PLCnext Technology - AWS IOT Client

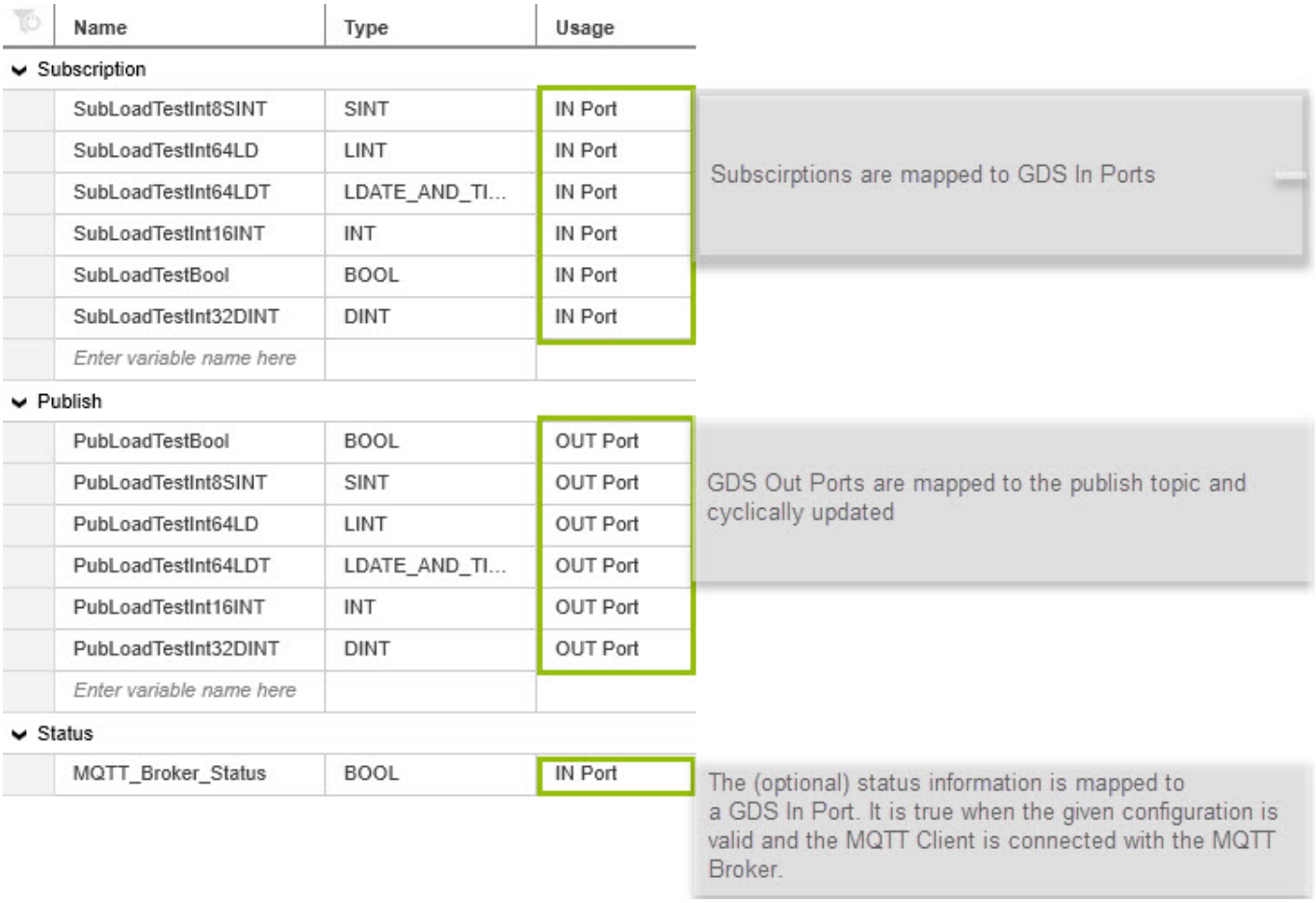
Date	Version
25.05.2019	1.1.1

IMPORTANT NOTE

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

Description

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



The component is configured with the file `mqtt_gds.settings.json` which is stored locally on the device.

```
{
    "brokers":[{
    "host": "ssl://abcdefghijklmn-abc.iot.us-west-2.amazonaws.com:8883",
    "client_name": "MyPLCnext",
    "connect_options":{
        "will_options":{
        "topic": "last_will_topic",
        "payload": "auf wiedersehen"
        },
        "ssl_options":
        {
        "trust_store":
"/opt/plcnext/projects/AwsIotClient/AmazonRootCA1.pem",
        "key_store": "/opt/plcnext/projects/AwsIotClient/d123c12345-
certificate.pem.crt",
        "private_key": "/opt/plcnext/projects/AwsIotClient/d123c12345-
private.pem.key",
        "enable_server_cert_auth": true
        }
    },
    "publish_data":[{
        "port"   : "Arp.Plc.Eclr/MainInstance.PubMessage",
        "qos": 0,
        "retained": false,
        "topics" :[
        "MyPubTopic"
        ]
    }],
    "subscribe_data":[{
        "topic" : "MySubTopic",
        "ports" :[
        "Arp.Plc.Eclr/MainInstance.SubMessage"
        ]
    }]
    }]
}
```

The entries in this file must conform to the defined JSON schema (please refer for more details the "configuration reference" section below).

Requirements

- AXC F 2152 with minimum firmware version 2019.3
- Valid account for the PLCnext Store with payment credentials (not needed for the trial version)
- The PLCnext Control must be connected to the internet and must be registered in the PLCnext Store

