WHITE PAPER

Device integration with MQTT

March 2022

AXIS COMMUNICATIONS

Summary

MQTT is a standard messaging protocol that facilitates efficient and reliable exchange of data between IoT devices and cloud applications. It allows devices (through their MQTT clients) to publish messages to a common MQTT broker (server) that mediates communication with other devices. The broker keeps track of who is publishing what and who wants to see the data, forwarding messages to only the clients that subscribe to the right topic.

In a typical VMS ecosystem, Axis event notifications from devices are traditionally streamed to a single destination via VAPIX/ONVIF API interface using the RTSP streaming protocol. But the same notifications can be distributed using the MQTT protocol via the device's built-in MQTT client (applicable for devices running AXIS OS 9.80 or a later version). This is possible both within VMS ecosystems and outside of them, and is particularly useful over the internet. Multiple subscribed MQTT clients on the network can then use and process the event notifications published by the Axis device. There are also Axis and third-party ACAP analytics applications having their own MQTT clients designed for specific systems, use cases, and subscribers.

As an example of a use case related to Axis products, people counting devices can send statistics over MQTT to data visualization software in the cloud. In another example, a third-party door sensor communicates over MQTT with a signaling device and a camera, which play an alarm and start a recording each time the door is opened.

Table of Contents

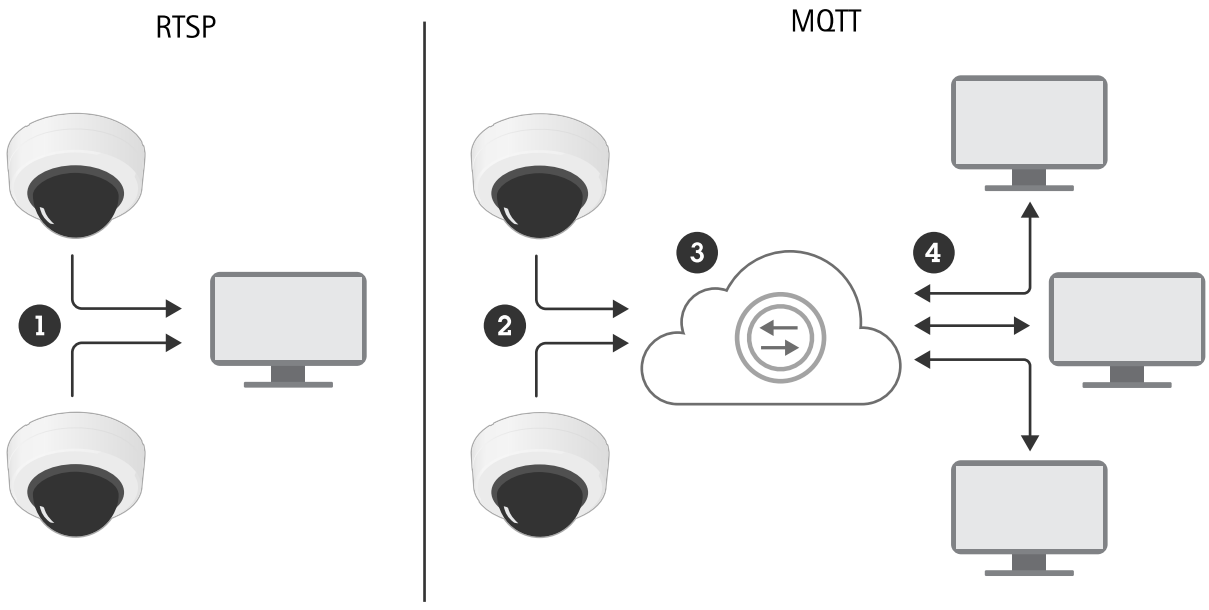
1 Introduction

MQTT (Message Queuing Telemetry Transport) is a standard messaging protocol for the internet of things (IoT). It was designed for simplified IoT integration and is used in a wide variety of industries to connect remote devices with a small code footprint and minimal network bandwidth. The MQTT client in AXIS OS can simplify integration of data and events produced in the device to systems that are not video management systems (VMS).

This white paper presents a technical background on MQTT including typical use cases, benefits, and limitations. It also provides details about MQTT clients in Axis devices and ACAP analytics apps.

