kamstrup

Kamstrup OpenlinkIQ® Concentrator

IoT Connectivity Guide

Contents

Terms and Abbreviations

Abbreviation	Description
MQTT	IoT Protocol – described in www.mqtt.org
SSL	Secure Socket Layer
TLS	Transport Layer Security
RF	Radio Frequency. Term used to refer to the radio media.
OpenlinkIQ®	Open LPWAN RF protocol. Specifications (like this document) is found at openlinkiq.org.
SSH	Secure Shell (ssh.com)

Introduction

The purpose of this document is to describe how the Kamstrup OpenlinkIQ® Concentrator process and delivers end-device data to a connected system.

The document both serves as a guide to the Kamstrup OpenlinkIQ® Concentrator as well as a guide and inspiration to developers that wish to design an OpenlinkIQ® receiver/concentrator or head-end system.

This guide assumes that the reader is familiar with the OpenlinkIQ® RF specification. It is not a pre-requisite for following this guide, but a few references are made to the specification without further explanation.

It is further more assumed that the reader is familiar with the MQTT protocol. A very short general introduction to MQTT is made in the following section – investigation into further details is left for the reader to perform.

Protocol

The Kamstrup OpenlinkIQ® Concentrator makes use of the MQTT protocol to publish received OpenlinkIQ® RF frames.

The MQTT protocol defines a light weight publish-subscribe architecture used by a vast number of IoT devices. It is a traditional client/server protocol where clients are publishers and/or subscribers and the server is a simple broker, distributing published data to subscribers.

Clients that wish to receive data published to a certain topic, makes a subscription to this topic.

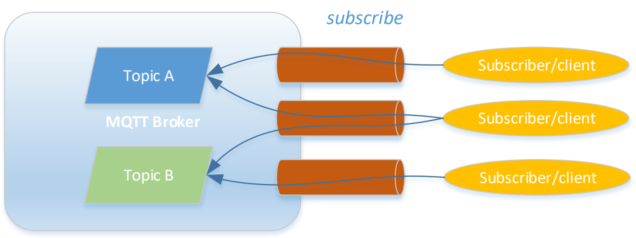


Figure 1. Subscribe to topic.

Data published to the topic will hereafter be propagated to all subscribers of this topic.

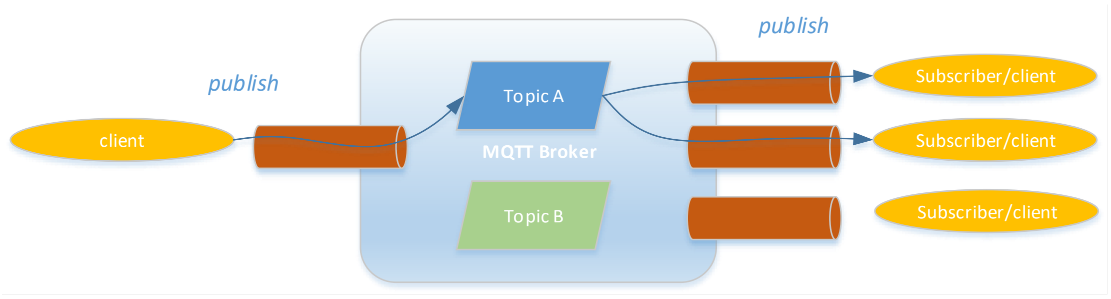


Figure 2. Publish to a topic and propagation.

The Kamstrup OpenlinkIQ® Concentrator runs MQTT over IP as illustrated below.

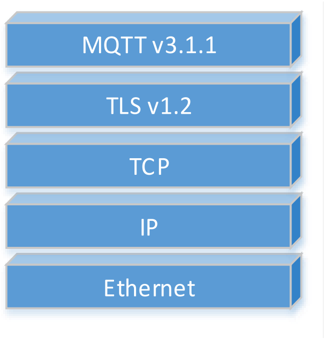


Figure 3. Protocol stack.

