PLCnext Technology - MQTT Client

Date	Version
29.03.2019	1.0.2

IMPORTANT NOTE

Installing or uninstalling this Function Extension will automatically restart the PLC.

Description

MQTT Client is a PLCnext Technology Function Extension that exchanges data between Global Data Space (GDS) ports and MQTT server topics.

Requirements

This Function Extension is designed for the AXC F 2152 controller with minimum firmware version 2019.3.

Features

The Function Extension connects to a single MQTT broker over an unencrypted or encrypted (SSL/TLS) connection.

Only one client, and one concurrent server connection, is currently supported.

The client will publish the value of each configured GDS ports at the specified frequency. Data from each GDS port can be published to multiple MQTT topics.

The client will subscribe to all configured MQTT topics. The payload from any new message on a subscribed MQTT topic can be written to one or more GDS ports.

There are no checks done on incoming MQTT message payloads. It is the users responsibility to ensure that messages on each subscribed topic contain payload data that is compatible with all the GDS ports associated with that subscription.

The following GDS data types are currently supported: Bool, Int8, Int16, Int32, Int64, Uint8, Uint16, Uint32, Uint64, Real32, Real64, String, DateTime.

'String' data is always published with a terminating NULL character.

When subscribing to 'String' data, incoming message payloads must always include a terminating NULL character.

Complex data types (including Arrays and Structures) are not currently supported.

Client-side message persistence has not been implemented in this version.

Quick start

This example exchanges data between a PLC and an iPhone or iPad via a public MQTT broker, over an unencrypted connection. It requires a PLC that is connected to the internet, and a PC with access to both the PLC and the internet. For this example, the PC must have PLCnext Engineer software installed.

1. Create an IEC 61131 project in PLCnext Engineer version 2019.3, with the following features:
    - One AXC F 2152 PLC with firmware version 2019.3
    - one program called "Main"
    - one program OUT port called "PubMessage" of type STRING
    - one program IN port called "SubMessage" of type STRING
    - one instance of the Main program, called "MainInstance"
2. Download the PLCnext Engineer project to the PLC.
3. Go online to the PLC and change the value of the "PubMessage" variable in the "MainInstance" program instance.
4. Install the MQTTool app on an iPhone or iPad.
5. Open the MQTTool app and connect to the public Mosquitto test broker (test.mosquitto.org, port 1883).
6. On a PC, create a text file named `mqtt_gds.settings.json`, containing the following configuration:

```
{ "brokers":[{
    "host": "tcp://test.mosquitto.org:1883",
    "client_name": "MyPLCnext",
    "connect_options":{
      "will_options":{
        "topic": "last_will_topic",
        "payload": "auf wiedersehen"
      }
    },
    "publish_data":[{
      "port"   : "Arp.Plc.Eclr/MainInstance.PubMessage",
      "qos": 0,
      "retained": false,
      "topics" :[
        "MyPubTopic"
      ]
    }],
    "subscribe_data":[{
      "topic" : "MySubTopic",
      "ports" :[
        "Arp.Plc.Eclr/MainInstance.SubMessage"
      ]
    }]
  }]
}
```

In the configuration file, change the `client_name` property to a value that is likely to be unique on a public server.

Change the publish and subscribe `topic` properties to values that are likely to be unique on a public server.

7. Using WinSCP (Windows) or `scp` (Linux), copy the `mqtt_gds.settings.json` file to the following directory on the PLC: `/opt/plcnext/projects/MqttClient/`. Create this directory on the PLC if it does not exist already.

   The default login credentials for the PLC are:

   - User name : admin
   - Password : <printed on the PLC housing>

   Make sure that the file has 'read' privileges for all users on the PLC.

8. Using a web browser, log in to the PLCnext Store and check that your PLC is registered on your account:

   - Select your use name (top right), and select "PROFILE".
   - If your PLC does not appear in the "Devices" list, select the "+ new device" button and follow the registration process. The Device ID can be found in the "Proficloud" section of the Web Based Management page on the PLC.

9. Install the "MQTT Client" Function Extension from the PLCnext Store.

10. In the MQTTool app on the iPhone or iPad, subscribe to the topic name that was entered in the `publish_data` section of the configuration file (e.g. "MyPubTopic").

11. The messages received on the iPhone or iPad now show the value of the PubMessage variable in the PLC.

12. On the iPhone or iPad, publish a message to the topic that was entered in the `subscribe_data` section of the configuration file (e.g. "MySubTopic"). This message now appears as the value of the SubMessage variable in the PLC.

You are now ready to create an MQTT Client configuration for your own application.

Configuration reference

The MQTT Client is configured with the file `mqtt_gds.settings.json`. This file must be located in the following directory on the PLC:

`/opt/plcnext/projects/MqttClient/`

This directory must be created on the PLC if it does not exist already.

For encrypted connections, a server certificate, client certificate and client private key may be required. These files should be copied to the PLC using WinSCP (Windows) or scp (Linux). The user is free to place these files anywhere on the PLC file system. The absolute path to these files must then be specified in the relevant `ssl_options` configuration fields.

After modifying any of these files, the PLCnext Runtime must be restarted.