2 MQTT protocol

MQTT is a publish/subscribe protocol. This means it has a different messaging pattern than RTSP or HTTP, which are request/response protocols. With RTSP, one side issues a request and the other side responds. Many mobile messaging apps are instead based on MQTT or similar publish/subscribe concepts. There are also publish/subscribe protocols that are optimized for closed or specific systems.



While RTSP allows only one-to-one communication, MQTT also enables one-to-many or many-to-many communication through the broker.

1 *Event stream*
2 *Publishing*
3 *MQTT broker*
4 *Subscribing*

The MQTT concept is that all clients connect to a common MQTT broker (server), which keeps track of who is publishing what and who wants to see the data. The connection is typically a TCP session on port 1883. A client may also connect over TLS (typically port 8883) or using WebSocket (typically port 1884/8884).

Clients publish messages with a topic. Another client may subscribe to that specific topic or use wildcards to get all subtopics. A message also includes a payload that is typically a JSON data structure, a string, or

even short binary data. The publisher does not know if other clients are subscribing. The broker will only forward messages to clients that have topic subscriptions.

With MQTT, it works a bit like sending an article to a magazine. People who subscribe to the magazine will be able to read the article and it can be either one-to-one or one-to-many communication (and MQTT even enables many-to-many communication). The article can also be read long after it was originally published.

In comparison, using RTSP would be more like making a phone call. There is one source and one target for your commands, and it is always one-to-one communication. If your target did not answer the phone, they missed the message.

When MQTT is used to distribute Axis event notifications from devices, multiple subscribed MQTT clients in the network can use and process the notifications. This is a great advantage compared with the traditional way (using VAPIX®/ONVIF® application programming interface (API) and RTSP), where the event notifications would instead be streamed to one single destination.

3 Benefits

There are several benefits to using MQTT. Compared with using a request/response protocol such as RTSP, the main benefits of MQTT include:

- **Reduced risk of device passwords being exposed.** There is no need for a client to access a device or server to get the data. This means that the client does not need to know the password nor know how the API works. This reduces the risk of device passwords being exposed to clients and users, thus reducing the risk of deliberate or accidental misuse.
- **Single point of integration.** If authorized, all clients can get all other clients' published messages with a single connection to a broker. In RTSP, a client needs to connect to each client that it wants data from. This means that the MQTT message flow may be one-to-one, one-to-many, or many-to-one without any extra burden for each client.
- **Publish and subscribe with an intact firewall** In RTSP, the device/server API needs to be accessible for the client. If the device is behind a firewall and the client is remote, the firewall needs to be configured to allow incoming requests, exposing the device API. With a public MQTT broker in between, clients behind a firewall can publish/subscribe specific data without poking a hole in the firewall (if the firewall allows outbound connections).
- **Quality of service.** When posting a critical message, a publisher can monitor if that message was received by another client and take alternative action if not.
- **Retained messages.** Publishers may mark a message as retained, meaning that the broker will keep a copy of the message and send that message to newly connected clients subscribing to that topic.
- **IoT client availability.** MQTT client packages are available for all common software development environments including Windows®, Linux®, Android™, iOS, Node.js®, PHP, and Python®. There are many more clients that can connect to a broker compared to setting up an RTSP data stream to a device.
- **Simplified message monitoring and debugging.** There are several MQTT tools that can be used to monitor all the published messages and also to publish message to troubleshoot if and how subscribers react.
- **Simplified data structure.** Because MQTT is often targeting unknown clients, the message payload will usually take that into consideration to simplify for the subscriber.

4 Limitations

There are some drawbacks to MQTT when compared with alternative protocols:

- **Single point of failure.** If the broker is unavailable all messaging stops working. However, the infrastructure may be designed with redundancy brokers.
- **Who published a message?** By design, MQTT focuses on the topic, and not on who published the message. Unless the publisher includes some ID as part of the topic or the payload, you would need to access the broker's log to know who published the message. It is common practice that publishers include some client ID in the topic or the payload based on what the use case is.
- **A malicious client** connected to the broker may publish/subscribe to any topic that it is authorized for. You need to protect the broker (see section on MQTT security).
- It is **not designed for continuous video/audio streaming.**

As with any server, the total bandwidth throughput needs to be considered. Dynamic scaling may be required for very large systems having many clients.

