed. For example, it would be possible for this to be replaced by other libraries such as <https://github.com/chkr1011/MQTTnet> or <https://www.eclipse.org/paho/clients/dotnet/>.

1.2.4 Topics

Communications are routed via a 'Topic' which typically identifies the source or recipient of data. A master will usually subscribe to a wildcard topic to receive data from multiple devices matching some characteristics. This is part of the Sparkplug B namespace.

The namespace is a sequence of elements separated by '/'. It is:

namespace/group_id/message_type/edge_node_id/[device_id]

Namespace

The first part (also called namespace) is a fixed string for Sparkplug B. Our driver just supports the single namespace, but you could expand this to support Sparkplug A or future versions if required.

Group id

Sparkplug B then uses a 'Group' name to separate collections of devices. Our driver uses a Broker object, corresponding to a Geo SCADA channel, which both defines broker properties and also this Group name, so that multiple Broker objects can be created with different Group names. Multiple broker objects can connect to the same broker.

There is currently no code to prevent Broker objects referring to the same group.

Message type

All of the Sparkplug B message types are supported.

Edge node id

The edge node is an object represented by the driver's 'Node device' driver object. It corresponds to a Geo SCADA Scanner, and owns points linked to it. In Sparkplug B, the Edge node is connected to the broker.

Note that the '/' between edge and device is omitted if no device is present, so the namespace should really be written as:

namespace/group_id/message_type/edge_node_id[/device_id]

Device id

The Device is a grouping of I/O – notionally a set of points connected via the Edge node. Consider the Edge node as the RTU, and a Device as something connected to the RTU such as a meter using Modbus. The Edge node presents the Device as a collection of points, and the protocol reports these points in a separate protocol message from the Edge node.

In our driver we have used the same Geo SCADA object type as the Edge node as for the Device. (Sparkplug B uses the term 'metric' for a point). This is because most of the functionality of these two things is the same. The minor variations are principally with the Birth/Death messages and sequence numbers.

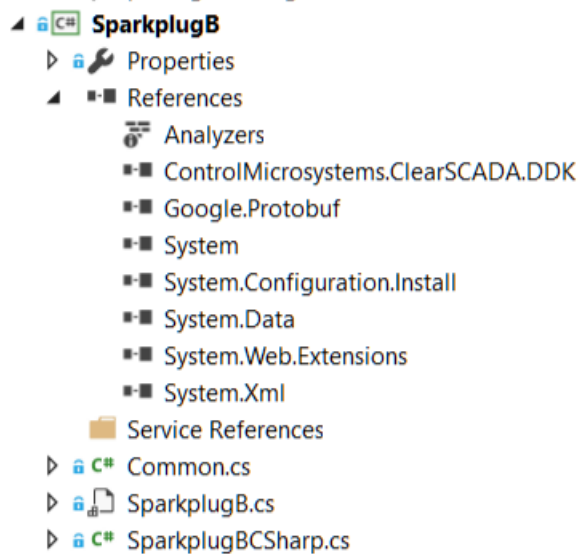
2 Building the Driver

This section lists the projects within the solution and notes on how to build them.

2.1 Sparkplug B

This is the .dll module which defines in-database objects, properties and behaviors. The module is loaded at startup by the Geo SCADA DBServer process.

2.1.1 References



DDK

The module refers to the Geo SCADA dll 'ControlMicrosystems.ClearSCADA.DDK.dll'. You should find this reference in the project, remove it and then reinstate it from the location used on your build computer.

If you are developing on a computer which has Geo SCADA installed, find this in c:\Program Files\Schneider Electric\ClearSCADA

Alternatively, you can copy this .dll to your own location from a Geo SCADA installation. In this case you may set up multiple builds for different Geo SCADA versions if you wish. Note that you need the major version number of this .dll (e.g. 81) to match the major version of the Geo SCADA target.

Protobuf

The Google Protobuf reference is included using the NuGet package manager. Version 3.12.2.0 is used. A package reference will add this for you, but the package manager command is here for reference: `Install-Package Google.Protobuf -Version 3.12.0`

