The MQTT Driver

The MQTT driver provides a method to publish values from a North device to an MQTT message broker. The broker stores these values and delivers them to any IoT application that subscribes to them. A range of MQTT message brokers are available, both cloud hosted and on-premises. Available for Commander and ObSys.

This document relates to MQTT driver version 2.0

Please read the *Commander Manual* or *ObSys Manual* alongside this document, available from *www.northbt.com*

Contents

Purpose of MQTT Driver

The MQTT driver provides a method to publish values from a North device to an MQTT (Message Queuing Telemetry Transport) message broker. The broker stores these values and delivers them to an IoT/M2M application that subscribes to the data topic. MQTT message brokers are available as part of many IoT platforms, both cloud hosted and on-premises.

MQTT uses a publish-subscribe model. The MQTT driver publishes a value from Essential Data within the North device using a topic name. Integrate data collected using North interface technology directly into your own IoT application by subscribing to these topics. Optionally, the driver can subscribe to a topic to receive value changes from your application.

The driver connects to an IP network, with access to an MQTT message broker provided on the local network or Internet.

The JSONNotify is also available, sending data to an endpoint server using Webhooks.

Values

MQTT presents values from the North device's Essential Data. As a value updates, or periodically, the values are published to an MQTT broker in JSON format.

Essential Data contains 640 values on Commander, and 1280 values on ObSys. Access to these values can be controlled by configuring a page range filter and privilege levels within the driver.

Prerequisites

An MQTT broker is required with support for non-secure TCP port 1883 (TLS is not supported by Commander).

MQTT brokers providing QoS level 1, and protocol version 5 or 3.1.1 are supported.

If the MQTT broker is Internet-based, the North device should have a DNS Server address and Gateway address configured.

Detailed Operation

MQTT is a lightweight protocol that's designed for connecting power-constrained devices over low-bandwidth networks. MQTT is the preferred protocol for connecting IoT devices to the cloud. Platforms such as HiveMQ, and Mosquitto all provide MQTT connectivity.

The protocol uses a publish-subscribe model. This is event driven, allowing messages to be pushed to clients.

MQTT Broker

The central communication hub is the MQTT broker (Fig. 1). It is in charge of dispatching all messages between the senders and receivers.

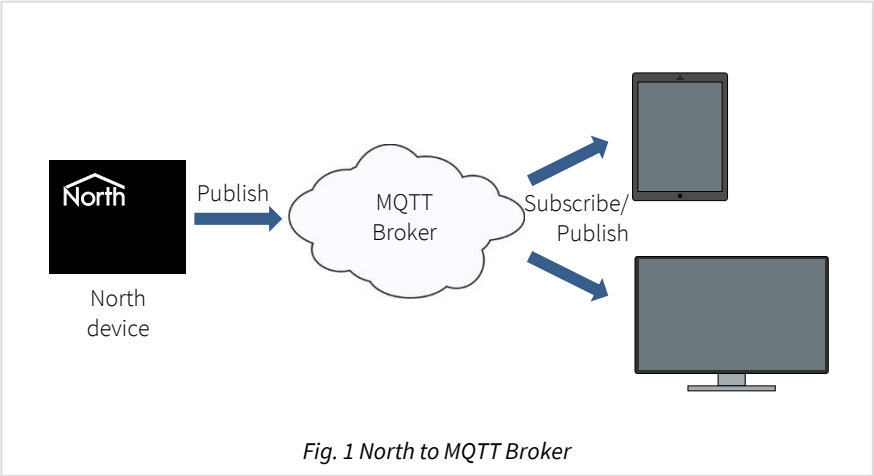

Fig. 1 North to MQTT Broker

The driver maintains a permanent connection to the broker. It publishes a value using a topic name. Your application can subscribe to these topics to receive the values.

See *Appendix A* for examples of connecting to MQTT brokers.

Topic Name

When the driver publishes a message to the broker, it includes a topic name in the message. This topic is the routing information for the broker. Each client that wants to receive messages subscribes to a certain topic and the broker delivers all messages with the matching topic to the client. Therefore the clients don't have to know each other, they only communicate using this topic.

A topic is a text string that can have levels of hierarchy, each separated by a slash ('/'). A sample topic for sending values from the North device could be 'North_MQTT/Building_5/HVAC'. A client could subscribe to this exact topic, or use a wildcard for a group of topics.

The driver publishes each value to the broker using a separate MQTT message. These messages can either all use the same topic name, or each value can have a unique topic name by incorporating a label from the value. A unique topic name for each value could be used alongside the retain value option.

You can set the topic name within the driver's *Publish/Subscribe* object.