5 Infrastructure

It is fairly easy to set up a local Eclipse Mosquitto™ broker or enable Node-RED® to act as a local broker, such as Aedes. There are also a number of internet service providers and others providing managed MQTT brokers, like Microsoft® Azure® IoT, HiveMQ™, CloudMQTT, and IBM® Cloud®.

If a system has no remote clients, it is recommended to use a local broker. A local broker can also act as a proxy to a public broker or be configured to act as a proxy to selected local broker messages and the public broker messages.

6 Security

The broker needs appropriate protection based on how critical messages are and on which threats a specific system may experience. MQTT offers several different authentication schemes including no authentication, user/password, and TLS client certificate authentication. Different users may have different authorizations on which topic it can publish or subscribe to. The broker can allow clients to connect over unencrypted TCP or over encrypted TLS (like HTTPS).

- **No authentication.** A local broker may disable authentication if the messages are non-critical and the broker is not exposed for internet clients. It is recommended to use this only for testing, sandbox development, and demonstrations.
- **User/password.** This is the most common setup. Depending on what risks the system has, the system administrator may let all MQTT clients share the same user/password or create users with restricted topic access.
- **TLS client certificates.** For internet-exposed brokers where the messages are classified as sensitive, the broker should be configured to allow only clients having a valid TLS certificate. This scheme requires a PKI (public key infrastructure) and a certificate authority that can issue client certificates that the broker trusts. Public MQTT internet service providers typically offer this.

In some situations, it may be suitable to segment different use cases with multiple brokers, either local and/or public brokers. Segmenting critical messages and non-critical messages is a security

control. Multiple brokers will also reduce the risk of single point of failure and enhance monitoring and troubleshooting. The cost is the additional deployment and maintenance of additional brokers.

7 MQTT client in Axis devices

In a standard VMS ecosystem, Axis event notifications from devices are traditionally streamed to a single destination via VAPIX/ONVIF API interface using the RTSP streaming protocol.

The same event notifications can be distributed using the MQTT protocol via the built-in MQTT client of an Axis device (running AXIS OS 9.80 or a later version). This is applicable both within VMS ecosystems and outside of them, and is particularly useful over the internet. With MQTT, multiple subscribed MQTT clients in the network can use and process the event notifications published by the Axis device.

8 MQTT clients in ACAP analytics applications

There are Axis and third-party ACAP apps having their own MQTT clients designed for specific systems, use cases, and subscribers. Axis Publisher is one example that adds additional features, structures, and behavior needed by some systems.