Features

- The AWS IOT Client app is compatible with MQTT version 3.1 and 3.1.1
- Automatic reconnect to the AWS IOT Server
- Easy handling due to GDS port mapping, no further configuration effort
- Cyclic update of Publish Topics, individually adjustable (minimum 500ms)
- Support of the following data types* (Bool, Int8, Int16, Int32, Int64, Uint8, Uint16, Uint32, Uint64, Real32, Real64, String**, DateTime)

**The named data types are C++ types. Please refer the PLCnext Technology Handbook (available in the PLCnext Community) for the corresponding IEC 61131-3 or Matlab™ Simulink data types.*
***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.*

Quick start

This example sends data from a PLC to an AWS IOT server. 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. In your AWS account, register your PLC as an AWS IOT device by following these instructions. Save the Server Certificate, Private Certificate and Private Key on your PC.
2. Make sure that your AXC F 2152 runs on firmware version >=2019.3, and that it has access to the Internet.
3. Register for a user account and authorize your AXC F 2152 in the PLCnext Store.
4. Deploy the app via the PLCnext Store.
5. Create an IEC 61131 project in PLCnext Engineer with the following configuration:

- one AXC F 2152 PLC with the template version >=2019.0 LTS
- one program called "Main"
- one program OUT port called "PubMessage" of STRING type
- one program IN port called "SubMessage" of STRING type
- one instance of the Main program, called "MainInstance"

1. Download the PLCnext Engineer project to the PLC.
2. Go online to the PLC and change the value of the "PubMessage" variable in the "MainInstance" program instance.

