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 M2M application that subscribes to them. MQTT message brokers are available as part of many IoT platforms, both cloud hosted and on-premises. Available for Commander and ObSys.

This document relates to MQTT driver version 1.0

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 broker. The broker stores these values and delivers them to any M2M application that subscribes to the data feed. MQTT message brokers are available as part of many IoT platforms, both cloud hosted and on-premises.

Message Queuing Telemetry Transport (MQTT) version 3.1.1 is an international and OASIS standard – ISO/IEC 20922.

MQTT uses a publish-subscribe model. The MQTT driver publishes values, as they change, from Essential Data and Extra Data within the North device using a topic name. Integrate data collected using North interface technology directly into your own application by subscribing to these topics.

The driver connects to an IP network, with access to a 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 and Extra Data. As a value updates, these updated values are published in near real-time to an MQTT broker in JSON format.

Essential Data contains 640 values on Commander, and 1280 values on ObSys. If necessary, start the Extra Data interface (which requires an interface licence) for an additional 1024 values. Access to these values can be controlled by configuring a page range filter or privilege levels within the driver.

The MQTT driver can also route North-format alarms to the broker.

Prerequisites

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

The Essential Data module should be configured. The MQTT driver requires Essential Data v3.0 (build 01/09/2015) or later.

If the MQTT broker is cloud-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 IBM Bluemix, 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.

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

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.

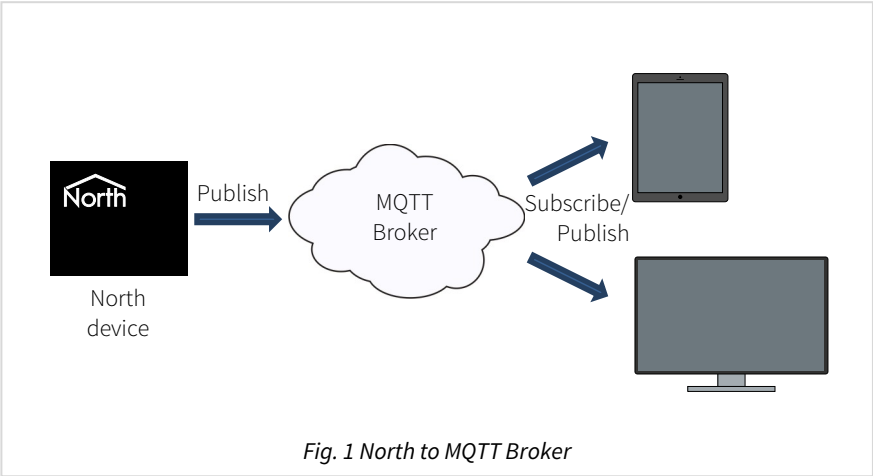

Fig. 1 North to MQTT Broker

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.

Topic Name

When the driver publishes a message to the broker, it includes a topic 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 the topic.

A topic is a text string that can have levels of hierarchy, each separated by a slash. A sample topic for sending a room temperature could be 'North_MQTT/Essential_Values/Living_Room/Temperature'. A client could subscribe to this exact topic, or use a wildcard for a group of topics.

Topic names for both value and alarm events can be set. By default, the driver publishes values from Essential Data using the topic name '*system label/essential data label/page label/object label*'.

You can set the topic name from the driver's *'Publish Value'* object.

The following variables may be included in the topic name:

- North device's serial number
- Driver's system label
- Essential/Extra Data label
- Object identifier
- Combined page and object label
- Page number or label
- Page-object number or label
- Object type
- Alarm system label
- Alarm point label

The driver will replace a space character with an underscore (_).

Security

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

The driver supports non-secure MQTT only, TLS connections are not supported (see *Appendix B*).

Publish Value

The MQTT driver sends a publish message with a single value as they change from Essential Data and Extra Data within the North device.