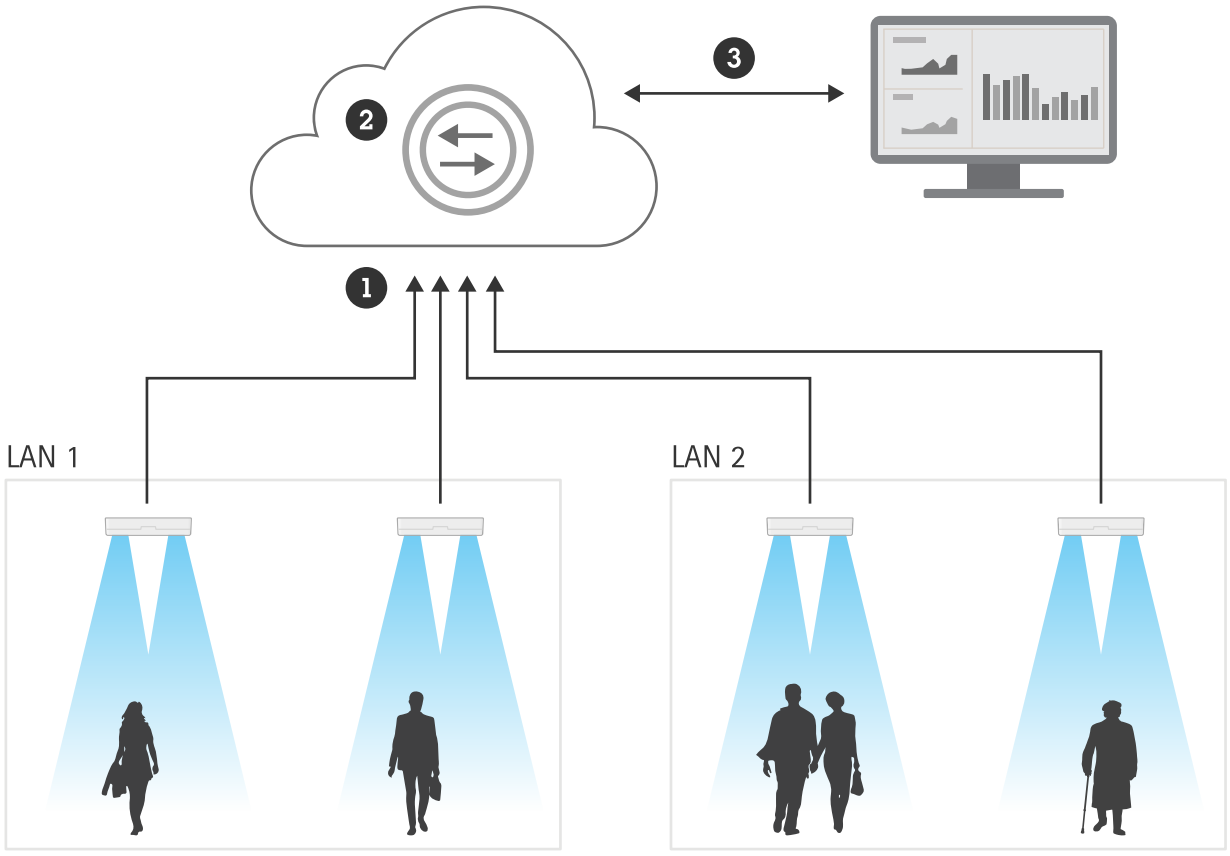
9 Other MQTT clients

There is a wide range of MQTT clients that can be installed on Linux, Windows, Android, and iOS that are designed as tools or services for specific use cases. MQTT is very well suited for scripting and middleware such as Node-RED/Node.js, Python, and PHP. Most internet service platforms such as Microsoft Azure IoT, AWS™, and Google Cloud Platform™ offer MQTT brokers to be integrated into services running on the platform. There are sensors, mobile apps, and (home) automation systems all having an MQTT client.

10 Example use cases integrating devices with MQTT

10.1 People counting analytics data to cloud platform dashboard

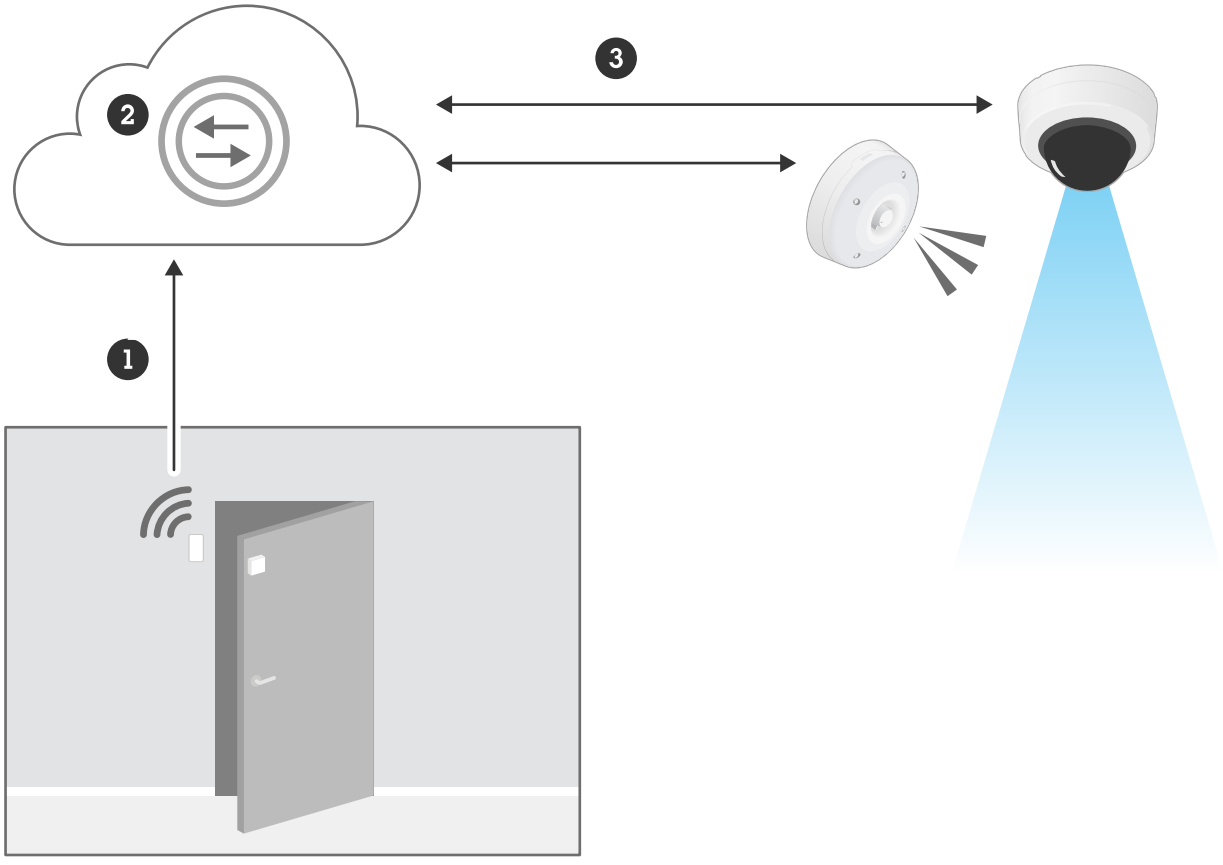
A device with people counting analytics generates an event notification whenever a person is detected passing "in" or "out" of a defined area. The notification is passed on to the MQTT client which publishes it in real time to the cloud platform. In the cloud platform, a connection to data visualization software (for example, a Microsoft® Power BI® dashboard) is created to display the real time statistics from the people counters.