Configuration file details

The configuration file `mqtt_gds.settings.json` must comply with the JSON schema defined in the file `mqtt_gds.schema.json`. After installing the MQTT Client Function Extension, this schema file can be found in the following directory on the PLC:

`/opt/plcnext/apps/60002172000048/`

A valid configuration consists of an array of broker objects. Each broker object represents one MQTT client-broker connection.

The following is a list of all available broker properties. Optional properties that do not appear in the configuration file will be given default values.

broker properties

Name	Required	JSON type	Description
host	Yes	string	The address of the server to connect to, specified as a URI.[1]
clientId	Yes	string	A client identifier that is unique on the server being connected to.
timeout	No	integer	Client timeout value in milliseconds.
status_port	No	string	The name of a boolean GDS port that will receive the client connection status.
reconnect_port	No	string	The name of a boolean GDS port that, on a rising edge, will trigger a reconnect attempt.
connect_options	Yes	object	The connection options. See table below.

Name	Required	JSON type	Description
publish_data	No	array of objects	MQTT publish information. See table below.
subscribe_data	No	array of objects	MQTT subscribe information. See table below.

Note:

1. The host *must* be specified in the following format:

   protocol://host:port

   ... where *protocol* must be *tcp*, *ssl*, *ws* or *wss*. For *host*, you can specify either an IP address or a domain name.

---

connect_options

Name	Required	JSON type	Description
keep_alive_interval	No	integer	60
connect_timeout	No	integer	30
username	No	string	NULL
password	No	string	NULL
max_inflight	No	integer	0
is_clean_session	No	boolean	true
mqtt_version	No	integer	0
automatic_reconnect	No	boolean	false
min_retry_interval	No	integer	0
max_retry_interval	No	integer	0
will_options	Yes	object	
ssl_options	No	object	

Note:

1. mqtt_version:
   - 0 = default: start with 3.1.1, and if that fails, fall back to 3.1
   - 3 = only try version 3.1
   - 4 = only try version 3.1.1

will_options

Name	Required	JSON type	Description
topic	Yes	string	The LWT message is published to the this topic.
payload	No	string	The message that is published to the Will Topic.
qos	No	integer	The message Quality of Service.
retained	No	boolean	Tell the broker to keep the LWT message after send to subscribers.

ssl_options

Name	Required	JSON type	Description
trust_store	No	string	The filename containing the public digital certificates trusted by the client.
key_store	No	string	The filename containing the public certificate chain of the client.
private_key	No	string	The filename containing the client's private key.
private_key_password	No	string	The password to load the client's privateKey (if encrypted).
enabled_cipher_suites	No	string	The list of cipher suites that the client will present to the server during the SSL handshake.
enable_server_cert_auth	No	boolean	Enable verification of the server certificate.

publish_data

publish_data must be an array of objects with the following properties:

Name	Required	JSON type	Description
port	Yes	string	The 'OUT' port from which data will be published[1].
period	No	integer	The publish frequency in seconds (max 86,400). If not specified, period=500ms.
qos	Yes	integer	The message Quality of Service.
retained	Yes	boolean	Tell the broker to keep messages after send to subscribers.
topics	Yes	array of strings	Message are published to all these topics.

Note:

1. Ports published from PLCnext Engineer projects must be specified in the following format:

   Arp.Plc.Eclr/ProgramInstance.PortName

   ... where *ProgramInstance* must be the name of the program instance in the PLCnext Engineer project, and *PortName* must the name of an OUT port variable defined in that program.

---

subscribe_data

subscribe_data must be an array of objects with the following properties:

Name	Required	JSON type	Description
topic	Yes	string	This topic is subscribed Message are published to all these topics.
ports	Yes	array of strings	The 'IN' ports to which subscription data will be written[1].

Note:

1. Ports in PLCnext Engineer projects that subscribe to MQTT topics must be specified in the following format:

   Arp.Plc.Eclr/ProgramInstance.PortName

   ... where *ProgramInstance* must be the name of the program instance in the PLCnext Engineer project, and *PortName* must the name of an IN port variable defined in that program.

Configuration examples

Examples of configuration files that you can use as a starting point for your own project are available here. Remember that your own configuration file must *always* be named `mqtt_gds.settings.json`.

Known issues

If the connection to the broker is lost, the client will attempt to reconnect, but will eventually give up. In this case, the PLC must be restarted in order to trigger more connection attempts.

Source code

This Function Extension uses two PLCnext Technology components. The source code for these components is available on Github:

- MQTT Client
- MQTT GDS Connector

Community contributions to these open-source projects are welcome.

Support and Feature Requests

Please raise any relevant issues on the MQTT GDS Connector Github project.

For general support and further information on PLCnext Technology, please visit the PLCnext Community website.

This code uses:

1. JSON for Modern C++

   The class contains the UTF-8 Decoder from Bjoern Hoehrmann which is licensed under the MIT License (see above). Copyright © 2008-2009 Björn Hoehrmann bjoern@hoehrmann.de

   The class contains a slightly modified version of the Grisu2 algorithm from Florian Loitsch which is licensed under the MIT License (see above). Copyright © 2009 Florian Loitsch

2. valijson