1. On a PC, create a text file named `mqtt_gds.settings.json`, containing the following configuration:

```
{
  "brokers":[{
    "host": "ssl://abcdefghijklmn-abc.iot.us-west-
2.amazonaws.com:8883",
    "client_name": "MyPLCnext",
    "connect_options":{
      "will_options":{
        "topic": "last_will_topic",
        "payload": "auf wiedersehen"
      },
      "ssl_options":
      {
        "trust_store":
"/opt/plcnext/projects/AwsIotClient/AmazonRootCA1.pem",
        "key_store": "/opt/plcnext/projects/AwsIotClient/d123c12345-
certificate.pem.crt",
        "private_key": "/opt/plcnext/projects/AwsIotClient/d123c12345-
private.pem.key",
        "enable_server_cert_auth": true
      }
    },
    "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 `host` property to the name of your AWS IOT server. This name *must* be preceeded by "ssl://", and *must* include the port number 8883.

   Change the `key_store` and `private_key` properties to include the name of your own AWS IOT device private certificate and key.

2. Using WinSCP (Windows) or `scp` (Linux), copy the `mqtt_gds.settings.json` file, the AWS Server Certificate, your Private Certificate, and your Private Key, 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 files have 'read' privileges for all users on the PLC.

3. Restart the PLC.
4. In your AWS IOT dashboard, select "Monitor". Check that the PLC has connected and is publishing data. select "Test" and subscribe to the topic name that was entered in the `publish_data` section of the configuration file (e.g. "MyPubTopic").
5. The AWS IOT Test screen now shows the value of the PubMessage variable in the PLC.

Configuration reference

The AWS IOT 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.

A server certificate, client certificate and client private key are required for all AWS IOT devices. 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`. This schema file must be located in the following directory on the PLC:

`/opt/plcnext/apps/60002172000053/`

Note: Every invalid configuration (invalid configuration or missing mandatory field) leads to a stop of the app. Debug information will appear in the Output.log file of the PLC: `/opt/plcnext/logs/Output.log`

---

server properties

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

Note: The AWS IOT Client version 1.1 only supports one server connection.

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.
status_port	No	string	The name of a boolean GDS port that will receive the client connection status.[2]
connect_options	Yes	object	The connection options. See table below.
publish_data	No	array of objects	MQTT publish information. See table below.

Name	Required	JSON type	Description
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.

2. Ports on PLCnext Engineer programs 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 a port variable defined in that program.

---

connect_options

Name	Required	JSON type	Default value	Description
username	No	string	NULL	The user name to use for the connection.
password	No	string	NULL	The password to use for the connection.
mqtt_version	No	integer	0	The version of MQTT to be used on the connect.[1]
will_options	Yes	object		The LWT options to use for the connection. See table below.
ssl_options	No	object		The SSL options to use for the connection. See table below.

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
2. LWT = Last Will Topic

---

will_options

Name	Required	JSON type	Description
topic	Yes	string	The LWT message is published to the this topic.
payload	Yes	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 sent to subscribers.

ssl_options

The ssl option is only needed when encrypted *ssl* communication is used. Depending on the MQTT Broker policies, different configurations may be needed.

Name	Required	JSON type	Description
trust_store	Yes	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.

server configuration example

The following example shows a valid server configuration.

```
{ "servers":[{
 "host": "ssl://abcdefghijklmn-abc.iot.us-west-2.amazonaws.com:8883",
 "client_name": "AWS_Test_App",
 "connect_options":{
     "will_options":{
     "topic": "last_will_topic",
     "payload": "auf wiedersehen"
     },
     "ssl_options":
     {
     "trust_store": "/opt/plcnext/projects/AwsIotClient/AmazonRootCA1.pem",
     "key_store": "/opt/plcnext/projects/AwsIotClient/d123c12345-
certificate.pem.crt",
     "private_key": "/opt/plcnext/projects/AwsIotClient/d123c12345-
private.pem.key",
     "enable_server_cert_auth": true
     }
 }
 }]
}
```

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[2].
retained	Yes	boolean	Tell the broker to keep messages after send to subscribers[3].
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.

2. The MQTT app version 1.1 only supports QoS 0

3. The MQTT app version 1.1 does not support 'retained'.

publish_data configuration example

The following example shows a valid configuration.

Note: Make sure that the assigned GDS port exists and that the data type is correct. An incorrect configuration will prevent the app from starting.

```
"publish_data":[{
  "port"   : "Arp.Plc.Eclr/TestBench1.PubLoadTestBool",
  "qos": 0,
  "retained": false,
  "topics" :[
    "LoadTestBool"
  ]
}
```

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.

subscribe_data configuration example

The following example shows a valid configuration.

Note: Make sure that the assigned GDS port exists and that the data type is correct. An incorrect configruation will prevent the app from starting.