Security

To protect against unauthorised clients, the driver can identify itself with the broker by supplying a user name and password.

On Commander, the driver supports non-secure MQTT only, TLS connections are not supported so a VPN should be used. On ObSys, the driver can additionally support TLS connections.

Publish Message

The MQTT driver sends a publish message containing a single value from Essential Data within the North device. The message can be sent either on change-of-value, or periodically.

Messages are sent in JSON format with utf-8 encoding.

Message Content

```
{
    "id": number,
    "name": string,
    "value": various,
    "status": string
    "time": string,
    "units": string,
    "type": string
}
```

Properties

Key	Value	Description
id	Number	Essential Data object identifier is in the range 1…1280 (depending on platform)
name	String	Combined label of page and object. The name can be prefixed with a device identifier if required
value	Number, Boolean, String, or Array	Value of object. The value depends on the object type configured in Essential Data: Float, Num, ENum – Number OffOn, NoYes – Boolean Text – String (max. 32 chars) DateTime – String in ISO-8601 format, '*yyyy-mm-dd*T*hh:mm:ss*' Date – String in ISO-8601 format, '*yyyy-mm-dd*' Times – Array of objects – each containing a start (s) and end (e) time Profile – Array of objects – each containing a time (s) and value (v)
s	String	Start time in 24hr format, 'hh:mm'. Provided with Times and Profile types
e	String	End time in 24hr format, 'hh:mm'. Only provided with times types
v	Number	Profile value. Only provided with profile types
status	String	Status of the value: 'ok' – value healthy, 'alarm' – out-of-range alarm, 'comms' – communications fault
time	String	Date and time (UTC) in ISO-8601 format, '*yyyy-mm-dd*T*hh:mm:ss*Z'
units	String	Units, if available, for value (max. 8 chars)
type	String	Essential Data object type, one of the following: text, noyes, offon, num, enum, float, datetime, date, times, or profile
ea	Array	Array of strings containing a label for each enumerated value (value-0, value-1, etc). Only provided with enum types

Example

```
{
   "id": 17,
   "name": "UPS Status - Load power",
   "value": 12.3,
   "status": "ok",
   "time": "2013-12-31T23:40:16Z",
   "units": "W",
   "type": "float"
}
```

Subscribe Message

The MQTT driver expects to receive a subscribe message containing a single value to write to Essential Data. The 'id' and 'value' keys must be present, any other properties will be ignored.

Objects must be set to adjustable within Essential Data, with an adequate access security level.

Messages must be sent in JSON format with utf-8 encoding.

Message Content

```
{
    "id": number,
    "value": various
}
```

Properties

Key	Value	Description
id	Number	Essential Data object identifier is in the range 1…1280
value	Number, Boolean, String, or Array	Value of object. DateTime – String in ISO-8601 format, '*yyyy-mm-dd*T*hh:mm:ss*' Date – String in ISO-8601 format, '*yyyy-mm-dd*' Times – Array of objects – each containing a start (s) and end (e) time Profile – Array of objects – each containing a time (s) and value (v)
s	String	Start time in 24hr format, 'hh:mm'. Provided with Times and Profile types
e	String	End time in 24hr format, 'hh:mm'. Only provided with times types
v	Number	Profile value. Only provided with profile types

Example

```
{
   "id": 17,
   "value": 14
}
```

Publish Message (Data Manager)

Available on ObSys platform only.

The MQTT driver sends a publish message with a single value from Data Manager. The message can be sent periodically, for example every 15 or 30 minutes.

Messages are sent in JSON format with utf-8 encoding.

Message Content

```
{
    "name": string,
    "value": number,
    "status": string
    "time": string,
    "units": string,
}
```

Properties

Key	Value	Description
name	String	Data Manager channel label. The name can be prefixed with a device identifier if required
value	Number	Value of object (floating point number)
status	Number	Status of the value: 'ok' – value healthy, 'alarm' – out-of-range alarm, 'comms' – communications fault
time	String	Date and time (UTC) in ISO-8601 format, '*yyyy-mm-dd*T*hh:mm:ss*Z'
units	String	Units, if available, for value (max. 8 chars)

Example

```
{
   "name": "UPS Status Load power",
   "value": 12.3,
   "status": 0,
   "time": "2023-12-11T15:07:46Z",
   "units": "W"
}
```

Using the Driver

On ObSys and Commander, the MQTT driver is pre-installed. Once started, you will need to set up the driver before it can communicate with an MQTT broker.