Message Content

```
{
    "oid": number,
    "type": string,
    "value": various,
    "updated": string,
    "isUnreliable": boolean,
    "isOutRange": boolean,
    "label": string
}
```

Parameters

Parameter	Value	Description
oid	Number	Object identifier is in the range 1…1280 for Essential Data (depending on platform), and 2001…3024 for Extra Data
type	String	Type of object, one of the following: text, noyes, offon, num, enum, float, datetime, date, times, or profile
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-ddThh: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
updated	String	Date and time value last updated (local time) in ISO-8601 format, '*yyyy-mm-ddThh:mm:ss*'
isUnreliable	Boolean	If 'true', communications with the sub-system are failing
isOutRange	Boolean	If 'true', value is outside min/max limits
units	String	Optional units for object value (max. 8 chars)
label	String	Combined label of page and object (max. 43 chars)
enum	Array	Optional array of objects, each containing an index (i) and label (l) for each value. Only provided with enum types
i	Number	Index value of enumeration. Only provided with enum types.
l	String	Label for enumeration. Only provided with enum types.

Example

```
{
   "oid": 17,
```

```
   "type": "float",
   "value": 12.3,
   "updated": "2013-12-31T23:40:16",
   "isUnreliable": false,
   "isOutRange": false,
   "units": "W",
   "label": "UPS Status - Load power"
}
```

Publish Alarm

Route North-format alarm messages to the driver, these will then be published to the broker.

Message Content

```
{
    "system": string,
    "point": string,
    "condition": string,
    "priority": number,
    "occurred": string
}
```

Parameters

Parameter	Value	Description
system	String	System label of the device in alarm
point	String	Label of the point in alarm
condition	String	Alarm condition
priority	Number	Priority of alarm, 1 (highest) … 9 (lowest)
occurred	String	Date and time alarm occurred in ISO-8601 format, '*yyyy-mm-dd*T*hh:mm:ss*'

Example

```
{
   "system": "Zip System",
   "point": "Boiler Room Door Contact",
   "condition": "Opened",
   "priority": 2,
   "occurred": "2013-12-31T23:40:16"
}
```

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 (Mc). 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

→ If your MQTT broker requires a particular topic name, navigate to **Publish Value** and set the **Topic name**.

Checking Communications

After configuring Essential Data or Extra Data, the driver will automatically 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) and **Last connect response** (CR) 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)	M*c*	Fixed Container: On the Commander platform this will be *[CDM v20\MQTT v10]* On the ObSys platform this will be *[OSM v20\MQTT v10]*
North MQTT Route alarm events to the MQTT broker	S*c*	Fixed Container: *[MQTT v10]*

MQTT Setup

Object Type: [OSM v20\MQTT v10]
Object Type: [CDM v20\MQTT v10]

The MQTT driver contains the following objects.

Description	Reference	Type
System label Label displayed when scanning the system, and can be included in topic name	DL	Obj\Text; Max 20chars; Adjustable
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 v10\Broker]* On the ObSys platform this will be *[OSM v20\MQTT v10\Broker]*
Database Control what data is available from Essential Data and Extra 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 Value Topic name and QoS for values from the database 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]*
Publish Alarm Topic name and QoS for alarm events sent to the broker	PA	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 Set the message timeout, message delay, and disable JSON content	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 v10\Broker]
Object Type: [CDM v20\MQTT v10\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 the driver to the server. Refer to broker documentation for format required. The following variables can be used: $(sn) – serial number of North device $(label) – System label	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. Default 60 seconds.	KA	Obj\Num: 15…6000; Adjustable
Connection state Displays drivers current connection state with the broker	CS	Obj\ENum: 0...3 Values: 0=Offline, 1=Connecting to broker, 2=Online, 3=Disconnecting
Last connect response Returns the last response from the broker	CR	Obj\Text; Max 40 chars

Database

Object Type: [OSM v20\MQTT v10\Network]
Object Type: [CDM v20\MQTT v10\Network]