The Kamstrup OpenlinkIQ® Concentrator requires adding security on the transport level using TLS version 1.2.

Connecting to broker, publishing data, disconnecting follows the MQTT standard sequence. Note that clients does not take a specific role (publisher/subscriber) – all clients can be both publishers and subscribers (in fact a client often will be both).

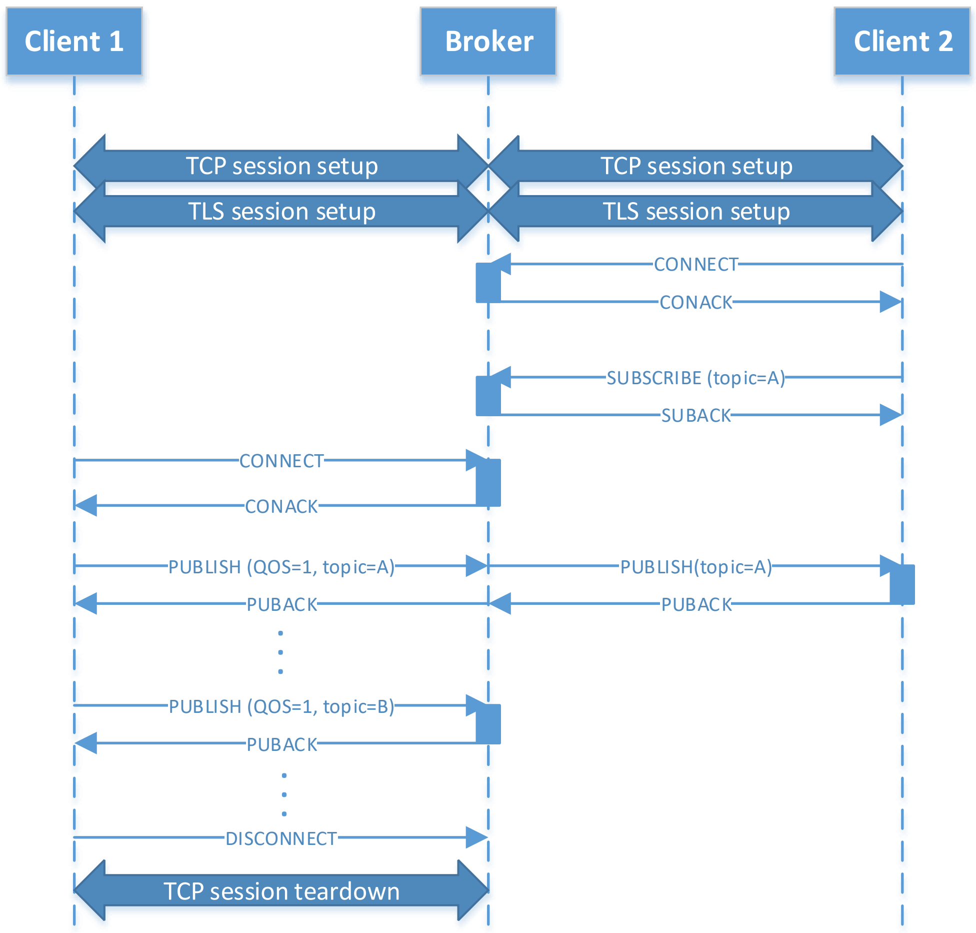


Figure 4. MQTT message flow.

After establishing a TCP session, the TLS session is setup. Hereafter follows the MQTT message sequence started by connecting clients to the broker. In Figure 4 the Kamstrup OpenlinkIQ® Concentrator will act as the depicted client number 1. Client subscribing to topic in Figure 4 can connect securely or not by choice (secure connection is recommended). In this example a secure connection is depicted.
Sections below describes in more detail what features and parameters are used by the Kamstrup OpenlinkIQ® Concentrator MQTT client.