Starting the Interface

💻 To start an interface using the MQTT driver, follow these steps:

→ **Start Engineering** your North device using ObSys

→ Navigate to **Configuration, Interfaces,** and set an unused **Interface** to 'MQTT' to start the particular interface

→ Navigate to the top-level of your North device and re-scan it

The driver setup object (Mc), labelled **MQTT Setup**, should now be available. If this object is not available, check an interface licence is available and the driver is installed.

Setting up the Driver

💻 To set up the driver, follow these steps:

→ Navigate to the **MQTT Setup** object (M*c*). For example, if you started interface 1 with the driver earlier, then the object reference will be 'M1'

→ Navigate to **MQTT Broker** and set the **Host name** and **Client ID** for your broker

→ Navigate to **Publish/Subscribe** and set the **Publish Topic Name**.

Checking Communications

After configuring Essential Data or Extra Data, the driver will automatically connect to send information to the broker.

Use the driver objects **MQTT Broker connected** (DS) to check if the driver has connected to the broker. If it has not connected, navigate to MQTT Broker and check **Connection State** (CS), **Connect Last Response** (CR) and **Disconnect Last Reason** (DR) objects for more information.

Object Specifications

Once an interface is started, one or more extra objects become available within the top-level object of the device. As with all North objects, each of these extra objects may contain sub-objects, (and each of these may contain sub-objects, and so on) - the whole object structure being a multi-layer hierarchy. It is possible to navigate around the objects using the ObSys Engineering Software.

Each object is specified below, along with its sub-objects.

Device Top-Level Objects

When an interface is started using the MQTT driver, the objects below become available within the top-level object of the device. For example, if interface 1 is started, then the object reference 'M1' becomes available.

Description	Reference	Type
MQTT Setup Set up the MQTT driver, started on interface *c* (*c* is the interface number)	Mc	Fixed Container: On the Commander platform this will be *[CDM v20\MQTT v20]* On the ObSys platform this will be *[OSM v20\MQTT v20]*

MQTT Setup

Object Type: *[OSM v20\MQTT v20]*
Object Type: *[CDM v20\MQTT v20]*

The MQTT driver contains the following objects.

Description	Reference	Type
Enable MQTT Enable the connection to the MQTT broker	E	Obj\NoYes; Adjustable
MQTT Broker connected Indicates the driver has connected and authenticated with the broker	DS	Obj\NoYes
MQTT Broker How to connect to the broker	N	Fixed Container: On the Commander platform this will be *[CDM v20\MQTT v20\Broker]* On the ObSys platform this will be *[OSM v20\MQTT v20\Broker]*
Database Control what values are available from Essential Data	D	Fixed Container: On the Commander platform this will be *[CDM v20\MQTT v10\Data]* On the ObSys platform this will be *[OSM v20\MQTT v10\Data]*
Publish/Subscribe Topic name rate for values from the database to be sent to the broker	PV	Fixed Container: On the Commander platform this will be *[CDM v20\MQTT v10\Publish]* On the ObSys platform this will be *[OSM v20\MQTT v10\Publish]*
Advanced Settings Additional settings	AS	Fixed Container: On the Commander platform this will be *[CDM v20\MQTT v10\Advanced]* On the ObSys platform this will be *[OSM v20\MQTT v10\Advanced]*
Debug Enable This will store additional debug information in the record file. Use this option only when instructed by North Support	DE	Obj\NoYes; Adjustable

MQTT Broker

Object Type: *[OSM v20\MQTT v20\Broker]*
Object Type: *[CDM v20\MQTT v20\Broker]*

The MQTT Broker contains the following objects required to connect to the server.

Description	Reference	Type
Host name or IP address Host name or IP address of MQTT broker. Set to 'reset', to load default driver settings (see also *Appendix A*)	IA	Obj\Text; Max 125 chars; Adjustable
Client ID The client identifier is unique and identifies North device to the server. Refer to broker documentation for format required	CID	Obj\Text; Max 125 chars; Adjustable
User ID User name to authenticate with broker	AID	Obj\Text; Max 125 chars; Adjustable
Password Password to authenticate with broker	APW	Obj\Text; Max 125 chars; Adjustable
Keep Alive (secs) Maximum time between messages before the driver should send a keep alive message	KA	Obj\Num: 15…6000; Adjustable Default: 60 seconds.
Connection State Displays current connection state between driver and the broker	CS	Obj\ENum: 0...3 Values: 0=Offline, 1=Connecting to broker, 2=Online, 3=Disconnecting
Connect Last Response Result of the last connection to the broker	CR	Obj\Text; Max 30 chars
Disconnect Last Reason Reason for last disconnection from broker	DR	Obj\Text; Max 30 chars
Online Since Date and time connection established with broker	CT	Obj\DateTime
Online Count Number of successful connections to broker	CC	Obj\Num

Database

Object Type: *[OSM v20\MQTT v20\Data]*
Object Type: *[CDM v20\MQTT v20\Data]*

Database contains the following objects. By default, all values from Essential Data are available to the MQTT driver. The Filter by Page (F) and Privilege Levels (P) objects provide finer control of which values are available.