Database contains the following objects. By default, all values from Essential Data and Extra 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
Objects Available The number of objects available from Essential Data and Extra Data. When a filter is used, this indicates the number objects available from Essential Data	C	Obj\Num: 0…2304
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 v10\Filter]* On the ObSys platform this will be *[OSM v20\MQTT v10\Filter]*
Database Privilege Levels Configure privilege levels to control read access to Essential Data and Extra Data	P	Fixed Container: On the Commander platform this will be *[CDM v20\MQTT v10\Security]* On the ObSys platform this will be *[OSM v20\MQTT v10\Security]*

Filter by Page

Object Type: [OSM v20\MQTT v10\Filter]
Object Type: [CDM v20\MQTT v10\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 v10\Security]*
Object Type: *[OSM v20\MQTT v10\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 and Extra Data when reading 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 Message

Object Type: [OSM v20\MQTT v10\Publish]
Object Type: [CDM v20\MQTT v10\Publish]

After the driver has connected to an MQTT broker, it can publish messages. Publish Message is available for the objects Publish Value, to send a value from the database, and Publish Alarm, to send an alarm event.

Each publish message contains a topic, 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. A sample topic for sending a room temperature could be 'North_MQTT/Essential_Values/Living_ Room/Temperature', or for sending an alarm event could be 'North_MQTT/alarm/Boiler_Control'. Use $() variables to substitute information about the value or alarm.

A quality of service (QoS) level is also required. The level (0, 1, or 2) specifies the guarantee of a publish message reaching the broker.

- QoS 0 (At most once) – a message is not acknowledged by the receiver, no guarantee or delivery is provided
- QoS 1 (At least once) – a message is acknowledged by the receiver, guaranteeing it to be delivered at least once
- QoS 2 (Exactly once) – a message is acknowledged twice by the receiver, guaranteeing it to be delivered only once. An additional overhead is required to send QoS2 messages.

Publish Message contains the following objects.

Description	Reference	Type
Topic name The topic name identifying the publish message. Use the following variables to make the topic unique for each database value or alarm: $(sn) – serial number of North device $(label) – System label For Essential Data and Extra Data values: $(el) – Essential/Extra Data label $(pl) – Page label $(ol) – Object label $(fl) – Page and Object label $(oid) – Object identifier $(pn) – Page number $(on) – Object number For Alarms: $(as) – Alarm system label $(ap) – Alarm point label The driver will replace a space character with an underscore (_).	TN	Obj\Text: 125 chars; Adjustable Default for publish value: '$(label)/$(el)/$(pl)/$(ol)' Default for publish alarm: '$(label)/alarm/$(as)'
QoS level Quality of service level required	QOS	Obj\Num: 0…2; Adjustable
Broker retain last value Indicates the MQTT broker should store the last value sent for a topic	RV	Obj\NoYes; Adjustable

Advanced Settings

Object Type: [OSM v20\MQTT v10\Advanced]
Object Type: [CDM v20\MQTT v10\Advanced]

Advanced Settings contains the following objects.

Description	Reference	Type
Message Timeout (ms) Maximum time to wait for a response from the remote server. Default 3000 milliseconds.	TO	Obj\Num: 10…30000; Adjustable
Delay between messages (ms) Block time between completing sending one message end sending the next. Default 50 milliseconds.	BT	Obj\Num: 10…6000; Adjustable
JSON Content Set which parameters are included with a publish value message	C	Fixed Container: On the Commander platform this will be *[CDM v20\MQTT v10\Content]* On the ObSys platform this will be *[OSM v20\MQTT v10\Content]*

JSON Content

Object Type: [OSM v20\MQTT v10\Content]
Object Type: [CDM v20\MQTT v10\Content]

JSON Content contains the following objects to select what content is included in a publish value message. Refer to *Publish Value* for a description of the parameters.