```
"subscribe_data":[{
   "topic" : "LoadTestBool",
   "ports" :[
     "Arp.Plc.Eclr/TestBench1.SubLoadTestBool"
   ]
 }
```

Configuration examples

A complete configuration example can be found here.
Remember that your own configuration file must **always** be named `mqtt_gds.settings.json`.

Known issues and limitations

- Only one client, and one concurrent server connection, is currently supported
- When the network connection to the broker is lost and restored, and a manual or automatic reconnect is triggered, the AWS IOT Client will block for precisely the number of milliseconds specified by the broker "timeout" property (default: 300 seconds).
- Complex data types (including Arrays and Structures) are not currently supported.
- Only QoS 0 is supported
- The app checks the assigned GDS port in terms of availability and type during the start-up process. Any changes to GDS ports (delete, rename or type) during operation (e.g. if a modified PLCnext Engineer project is downloaded without stopping the PLC) can lead to an undefined behaviour!

Error handling

The app logs the complete startup, connection and error history into the Output.log file (`/opt/plcnext/logs/Output.log`) of the PLC which gives the user a reliable indication of the current state of the app. In addition, a status port can be configured (please refer the chapter 'broker properties'). The value *true* indicates that the app was successfully started (configuration correct) and that connection to the configured MQTT Broker could be established. The values goes to *false* when the app could not be started (wrong configuration) or when the connection to the MQTT Broker could not be established or is interrupted.

Note: Phoenix Contact strongly recommend the usage of the "status_port".

Diagnostic Log

This chapter explains the most important diagnostic messages and is intended to support troubleshooting the AWS IOT app.

All diagnostics are printed in the output.log file (`/opt/plcnext/logs/Output.log`).

In the following is a positive start up of the app logged and described.

```
PxceTcs.Mqtt.GdsConnectorComponent    INFO  - Loaded configuration from
file: /opt/plcnext/projects/MqttClient/mqtt_gds.settings.json
PxceTcs.Mqtt.GdsConnectorComponent    INFO  - Loaded configuration schema.
```

The configuration was loaded successfully, no schema violations were detected.

Note: Every schema violation (missing json separators or missing mandatory fields) will lead to an error at this position. The message indicates the error type.

```
PxceTcs.Mqtt.GdsConnectorComponent    INFO  - No reconnect port has been
specified.
PxceTcs.Mqtt.GdsConnectorComponent    INFO  - Created MQTT Client with ID:
56022
PxceTcs.Mqtt.GdsConnectorComponent    INFO  - No MQTT SSL/TLS Options
provided. Defaults will be used.
PxceTcs.MqttClient.MqttClientManager  INFO  - Connecting to MQTT server
PxceTcs.MqttClient.MqttClientManager  INFO  - Connected
PxceTcs.Mqtt.GdsConnectorComponent    INFO  - Connected to MQTT Client
56022
```

After the configuration was loaded, generates the MQTT client a random client ID and tries to connect with the defined MQTT Broker.

Note: Connection or certification problems will lead to an error at this position.

```
PxceTcs.Mqtt.GdsConnectorComponent    INFO  - Subscribed to MQTT topic
LoadTestBool
PxceTcs.Mqtt.GdsConnectorComponent    INFO  - Subscribed to MQTT topic
LoadTestBool1
PxceTcs.Mqtt.GdsConnectorComponent    INFO  - SetupConfig(): Worker thread
has been started.
```

As soon as the connection is valid, the MQTT Client will start to publish and subscribe the configured topics. The cyclic update is performed with a worker thread (500ms). The app is now running.

Known issues

If the connection to the server 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.

Support and Feature Requests

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

How to get support

The AWS IOT Client app is supported in the forum of the PLCnext Community. Please raise an issue with a detailed error description and always provide a copy of the Output.log file.

This code uses:

1. JSON for Modern C++

2. valijson