Description	Reference	Type
Values Available The number of values available to MQTT with any filter applied	C	Obj\Num
Database Source Available on ObSys platform only	S	Obj\Enum; Adjustable 0: Essential Data, 1: Data Manager
Filter by Page Restrict access to only the Essential Data pages specified	F	Fixed Container: On the Commander platform this will be *[CDM v20\MQTT v20\Filter]* On the ObSys platform this will be *[OSM v20\MQTT v20\Filter]*
Read Privilege Levels Configure privilege levels to control read access to Essential Data	RP	Fixed Container: On the Commander platform this will be *[CDM v20\MQTT v20\Security]* On the ObSys platform this will be *[OSM v20\MQTT v20\Security]*
Write Privilege Levels Configure privilege levels to control write access to Essential Data	WP	Fixed Container: On the Commander platform this will be *[CDM v20\MQTT v20\Security]* On the ObSys platform this will be *[OSM v20\MQTT v20\Security]*

Filter by Page

Object Type: *[OSM v20\MQTT v20\Filter]*
Object Type: *[CDM v20\MQTT v20\Filter]*

Filter by Page limits the data available to the MQTT driver. Specify a start and end page from Essential Data, and only values from these pages will be sent to the MQTT broker.

Description	Reference	Type
Start Page First page of Essential Data objects to include. Set to '0' to remove the filter.	SP	Obj\Num: 0…128; Adjustable
End page (inclusive) Last page of Essential Data objects to include.	EP	Obj\Num: 0…128; Adjustable
Active filter Information on the start and end page-objects	I	Obj\Text

Database Privilege Levels

Object Type: *[CDM v20\MQTT v20\Security]*
Object Type: *[OSM v20\MQTT v20\Security]*

Security Areas and Levels

Within the North security model, there are eight security areas. Security areas could be actual areas in a building, but are normally functional areas – for example, 'environmental control' and 'North engineering' areas would allow a user to have different privileges in controlling set points and engineering Commanders.

Typically, a user is assigned a privilege level in each of the eight areas. The level is in the range zero to seven, seven being the most powerful. When a user wishes to pass a door, his/her privilege level in the door's area is checked against the minimum required for that area – and then either allowed to pass, or rejected.

The engineer must decide the use of the eight areas. The engineer must also decide the power of the privilege levels. Most systems use only a few levels per area: 0=None, 1=Guest, 2=User, 7=Administrator.

As an example, imagine a page of values in Essential Data. The page needs a user to have a minimum privilege level of 2 in area 1 before it can be viewed. The page is available in a Web browser that checks users with a security database. User A has privilege level 7 in area 1 – she can view the page. User B has privilege level 5 in area 1 – he can also view the page. User C has privilege level 1 in area 1 – she cannot view the page.

The example continues: within this page of values in Essential Data is a temperature set point object. Users need a minimum privilege level of 6 in area 1 to adjust it – therefore User A can adjust the set point, but User B cannot.

Specifying Access Security

Essential Data and Extra Data have Access Security objects to control who can view a page, and who can adjust an adjustable object.

Each Access Security object has a two-digit value. Each controls the access to a particular feature - such as viewing the page, or adjusting the value. The two-digit value is made up of the area digit (1-8), followed by the minimum privilege level (1-7) – for example, if the minimum privilege level is 6 in area 2, then the two digit value is 26. If the value is 00, then no security checks are made.

MQTT Driver

The Database Privilege Levels object contains a privilege level for each of the eight security areas, representing a virtual user. The MQTT driver uses these to control access to Essential Data when reading or writing a value.

Description	Reference	Type
Privilege Level in Area *x* The area, *x*, can be in the range 1…8	P*x*	Obj\Num; Adjustable; Range: 0…7

Publish/Subscribe

Object Type: *[OSM v20\MQTT v20\Publish]*
Object Type: *[CDM v20\MQTT v20\Publish]*

After the driver has connected to an MQTT broker, it can *publish* messages to send a value from the database, and *subscribe* to a topic and receive a value.

Each publish message contains a topic name, which will be used by the broker to forward the message to interested subscribers. A topic is a text string that can have levels of hierarchy, each separated by a slash ('/').

