EcoStruxure™

Geo SCADA Expert

Basic MQTT Client

Simple Driver Development Kit Sample Driver

www.schneider-electric.com

Issue Details

Issue	Date	Author	Comments
1	14.08.20	S. Beadle	New.

References

Simple Driver Development Kit reference:
https://tprojects.schneider-electric.com/telemetry/display/CS/Technical+Guides

MIT License

Table of Contents

1 Introduction

This document describes the code for a basic MQTT Client 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 MQTT 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 control output, using a 'one Topic per point' model and simple text for point values, no time tagging.

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.

You can discuss these features, driver development and MQTT 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-BasicMQTTClient:

URL: https://github.com/GeoSCADA/Driver-BasicMQTTClient

1.2 The MQTT 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.

This driver uses simple text representations of data such as numbers to send and receive data. For example, an Analog point can be created to receive data on a Topic named "Site A/Analog" and the data might be a string value "1.234". A point can have a control capability, where the controlled value (also text) is sent to the Topic named in the point's Control tab of the configuration form.

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 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/.

1.2.3 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.

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

2 Building the Driver

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

2.1 MQTTClient

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

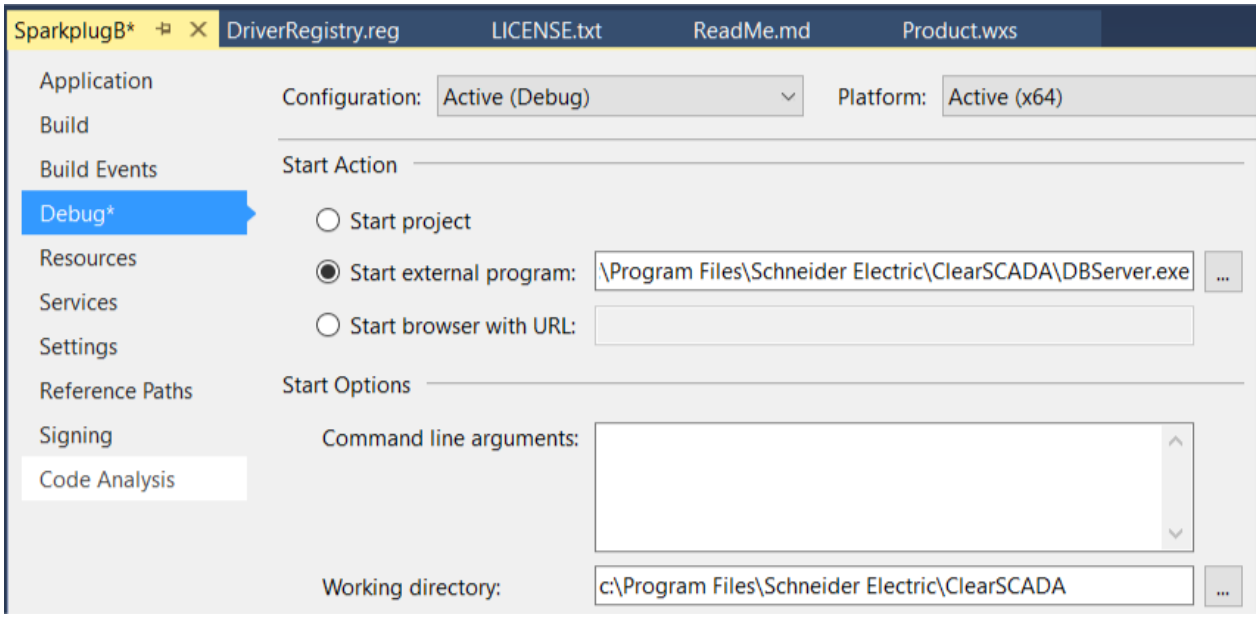
Alternatively, you can copy this .dll to your own location from a Geo SCADA installation. In this case you may set up multiple builds for different Geo SCADA versions if you wish. Note that you need the major version number of this .dll (e.g. 81) to match the major version of the Geo SCADA target.

2.1.2 Files

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

2.1.3 Debugging

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



2.2 DriverMQTTClient

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

2.2.1 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 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.

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)\MQTTClient.dll c:\Program Files\Schneider Electric\ClearSCADA\

copy $(TargetDir)\MQTTClient.PDB c:\Program Files\Schneider Electric\ClearSCADA\

copy $(TargetDir)\M2Mqtt.Net.dll c:\Program Files\Schneider Electric\ClearSCADA\

copy $(TargetDir)\DriverMQTTClient.exe c:\Program Files\Schneider
Electric\ClearSCADA\

copy $(TargetDir)\DriverMQTTClient.PDB c:\Program Files\Schneider
Electric\ClearSCADA\
```

2.3 DriverMQTTClientInstaller

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:

"DriverMQTTClient.exe"

"MQTTClient.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\DriverMQTTClient]
@="MQTTClient"
"AssemblyName"="C:\\Program Files\\Schneider
Electric\\ClearSCADA\\MQTTClient.dll"
"DebugMode"="False"
"TaskName"="DriverMQTTClient.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:

```
%WindDir%\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe MQTTClient.dll
```

3.2 Configure Objects

To get started, create a group for configuration objects, then create a Broker – create the MQTTClient driver broker.

3.3 Configure a Broker

A Broker object defines the connection to the broker.

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.

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 Testing

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 include Python and Node-Red.

3.5 Processing Data

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.

3.6 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.

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.