Note that the MQTT session is maintained in the lifetime of the client – same connection is used for all publish-operations until terminated (by the client, network disturbances etc.).

Note also that MQTT connections can come and go within the same TCP session.

In the design of the MQTT client of the Kamstrup OpenlinkIQ® Concentrator, several choices of MQTT parameters and settings has been made. These are all described in the following sections.

MQTT Connect

The Kamstrup OpenlinkIQ® Concentrator initiates a connection to the broker whenever data for transfer is available.
In case the connection is lost, reconnect is only performed when new data is to be transferred.

The following MQTT connect parameters are used and supported:

- Only MQTT version 3.1.1 is supported.
- Clean session = true
  Broker is not expected to persist messages from the Concentrator.
- Keep Alive = 60 seconds
  Broker will abort session if no keep alive message is received every max 60 seconds.

See section on configuring the broker URL.

Lost MQTT connection

In case the MQTT connection is lost, the Kamstrup OpenlinkIQ® Concentrator MQTT client will re-connect when new RF data is available for transfer.

If establishing a connection to the MQTT broker turns out to be difficult, an exponential backoff mechanism between retries is implemented. The backoff delays between retries are determined as follows.

2*n seconds + random(0-999) ms, where n=0-6.
Beyond the 7th retry (n=6), waiting time is calculated with n=6.

Publish end-device data

When publishing data the Kamstrup OpenlinkIQ® Concentrator makes use of the following MQTT parameters.

- QoS = 1.
  Concentrator assumes that when the broker returns a PUBACK, data has been received and processed with success. When receiving PUBACK the concentrator will delete relevant cached data.
- Retained = false
  The message is not to be saved by the broker as the last known good value for the topic. If re-tain=true a new subscriber to the topic would receive the last message retained on the topic right away.
- Duplicate flag = false
  Indicates that the message is a duplicate and was resent because the broker did not acknowledge the original message.

The end-device data is embedded in the payload of the publish message and is JSON encoded with the following format.

```
{ “version”: 1,
  “frames”:   [
    { “frametype”: “OpenlinkIQ”,
      “timestamp”: <time of reception (“2021-11-13T21:54:01Z”)>,
      “frame”: <MBAL frame (“...”)>,
      “meta”:
        {“rssi”: <int>,
         “noofcorrectedbits”: <int>
        }
    },
    { additional frames as shown above }
]}
```

The “**version**” property is set to 1. Future changes to the embedded format will be tracked with this key.

The “**frames**” property consists of a list of received OpenlinkIQ® RF frames, identified by the following fields.

Field	**Description**	**Type**
frameType	The type/protocol of the frame in the ‘frame’ field.	String
Frame	The MBAL frame (see OpenlinkIQ® Specification).	Base64-encoded string
Timestamp	Time of reception.	String
Meta	Structure of data concerning the RF reception.	JSON object
meta.rssi	The signal strength at which the frame was received.	Integer
meta.noofcorrectedbits	The number of bits corrected at the reception of the frame.	Integer

Topics

When designing and configuring a MQTT setup, it is important to consider the hierarchy of data publishing **topics**. This enables a simple method of identifying key data elements and performing efficient filtering on subscribed data.
The Kamstrup OpenlinkIQ® Concentrator makes no assumptions on how the topics are defined, and let’s the operator enter any topic string that fulfils the MQTT specification on topic.

Prioritized data

The OpenlinkIQ® RF specification defines the concept of frames with high priority (defined in the control field of the RF frame). The Kamstrup OpenlinkIQ® Concentrator will detect whether a received frame is of high priority and leaves the option of delivering this frame to a separate MQTT topic.
This is further described in the section on configuration.

Subscribe to topics

The Kamstrup OpenlinkIQ® Concentrator does not subscribe to any topics published by the broker.

Disconnect – Last Will

When connecting the client to the broker a "Last Will" option can be applied. This enables subscribers to be notified if the Concentrator disconnects ungracefully (unintentionally).