The driver publishes each value to the broker using a separate MQTT message. These messages can either all use the same topic name, such as 'North_MQTT/Building_5/HVAC', or each value can have a unique topic name by incorporating a variable, such as 'North_MQTT/Building_5/$(ol)'. A unique topic name for each value could be used alongside the Publish Retain Value (RV) option.

Publish Message contains the following objects.

Description	Reference	Type
Publish Topic Name The topic name identifying the publish message. Use the following variables to make the topic unique for each database value: $(oid) – Object identifier (1…1280) $(fl) – Full label $(pl) – Essential Data Page label $(ol) – Essential Data Object label. Space characters will be replaced with an underscore (_).	PT	Obj\Text: 125 chars; Adjustable
Publish Rate Frequency to publish values	PR	Obj\ENum; Adjustable 0: COV, 3: 15mins, 4: 30mins, 5:1hr
Publish Retain Value Indicates the broker should retain the last value using this topic name	RV	Obj\OffOn; Adjustable
Subscribe to Changes Subscribes to value changes made by other clients. These values are written to Essential Data. Available with MQTT v5 brokers only	SE	Obj\NoYes; Adjustble
Subscribe Topic Name Topic name to subscribe to. This may contain the wildcard '#'. E.g. 'North_MQTT/Building_5/#'	ST	Obj\Text: 125 chars; Adjustable

Advanced Settings

Object Type: *[OSM v20\MQTT v20\Advanced]*
Object Type: *[CDM v20\MQTT v20\Advanced]*

Advanced Settings contains the following objects.

Description	Reference	Type
MQTT Version	V	Obj\Enum; Adjustable 0: v3.1.1, 1: v5.0
TLS Connection Connect to the broker using TLS on port 8883. Available on ObSys platform only	TLS	Obj\NoYes; Adjustable
Prefix 'name' Specify a device identifier to prefix the JSON 'name' key	PN	Obj\Text; Adjustable
Authentication Method Available on ObSys platform only	AM	Obj\Enum; Adjustable 0: Basic, 1: Azure SAS token
Legacy Mode Send messages using for format of MQTT v1.0 driver	LM	Obj\OffOn; Adjustable

Appendix A: Example MQTT Brokers

Several MQTT brokers are available. The following public brokers are useful for testing your IoT application.

Links last checked in 2024.

HiveMQ

HiveMQ is an enterprise MQTT broker, available both cloud hosted and on-premises.

Website: *www.mqtt-dashboard.com* or *www.hivemq.com/mqtt/public-mqtt-broker*

💻 To configure the MQTT driver for HiveMQ, follow these steps:

- → Navigate to **MQTT Setup, MQTT Broker** and set **Host name** to 'hivemq'. This will auto-configure the driver to connect to broker.hivemq.com
- → Next, in a web browser, navigate to the *websocket client* on the HiveMQ website
- → Using the default connection settings, click connect. Once connected, click Add New Topic Subscription, and set Topic to 'North_MQTT/#' and click Subscribe

  The '#' is a wildcard, and you should now see all messages with the topic name starting 'North_MQTT/'
- → Update a value in Essential Data, and this should be sent to the broker.

Azure IoT Hub

Available on ObSys platform only.

Azure IoT Hub is part of the Microsoft Azure cloud hosted platform.

IoT Hub supports a limited feature MQTT connection.

💻 To configure the MQTT driver for Azure IoT Hub, follow these steps:

- → From Microsoft Azure, create an IoT Hub resource then add a device
- → Copy the Primary connection string for the device on Azure
- → Navigate to **MQTT Setup, MQTT Broker** and edit **Host name**. Paste the Primary connection string to auto-configure the driver to connect to Azure IoT Hub. Alternatively, set **Host name** to 'iothub' and complete the information manually
- → To monitor publish messages, download the Azure IoT Explorer, add the IoT Hub, then monitor the device's Telemetry data
- → Update a value in Essential Data, and this should be sent to the broker.

Driver Versions

Version	Build Date	Details
1.0	9/08/2016	Driver released
2.0	08/01/2024	Added support for MQTT version 5. Removed support for QoS level 2. Removed ALARM object. JSON payload, objects renamed (‘updated’, ‘label’, ‘isUnreliable’, ‘isOutRange’) Added support for SUBSCRIBE Added TLS support for ObSys platforms only

Next Steps…

If you require help, contact support on 01273 694422 or visit *www.northbt.com/support*

North Building Technologies Ltd
+44 (0) 1273 694422
support@northbt.com
www.northbt.com

Author: JF
Checked by: AB

Document issued 10/01/2024.