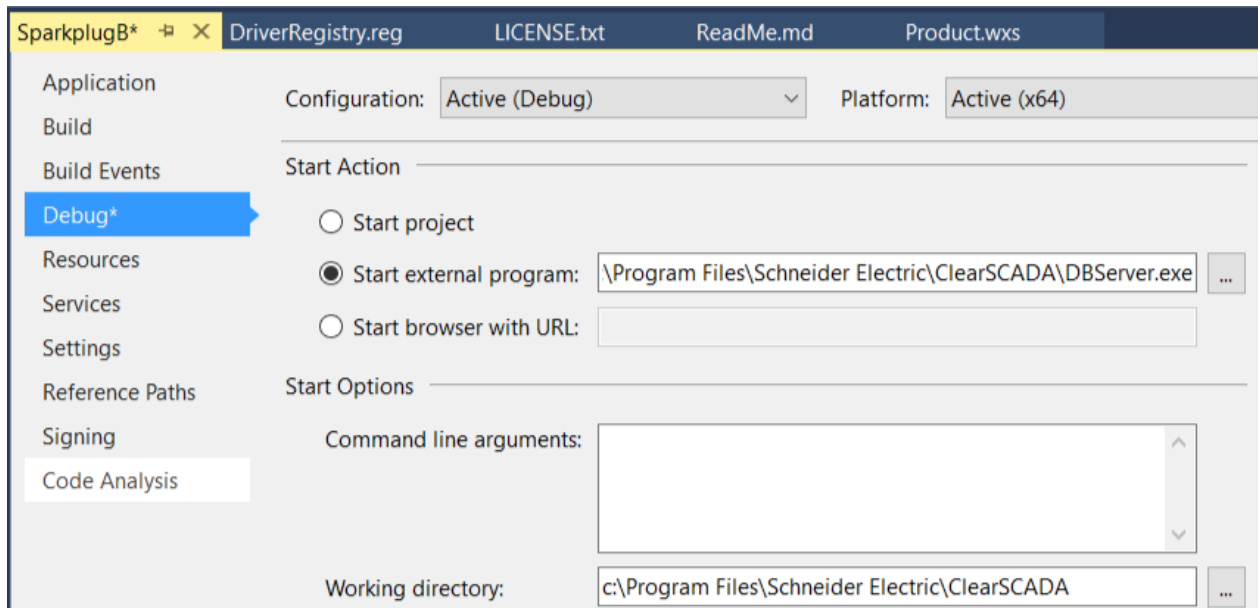
2.1.2 Files

In the file Common.cs the base number for OPC IDs is defined as 0x0468D000. Keep this unchanged, and if you add new fields please remain within the range of 0x0468D000 to 0x0468DFFF.

The file SparkplugB.cs contains the object definitions and will be of most interest. The file SparkplugBCSharp.cs is generated by the protocol buffer compiler and should not be changed.

2.1.3 Debugging

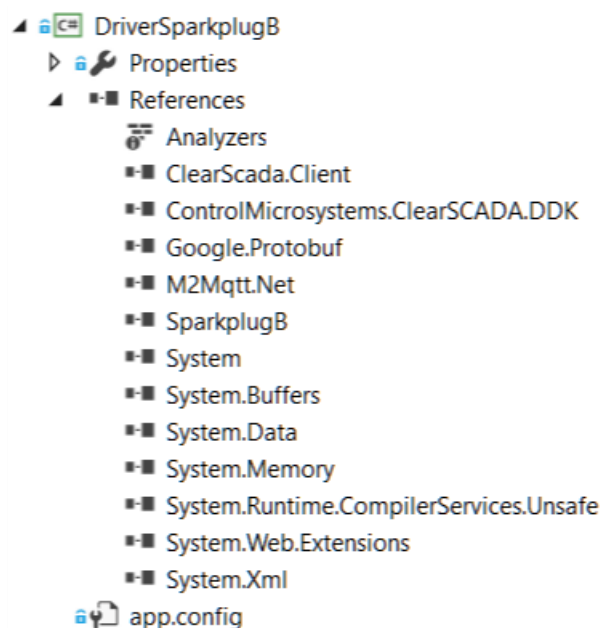
If you wish to debug this module, set the external program and working directory as shown here:



2.2 DriverSparkplug B

This is the .exe file which runs independently of the database and contains the functionality necessary to interact with the broker and to interpret Sparkplug B messages.

2.2.1 References



DDK and Client

The module refers to the Geo SCADA dll 'ControlMicrosystems.ClearSCADA.DDK.dll', AND the dll 'ClearSCADA.Client'. You should find these references in the project, remove them and then reinstate them from the location used on your build computer.

If you are developing on a computer which has Geo SCADA installed, find them in c:\Program Files\Schneider Electric\ClearSCADA

Alternatively, you can copy these .dll to your own location from a Geo SCADA installation. In this case you may set up multiple builds for different Geo SCADA versions if you wish. Note that you need the major version number of these .dll (e.g. 81) to match the major version of the Geo SCADA target.

It is not usual for a driver to use the Client dll. In this case it allows database configuration as described later.

Sparkplug B

This module requires a reference to the dll built above. You will therefore first build the SparkplugB project, then remove and re-add the reference to the .dll just built (default <path>\Driver-SparkplugB\SparkplugB\bin\x64\Debug\SparkplugB.dll).

Protobuf

The Google Protobuf reference is included using the NuGet package manager. Version 3.12.2.0 is used. A package reference will add this for you, but the package manager command is here for reference: Install-Package Google.Protobuf -Version 3.12.0

M2Mqtt.Net

The M2Mqtt.Net reference is included using the NuGet package manager. Version 4.3.0 is used. A package reference will add this for you, but the package manager command is here for reference: Install-Package M2Mqtt -Version 4.3.0

2.2.2 Post-Build Operations

The driver build properties includes a copy operation at the end of build. It will copy the .exe and .dll files to the target folder. You will need to change these to: c:\Program Files\Schneider Electric\ClearSCADA

i.e.

```
copy $(TargetDir)\SparkplugB.dll c:\Program Files\Schneider Electric\ClearSCADA\
copy $(TargetDir)\SparkplugB.PDB c:\Program Files\Schneider Electric\ClearSCADA\
copy $(TargetDir)\M2Mqtt.Net.dll c:\Program Files\Schneider Electric\ClearSCADA\
copy $(TargetDir)\Google.Protobuf.dll c:\Program Files\Schneider Electric\ClearSCADA\
copy $(TargetDir)\DriverSparkplugB.PDB c:\Program Files\Schneider Electric\ClearSCADA\
```

2.3 DriverSparkplugBInstaller

This project is a basic WIX installer for the driver. It produces a .msi file for execution on a target computer. There is no upgrade capability – just remove and reinstall. Version numbers, as for the projects, are fixed and could be changed by you.

The installer places the driver .exe and .dll into the correct location and inserts the registry entries required for the driver to run.

The installer, like the driver, is unsigned.

3 Using the Driver

This section describes how to get started with the driver.

3.1 Installation

3.1.1 Automated Installation

Automated installation can be achieved with the installer within the development project. The installer project will copy the required files into the right place and register the DLLs.

3.1.2 Manual Installation

If you wish to install manually, this driver is installed in the same way as other DDK drivers.

Manual installation just consists of copying these files into the Geo SCADA executable folder:

"C:\Program Files\Schneider Electric\ClearSCADA"

These files are:

"DriverSparkPlugB.exe"

"SparkPlugB.dll"

"Google.Protobuf.dll"

"M2Mqtt.Net.dll"

There are some registry settings needed to allow the driver to work. You can import these manually or paste this into a .reg file and open it.

Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\Schneider
Electric\ClearSCADA\DriverSparkplugB]

@="SparkplugB"

"AssemblyName"="C:\\Program Files\\Schneider
Electric\\ClearSCADA\\SparkplugB.dll"

"DebugMode"="False"

"TaskName"="DriverSparkplugB.exe"

"LogEnable"="True"

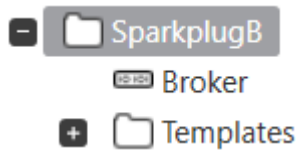
Alternatively, you can call the .Net installation command as follows:

1. Open a command prompt with administrative permissions and set the current directory to the ClearSCADA directory (usually c:\Program Files\Schneider Electric\ClearSCADA).
2. Run the Microsoft .NET Framework InstallUtil.exe on the DLL:

```
%Windir%\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe SparkplugB.dll
```

3.2 Configure Objects

To get started it is recommended to create a structure like this:



Descriptions:

- Sparkplug B – create a group for configuration objects
 - Broker – create the SparkPlug B driver broker
 - Templates – create a group for templates (start with this empty).

3.3 Configure a Broker

A Broker object defines the connection to the broker, the Group Id of the devices communicating and other characteristics of the group such as the automatic configuration features.

Example:

The top section of this object is concerned with the MQTT broker connection, such as the host details, user name and password, client Id, version and certificate security.

The fields relevant to Sparkplug B protocol are:

- Group Id – an identifier for the devices and nodes belonging to this namespace. This should be unique amongst the broker objects, but it is not checked and there is no real reason why data cannot be duplicated.
- SCADA Host Id – a name used by the Sparkplug protocol to identify this master's state, using this name as the Last Will and Testament (LWT) topic which an edge node can subscribe to. For redundant servers, this name must be unique per server, however this is not yet implemented. Implementation could be to simply append the server host name to this Id.
- Automatic Configure – as will be described later, self-configuration is a key feature of this driver. The feature enabled by this checkbox will cause self-configuration to happen when any device connects with valid namespace and

payload data in its birth certificate. When unchecked, new nodes and devices will be placed in a queue, pending an action on the broker to cause configuration.

- Server Port, User Name, Password – these fields are a valid Geo SCADA user credentials which will execute the configuration actions. When not configured, the driver will be unable to create or modify devices.
- Create Devices or Instances in Group – Enter a group name. e.g. "SparkplugB". New Nodes will be created within this group.
- Node Template – Enter a template name. This template will contain a Node object and zero or more points, plus any displays, logic, trends or other objects as required. When a new Node is configured by the driver, this template will be instanced. (Leave blank for now. More about this later).
- Device Templates – Enter a group name. e.g. "SparkplugB.Templates". This group will contain multiple templates. Each template will be named according to the Device name they correspond to. When a new Device is seen and then configured, the driver will attempt to find the template matching the Device name, otherwise it will create a group and device as non-templated objects. (More about this later).

When a broker is configured correctly and is subscribed to a broker, the broker state will be Healthy.

You are now ready to connect Sparkplug B nodes to your broker! The next section describes what happens when you do.

3.4 Test Nodes and Devices

Using Test Software

If you do not yet have test devices ready to connect, then it is possible to use test software to simulate the behavior of a Sparkplug data source. Examples here are for Python and Node-Red, and the follow-on images relate specifically to the Node-Red example.

3.4.1 Python

You can run Python on any Windows or Linux version. Find a sample Python (2.7) implementation for the Raspberry Pi here:

https://github.com/eclipse/tahu/tree/master/sparkplug_b/raspberry_pi_examples/python

The example is designed to work on a Raspberry Pi using the PiBrella GPIO 'HAT' extension, but it will work without that without modification.

Find a generic example here:

https://github.com/eclipse/tahu/blob/master/sparkplug_b/stand_alone_examples/python/example.py

You will need to edit the code and as a minimum, add the Broker/Server connection details.

```
serverUrl = "192.168.1.53"
myGroupId = "Sparkplug B Devices"
myNodeName = "Python Raspberry Pi"
mySubNodeName = "Pibrella"
```

```
myUsername = "admin"
myPassword = "changeme"
```

The sample requires a Python Library for MQTT and for Sparkplug. The former can be added with Python PIP. The Sparkplug library is available here:

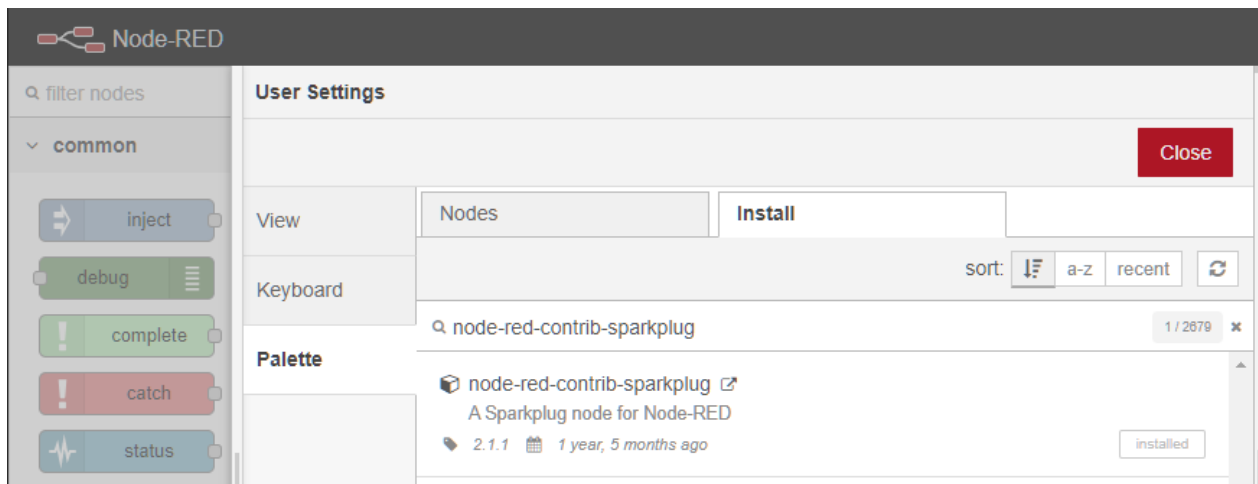
https://github.com/eclipse/tahu/tree/master/client_libraries/python

Copy these files into the same folder as example.py and run the example after making edits. The program will connect as a Node and Field Device, send data and respond to commands (controls).