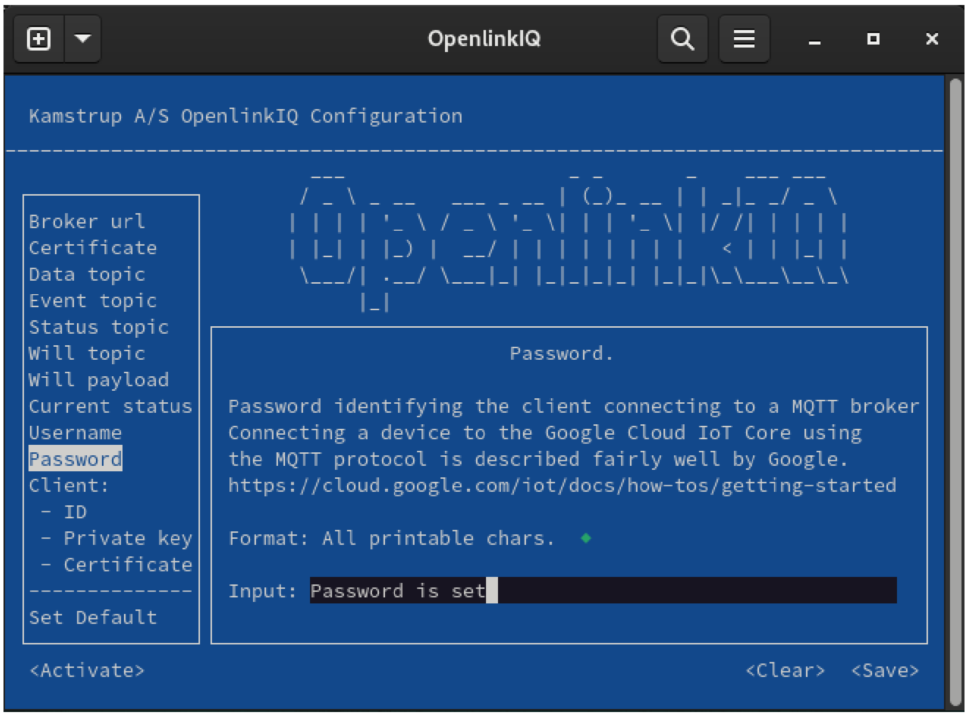
By defining a "Last Will" topic this option is included when connecting to a broker.

Last Will message is requested to be published with

- QoS = 1
- Retain = false

Configuration

The Kamstrup OpenlinkIQ® Concentrator offers a number of configuration options, several of which has been mentioned in previous sections. Each options is describes in this section.



```
OpenlinkIQ

Kamstrup A/S OpenlinkIQ Configuration
--------------------------------------------------------------------------------
                    ___                   _ _       _    ___ ___
                   / _ \ _ __   ___ _ __ | (_)_ __ | | _|_ _/ _ \
 Broker url       | | | | '_ \ / _ \ '_ \| | | '_ \| |/ /| | | | |
 Certificate      | |_| | |_) |  __/ | | | | | | | |   < | | |_| |
 Data topic        \___/| .__/ \___|_| |_|_|_|_| |_|_|\_\___\__\_\
 Event topic                |_|
 Status topic
 Will topic                          Password.
 Will payload
 Current status   Password identifying the client connecting to a MQTT broker
 Username         Connecting a device to the Google Cloud IoT Core using
 Password         the MQTT protocol is described fairly well by Google.
 Client:          https://cloud.google.com/iot/docs/how-tos/getting-started
  - ID
  - Private key   Format: All printable chars.
  - Certificate
 --------------   Input: Password is set
 Set Default

 <Activate>                                                  <Clear>   <Save>
```

Figure 5. Configuration Interface of the Kamstrup OpenlinkIQ® Concentrator.