Exclude unrequired parameters to reduce the publish value message size. This could be useful if the MQTT broker charges based on the amount of data sent.

Description	Reference	Type
Include value Object's value	T0	Obj\NoYes; Adjustable
Include label Combined page & object label	T1	Obj\NoYes; Adjustable
Include oid Object's identifier	T2	Obj\NoYes; Adjustable
Include updated Date & time value last updated	T3	Obj\NoYes; Adjustable
Include is Unreliable Indicates if the communications with the sub-system are failing	T4	Obj\NoYes; Adjustable
Include is OutRange Indicates if value is outside max/min limits	T5	Obj\NoYes; Adjustable
Include units Optional units for object value	T6	Obj\NoYes; Adjustable
Include type Type of object	T7	Obj\NoYes; Adjustable
Include enum For enum types, a list of value label pairs	T8	Obj\NoYes; Adjustable

North MQTT

Object Type: [MQTT v10]

The North MQTT system contains a single object to send North-format alarm events.

Configure the Publish Alarm (PA) driver object with the topic name and QoS to send the message to the MQTT broker. Refer to *Publish Alarm* for more details of the message sent.

Description	Reference	Type
Alarm Route North-format alarm events to this object and they will be published to the MQTT broker	ALARM	Obj\Alarm; Adjustable

Appendix A: Example MQTT Brokers

Several MQTT brokers are available. The following public brokers are useful for testing your IoT application, so you can quickly build a proof of concept.

Links last checked September 2016.

HiveMQ

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

Website: *www.hivemq.com/try-out/*

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

→ Navigate to **Configuration, Essential Data**. If a page-object has not already been configured, create one with a float value (maybe from a thermistor with the Zip M7004, provided in the training pack)

→ 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/#'. Finally, 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 passed to the broker.

IBM Watson IoT

The IBM Watson IoT Platform, part of IBM Bluemix, is a cloud platform for building analytics applications, visualization dashboards, and mobile IoT apps. MQTT is the primary mechanism that devices use to communicate with the Watson IoT Platform.

Website: *internetofthings.ibmcloud.com*

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

→ Navigate to **Configuration, Essential Data**. If a page-object has not already been configured, create one with a float value (maybe from a thermistor with the Zip M7004, provided in the training pack)

→ Navigate to **MQTT Setup, MQTT Broker** and set **Host name** to 'ibm'. This will auto-configure the driver to connect to Watson IoT Platform's quickstart.

Next, in a web browser, navigate to *quickstart* on the IBM Watson IoT Platform website.

Accept the terms of use, and enter the serial number of your North device – only enter the first eight digits. (You will find the serial number by navigating to Configuration, Interfaces, Interface Licences).

The webpage will update when the driver has connected. You should see a table of events along with columns for datapoint and value.

Appendix B: TLS Connections

The driver only supports non-secure MQTT connections to port 1883. The driver does not support secure TLS connections to port 8883. If a TLS connection is required, to connect with AWS or Microsoft Azure cloud platforms, then a separate proxy will be required.

North does not provide support for the configuration or use of any proxy. The information given below is provided “as is”, without warranty of any kind.

As an example, Stunnel proxy software (available from *www.stunnel.org*) could be installed on a PC on the local network. Configure the proxy to accept inbound TCP connections on port 1883, and connect to the Internet MQTT broker via TLS.

An example Stunnel service for an MQTT broker may look something like:

```
[mqtt]
client = yes
accept = :1883
connect = contoso.azure-devices.net:8883
verify = 2
CAfile = ca-certs.pem
checkHost = azure-devices.net
sslversion = TLSv1.2
ciphers = ECDHE-RSA-AES256-GCM-SHA384:AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-
SHA256:AES128-GCM-SHA256
```

Configure the MQTT driver to connect to the IP address of the local proxy, rather than real host name.

Driver Versions

Version	Build Date	Details
1.0	9/08/2016	Driver released

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: BS

Document issued 07/12/2016.