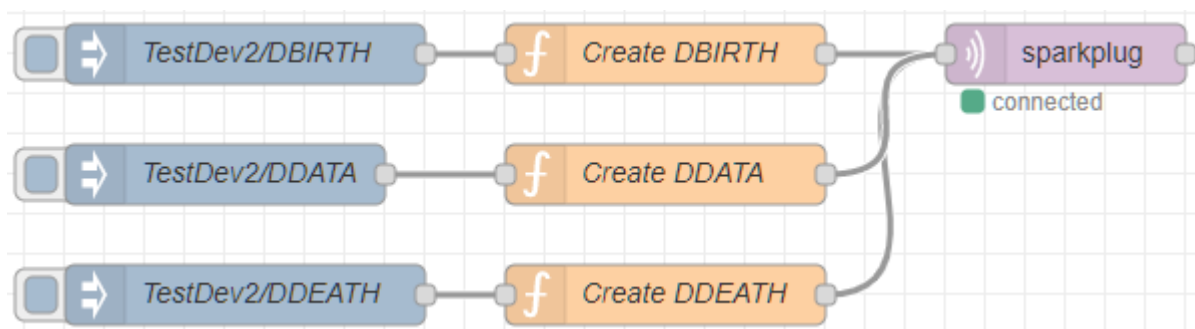
3.4.2 Node-Red

An alternative to Python is to use Node-Red. This is a web-based graphical environment for designing and testing data flows. See <https://nodered.org/> for download and installation guides.

Node-red requires a 'palette' of items which provide MQTT and Sparkplug capability. The web interface of Node-Red has a menu at the top left – select 'Manage Palette'. Then select the 'Install' tab and enter: node-red-contrib-sparkplug



Install this item, then use the designer to create the following:



Each of the three grey Inject nodes specify the Topic name, using the Device Id 'TestDev2'. The three orange function nodes specify the message payload in JSON format which the next sparkplug node converts into Protobuf for sending.

The DBIRTH message and DDATA messages follow similar formats, with the DDATA message using the metric (point/tag) Alias feature to specify the metric numerically to reduce bandwidth for data updates.

The example below creates 50 digital and 50 analogue points and feeds them with random numbers. Paste the script in the two boxes below into the DBIRTH and DDATA function nodes.

```
// DBIRTH Message
var timenow = new Date();
msg.payload = {"timestamp": timenow*1,
  "metrics": [
  ]
};
var i;
for (i = 1; i <= 5; i++) {
  msg.payload.metrics.push(
    {
      "name": "Binary " +
('0'+i).slice(-2),
      "alias": i,
      "value": false,
      "type": "boolean"
    }
  );
}
for (i = 6; i <= 10; i++) {
  msg.payload.metrics.push(
    {
      "name": "Analog " +
('0'+i).slice(-2),
      "alias": i,
      "value": 0,
      "type": "float"
    }
  );
}
return msg;
```

```
// DDATA Message
var timenow = new Date();
msg.payload = {"timestamp": timenow*1,
  "metrics": [
  ]
};
var i;
for (i = 1; i <= 5; i++) {
  msg.payload.metrics.push(
    {
      "timestamp": timenow*1,
      "alias": i,
      "value": Math.random() < 0.5 ?
false : true,
      "type": "boolean"
    }
  );
}
for (i = 6; i <= 10; i++) {
  msg.payload.metrics.push(
    {
      "timestamp": timenow*1,
      "alias": i,
      "value": Math.random() * 100.0,
      "type": "float"
    }
  );
}
return msg;
```

The DDEATH message just needs a simple expression:

```
var timenow = new Date(); msg.payload = {"timestamp": timenow*1}; return msg;
```

The final sparkplug node requires the configuration of fields related to the MQTT Broker and the node. Here is an example setup – just use your own server/port addresses for the Broker and set up a Group Id and Edge Node name.

Edit sparkplug node

Delete

Cancel

Done

⚙️ Properties

⚙️ 📄 🖨️

🌐 Server

tcp://192.168.86.123

🌐 Port

1883

👤 Username

fred

🔒 Password

.....

🔑 Client ID

NodeREDSimpleEdgeNode

🔑 Group ID

SparkplugDevices

🔑 Edge Node

Node-REDEdgeNode

🔑 Version

Sparkplug B ▾

To use the simulation, click the BIRTH inject node, then click the DATA inject nodes each time to send in all metric values.

3.5 Your First BIRTH

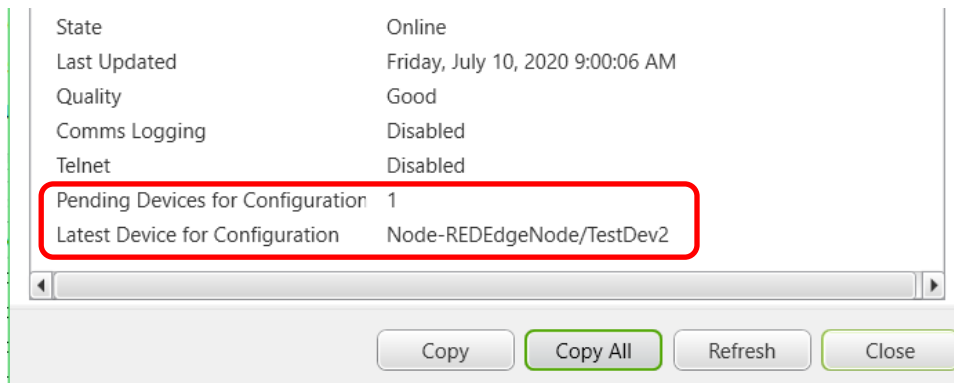
On connecting a device, the Node Birth and optionally Device Birth messages are sent via broker to the driver. What happens next will depend on how the 'Automatic Configure' checkbox is set on your Broker object.