The above interface is displayed when connecting via SSH to the Kamstrup OpenlinkIQ Concentrator as the user "tech" (password is printed on a label mounted on the case of the unit).

The configuration interface is very basic, listing all possible parameters on the left-hand side. Move down the list using the up/down keys. A short description and a field for entering a value is shown on the right-hand side.

The button line includes a option for clearing the current value in the input field (*<Clear>*). Saving a entered value is done by pressing the *<Save>* button.
When all a complete configuration is entered, it can be activated by pressing the *<Activate>* button.

Navigating between *<Activate>*, *<Clear>*, *<Save>* and the list of options, is done using the TAB key.

Broker URL

The URL defines the address of MQTT broker to connect to, the transport layer to use and the TCP port to connect to.

Format: ssl://<broker address (FQDN)>:<tcp port>

TLS (version 1.2) is used to establish a secure connection, which is indicated by “ssl”. No other options are supported.

The address of the broker server must be a fully qualified domain name.

An example: A secure connection to the EMQX Cloud MQTT broker would be obtained via the URL “*ssl://broker.emqx.io:8883*”

Topic for End-device Data

The Kamstrup OpenlinkIQ® Concentrator can publish end-device data to two different topics.

- one for OpenlinkIQ® frames marked as being of priority (event) and
- another topic for non-priority frames (data).

General rules defined for MQTT topics apply for the syntax of the topics.

Example of a MQTT topics:

```
/device/KAM73471100/data/event
/device/KAM73471100/data/endpointdata
```

Topic for Last Will

Define the topic on which the Last Will message of the Concentrator must be published. By defining this topic the Last Will option is included in the MQTT connect operation.

Example: `/device/KAM73471100/will`

Payload for Last Will

Payload can be added to the “Last Will” message in order to add details to the notification. This configuration option defines the payload. An example could be *{ “connected”: 0 }*.

Topic for Status Notifications

Whenever the MQTT client of the Kamstrup OpenlinkIQ® Concentrator connects successfully to the MQTT broker, the client publishes the payload “*{ “connected”: 1 }*” to the Status Notification topic.

Example: `/device/KAM73471100/status`

MQTT Broker authentication / Certificate

Certificate to verifying the authenticity of the MQTT Broker pointed to by the broker URL.
PEM format is required.

The certificate is pasted into the input-field of the configuration interface.

MQTT Client Username

Username included in the MQTT Connect message to identify and authenticate the client.

MQTT Client Password

Password included in the MQTT Connect message to identify and authenticate the client.

MQTT Client ID

The MQTT client is identified by the client-Id in the Connect message.

MQTT Client Certificate

In case the MQTT broker requires authentication of the client, the client certificate (PEM format) can be entered (pasted into the input-field of the configuration interface).

MQTT Client Private Key

In cast the MQTT broker requires authentication of the client, the client certificate is transferred during TLS connection. The private key (PEM formatted) matching the public key in the certificate can be entered.

Receiving OpenlinkIQ® frames

Whenever an OpenlinkIQ® frame is received, it is attempted to be forwarded to the connected system/broker right away. If this is not possible, due to lack of network connection or similar reasons, frames are cached.

When a connection to the MQTT broker is established, all cached frames are published. This done in iterations of 500 frames. Within each batch of 500 frames, all frames that are marked as "prioritized" are published first, followed by all non-priority frames.

Appendix A

The Kamstrup OpenlinkIQ® Concentrator includes sufficient configuration options to connect to most MQTT enabled systems. Below is a short list of systems that might be source of inspiration to selecting a data collection system or even how construct one of your own.

Connecting to AWS

The AWS IoT Core.

Guide on how to connect.

ThingsBoard

Setting up account and devices.

Connecting using the MQTT protocol.

Plain MQTT brokers

Several MQTT brokers exist for public use. These should only be used for test purposes.

Examples: EMQX, Eclipse

Or spin up your own (mosquitto is a good example).