If this field of the Broker is **unchecked**: Automatic Configure ☐

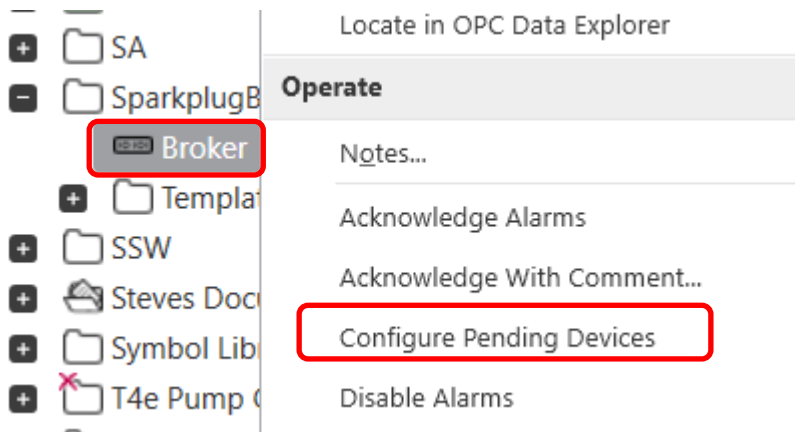
Nothing will appear to happen, but the Broker object will cache what it has received. You can see this in the Status panel of the Broker:

Broker Status Information ✖	
Attribute	Value
Id	400330
Full Name	SparkplugB.Broker
Type	SparkplugB Broker
Last Modified	7/10/2020 9:00:06.254 AM by s (Version 14)
Events	Current Hour Event Count 19, Previous Hour Event Count 9

~~~~~



You can now configure the Node by selecting the context menu item 'Configure Pending Devices' on the Broker:



Alternatively, if this field of the Broker is **checked**:

Automatic Configure ☒

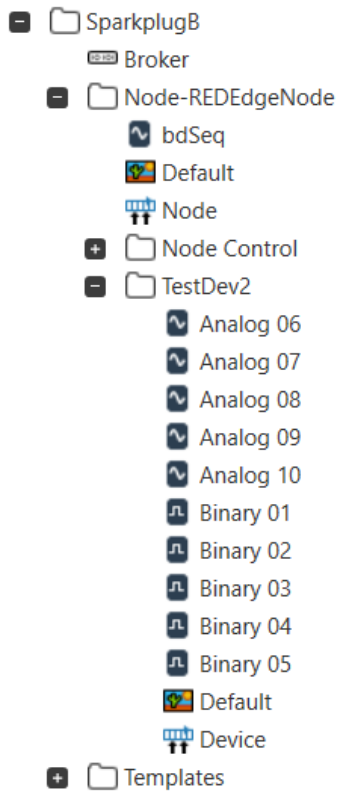
Then the configuration will be fully automatic, and the Edge Node, Devices and Points will be created with no user intervention.

The 'Configure Pending Devices' method on the Broker should request all 'queued' configurations be action and then remove each of them from that queue. The code could be extended to allow individual node/device to be discovered and configured.



## 3.6 Created Configuration

The driver will configure all database objects needed and set various properties as required for them to be valid and useful.

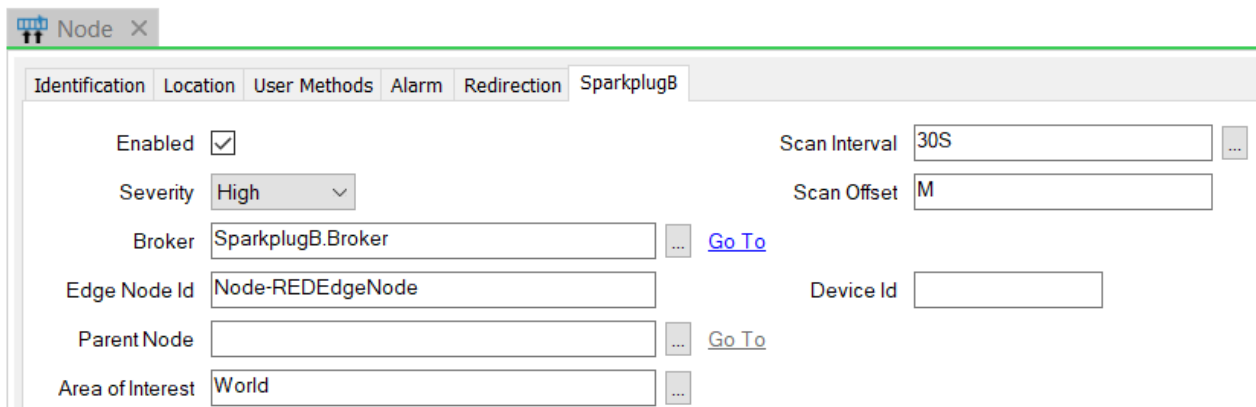


A group has been created for the Edge Node, and a Node/Device object created. A group has also been created for the Device, and a Node/Device object created too. Also, all the point objects have been created too. Every item has an appropriate name and has database fields set up correctly.

The Node and Device objects are actually the same database object, with different attributes.

### 3.6.1 Node

The Node is configured as follows:



There is a link to the Broker object. The scan interval and offset are irrelevant here and could be hidden. The specific fields here are:

- Edge Node Id – the name of this Node within the namespace. Note that this will usually match the object's group name when created automatically but does not

have to. The Id here is used to identify the object from protocol messages, and it could also contain some special characters not allowed in the Geo SCADA object name.

- Device Id – blank as this is a Node.
- Parent Node – empty as this is a Node. (Validation could be added for this field)

### 3.6.2 Device

The Device object configuration is:

The screenshot shows the 'Device' configuration window with the 'SparkplugB' tab selected. The form contains the following fields and values:

| Field            | Value                               |
|------------------|-------------------------------------|
| Enabled          | <input checked="" type="checkbox"/> |
| Severity         | High                                |
| Broker           | SparkplugB.Broker                   |
| Edge Node Id     | Node-REDEdgeNode                    |
| Parent Node      | SparkplugB.Node-REDEdgeNode.Node    |
| Area of Interest | World                               |
| Scan Interval    | 30S                                 |
| Scan Offset      | M                                   |
| Device Id        | TestDev2                            |

The specific fields here are:

- Edge Node Id – the name of this Device's Node within the namespace, as used for the Node.
- Device Id – the name of this Node within the namespace. Note that this will usually match the object's group name but does not have to. The Id here is used to identify the object from protocol messages, and it could also contain some special characters not allowed in the Geo SCADA object name.
- Parent Node – a database reference to the Node. (Validation could be added for this field)

### 3.6.3 Points

Finally, points are created to match the metrics in the list within the Birth messages for Node and Device. These are placed in a group/folder path appropriate to their name, so for example if a metric is named "pump1/speed" then the point is named "speed" within a subfolder named "pump1".

Metrics have an optional 'Alias' field which is populated with a number. This field is used (optionally) by Data messages in the Sparkplug B protocol to reduce message payload size.

The screenshot shows the 'Binary 01' configuration window with the 'SparkplugB' tab selected. The form contains the following fields and values:

| Field              | Value                                   |
|--------------------|-----------------------------------------|
| In Service         | <input checked="" type="checkbox"/>     |
| Device             | SparkplugB.Node-REDEdgeNode.TestDev2.De |
| Address Alias      | 1                                       |
| Sparkplug Name     | Binary 01                               |
| Sparkplug Datatype | 11                                      |

As with the Node and Device, the Sparkplug Name field works like the Id fields to identify a metric, and the point's actual name is created to match it but could be renamed to be different.

The point type is defined by the Sparkplug Datatype field, which is a number used to indicate the format, for example Integer, Float, Boolean, String. The mapping is fixed within a function of the code.

The driver automatically enables historic data storage and filtering so that all data is stored in the Geo SCADA historian. These configuration properties are set when new configuration is received, so if you do not wish for the driver to do this you could use templates and property overrides to control it.

### 3.7 Receiving New Configuration

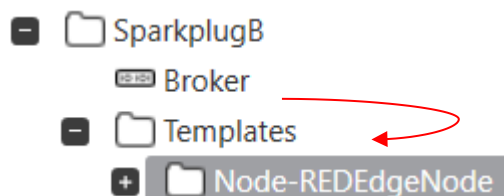
Each time a Sparkplug B Node or Device connects to the Broker it should resend the Birth message with all metric configuration data. This would be a burden for the server if it had to analyse and reconfigure all devices and points, so the driver stores a checksum (hash) of the Birth configuration data and will only attempt to reconfigure objects when the hash changes.

Metrics which are no longer within the Birth message are not deleted by the driver and therefore will be retained until a user wishes them to be removed. It could be a useful enhancement for the configuration process to find and mark such points with a data or configuration field to identify them as unused.

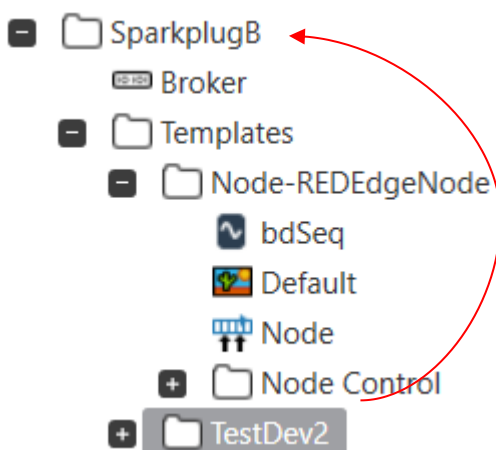
### 3.8 Making Templates of the Configuration

As well as automating the configuration process, this Sparkplug B driver integrates with the Geo SCADA templates and instances feature. A good way to understand the features is to start from the configuration created above and then use that to define templates.

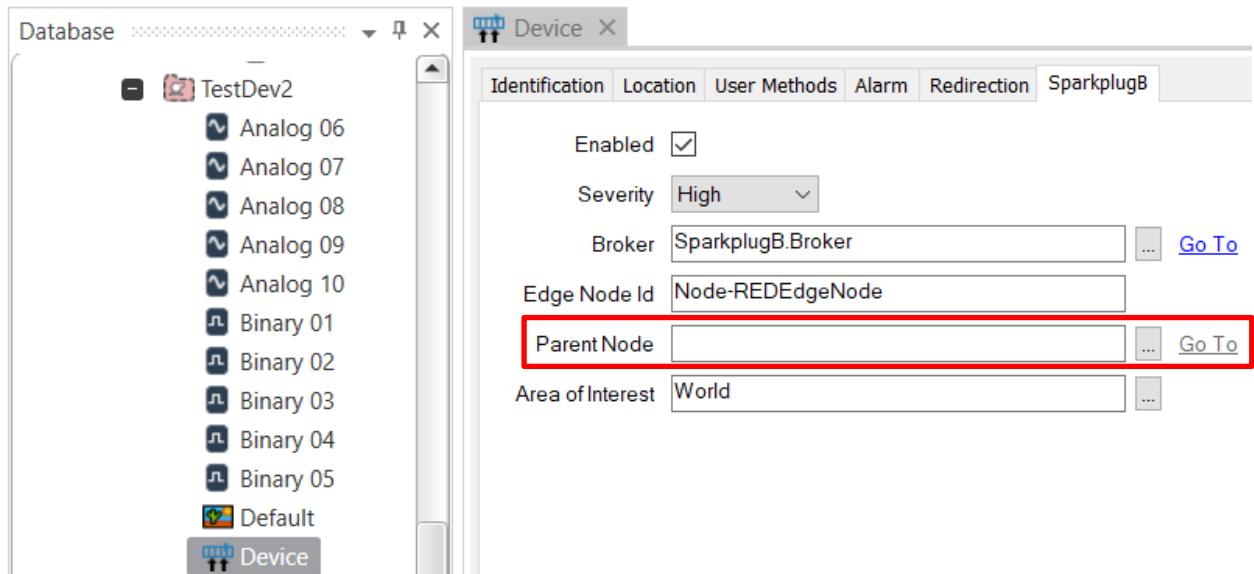
First, drag the configuration which the driver had made into the Templates folder you created:



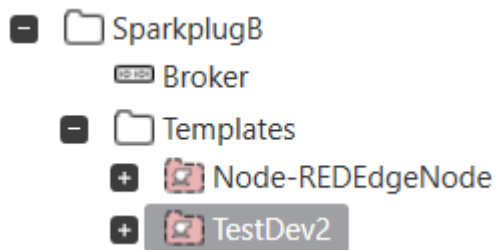
Then (optional, but recommended), drag the Device folder out into the parent Template folder:



If you do that then you will first need to clear the reference from the Device object to the Node before the template conversion will work:

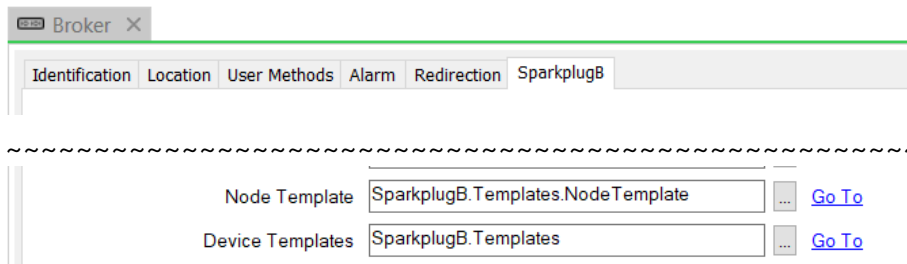


Finally convert these two groups into templates with the 'Convert' to Template context menu:



We now have templates for Node and Device, and they can be modified as needed, including the addition of mimics, trends, Logic and point properties such as ranges, limits and alarm settings. You may want to rename the Node template if you wish – there can only be one of these per Broker. The Device template must currently be named by the Device Id name in the Sparkplug B namespace. It may be useful to modify the driver to search by Device Id property instead of relying on the template name.

The broker can now be configured to refer to the Node template and the group containing the Device templates. Please configure as in this screenshot:

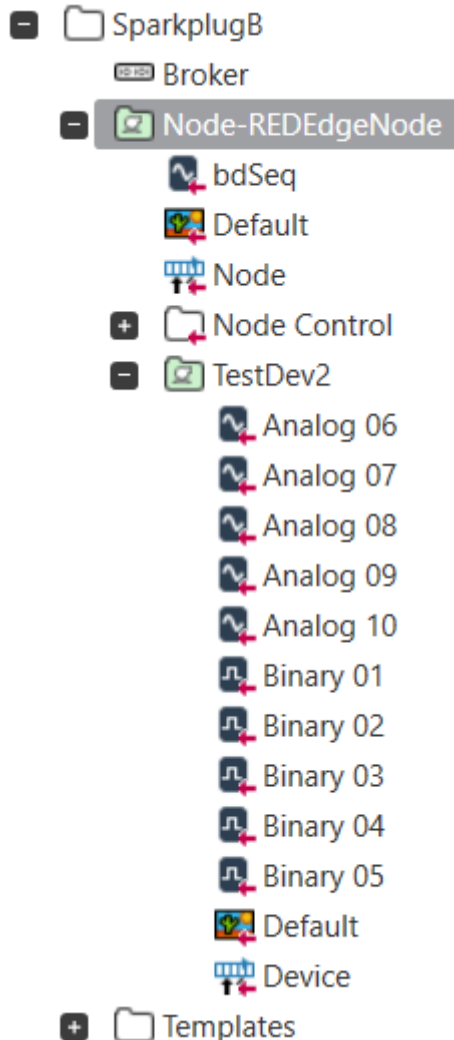


The Node Template will be named 'Node-REDEdgeNode' unless you have renamed it.

### 3.9 Automated Configuration of Instances

Now, go back to your simulated node and resend its Birth message. If you need to use the pick action to cause configuration, please execute it.

Now the driver has created configuration based on the templates, and the instances are therefore controlled by the configuration in each template. Your mimics within the templates will be used automatically for new automatic configuration.



We hope you can see the power of this:

- Configuration created automatically
- Templates created automatically
- All with user supervision!

### 3.10 Property Configuration

An additional feature of the driver uses the capability for the Sparkplug B protocol to handle additional user-definable properties for metrics. The driver includes sample functionality to set up the Units, Full Scale and Zero Scale values for analogue points. To use the feature, the Sparkplug B Birth payload message needs to have these properties defined. An example of this is shown here with the Node-Red data source.

Modify the latter part of the DBirth function and replace the analogue part with this:

```
var propertySet = {};  
propertySet["Units"] = { "type": "String", "value": "feet"};  
for (i = 6; i <= 10; i++) {  
    msg.payload.metrics.push(  
        {  
            "name": "Analog " + ('0'+i).slice(-2),  
            "alias": i,  
            "value": 0,  
            "type": "float",  
            "properties": propertySet  
        }  
    );  
}  
return msg;
```

This will then cause the Geo SCADA database field for units to be specified by the Birth message. The code can easily be made to handle other configuration fields, or could be made to do this dynamically for any field.

We hope you can see the power of this:

- All Geo SCADA Configuration fields created automatically

### 3.11 Processing Data

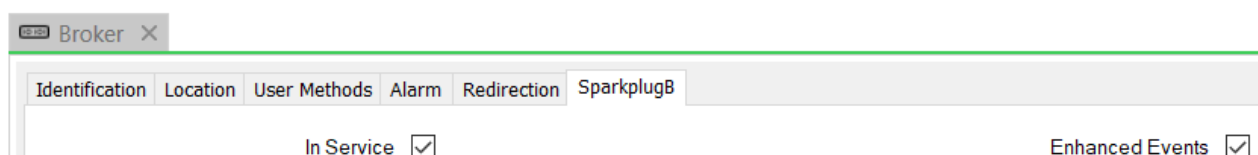
When a Birth message is sent, it includes the current value of metrics. The driver buffers these values until configuration is complete, and then calls for the values to be processed.

After this, all Data messages are processed in the normal way. Currently all values received by the driver are sent to the server straight away. It would be a better solution if there is a degree of buffering to reduce database loading.

### 3.12 How to Diagnose any Problems

This driver logs information to the server and driver log files, similar to other drivers. Please see the product help on Logging, and the DDK driver guide's relevant sections on logging and debugging.

An additional feature provided by this driver is Enhanced Events. This is enabled with a checkbox on the Broker object:



When checked, many driver log items will additionally be sent to the Geo SCADA Event log for the Broker or Node/Device objects. This brings an immediacy to driver behavior but at a significant performance cost. Please do not keep this checkbox enabled for long periods because excessive event logs will be produced. An example of the output is shown here:

Broker Events on area "Spar...plugB" on Local X

| Severity | Time                    | Source                           | Message                                                                | User   | Category          |
|----------|-------------------------|----------------------------------|------------------------------------------------------------------------|--------|-------------------|
| High     | 10/07/2020 15:33:30.117 | SparkplugB.Node-REDEdgeNode.Node | Point: SparkplugB.Node-REDEdgeNode.bdSeq Found a value.                | System | SparkplugB ND     |
| High     | 10/07/2020 15:33:30.002 | SparkplugB.Node-REDEdgeNode.Node | Point: SparkplugB.Node-REDEdgeNode.Node Control.Rebirth Found a value. | System | SparkplugB ND     |
| High     | 10/07/2020 15:33:30.001 | SparkplugB.Node-REDEdgeNode.Node | Process pending data from Birth.                                       | System | SparkplugB ND     |
| High     | 10/07/2020 15:33:30.000 | SparkplugB.Node-REDEdgeNode.Node | Data buffer not empty (in OnScan).                                     | System | SparkplugB ND     |
| High     | 10/07/2020 15:33:07.653 | SparkplugB.Node-REDEdgeNode.Node | Online                                                                 | System | Scanner Status    |
| High     | 10/07/2020 15:33:07.653 | SparkplugB.Node-REDEdgeNode.Node | Received configuration is the same as the saved version.               | System | SparkplugB ND     |
| High     | 10/07/2020 15:33:07.653 | SparkplugB.Node-REDEdgeNode.Node | Received Device Configuration.                                         | System | SparkplugB ND     |
| High     | 10/07/2020 15:33:07.653 | SparkplugB.Node-REDEdgeNode.Node | SparkplugCommError: Offline Last Will Received - Alarm cleared         | System | SparkplugB ND     |
| High     | 10/07/2020 15:33:07.653 | SparkplugB.Node-REDEdgeNode.Node | Received Birth Certificate. Time: 10/07/2020 14:33:07                  | System | SparkplugB ND     |
| High     | 10/07/2020 15:33:07.652 | SparkplugB.Broker                | Birth: Node-REDEdgeNode, Timestamp: 1594391587814 Sequence: 0          | System | SparkplugB Broker |
| High     | 10/07/2020 15:33:07.652 | SparkplugB.Broker                | RX Message bytes: 50                                                   | System | SparkplugB Broker |
| High     | 10/07/2020 15:33:07.652 | SparkplugB.Broker                | RX Topic: spBv1.0/SparkplugDevices/NBIRTH/Node-REDEdgeNode             | System | SparkplugB Broker |
| High     | 10/07/2020 15:33:06.719 | SparkplugB.Node-REDEdgeNode.Node | SparkplugCommError: Offline Last Will Received - Alarm raised          | System | SparkplugB ND     |
| High     | 10/07/2020 15:33:06.719 | SparkplugB.Node-REDEdgeNode.Node | Received Last Will and Testament. Seq: 0                               | System | SparkplugB ND     |
| High     | 10/07/2020 15:33:06.718 | SparkplugB.Broker                | RX Message bytes: 20                                                   | System | SparkplugB Broker |
| High     | 10/07/2020 15:33:06.718 | SparkplugB.Broker                | RX Topic: spBv1.0/SparkplugDevices/NDEATH/Node-REDEdgeNode             | System | SparkplugB Broker |

## 4 The Code

The code is written with onward development in mind. There are many comments, and a structure which you can take on and modify or extend.

### 4.1 Where Next?

#### ***Add point processing logic.***

Currently all new data is processed regardless of change. You could add on-change options which prevent data processing for identical values.

#### ***Device statistics***

Add counters for the different types of messages received or sent by the driver.

#### ***Point properties***

Extend the simple Units/Overrange and Underrange properties set from the protocol into other properties. Some example fields are in the code comments. You could add a mapping between property names and the Geo SCADA field names.