1 Publishing
2 MQTT broker
3 Subscribing

10.2 Door sensor data over MQTT triggers signaling device alarm and camera recording

A third-party MQTT door sensor is used to trigger an event notification if the door is opened. The door sensor publishes an MQTT message to the MQTT broker in the cloud. The signaling device and the camera subscribe to the topic for the door sensor and play an alarm and start a recording if the door is opened.



1 *Publishing*
2 *MQTT broker*
3 *Subscribing*

11 Glossary

ACAP	*AXIS Camera Application Platform*, a framework for applications that add functionality and intelligence at the edge
Aedes	An MQTT broker
API	*Application programming interface*, code that allows two software programs to communicate with each other
AWS™	A cloud service platform
AXIS OS	The operating system for edge devices from Axis
CloudMQTT	An MQTT broker
Eclipse Mosquitto™	An open-source message broker that implements MQTT protocols
Google Cloud Platform™	A cloud service platform
HiveMQ™	An MQTT broker
HTTP	*Hypertext Transfer Protocol*, a data transfer protocol used on the World Wide Web
IBM Cloud®	A cloud service platform
IoT	*Internet of things*, the interconnection over the internet of computing devices embedded in everyday devices and appliances
JSON	*JavaScript Object Notation*, a compact file format and data interchange format
Microsoft® Azure® IoT	A cloud service platform
Microsoft® Power BI®	An interactive data visualization software program focused on business intelligence
MQTT	*Message Queuing Telemetry Transport*, a messaging protocol for the internet of things
Node.js®	An open source development platform for executing JavaScript code serverside
Node-RED®	A programming tool for wiring the internet of things
ONVIF®	An open industry forum that provides and promotes standardized interfaces for effective interoperability of IP-based physical security products
PHP	A general-purpose scripting language geared towards web development
Python®	A general-purpose programming language
RTSP	*Real-Time Streaming Protocol*, a network protocol for establishing and controlling media sessions between endpoints
TCP	*Transmission Control Protocol*, a data transport protocol that is one of the main internet protocols
TLS	*Transport Layer Security*, a protocol that provides confidentiality and integrity of communications over computer networks
VAPIX®	The open application programming interface (API) for Axis products
WebSocket	A communications protocol that provides two-way communication channels over a single TCP connection
VMS	*Video management software* or *video management system*

12 Trademark attributions

Android and Google Cloud Platform are trademarks of Google LLC.

AWS is a trademark of Amazon.com, Inc. or its affiliates in the United States and/or other countries.

Eclipse Mosquitto is a trademark of Eclipse Foundation, Inc.

HiveMQ is a trademark of HiveMQ GmbH.

IBM and IBM Cloud are trademarks of International Business Machines Corp, registered in many jurisdictions worldwide.

IOS is a trademark or registered trademark of Cisco Systems, Inc and/or its affiliates in the United States and certain other countries and is used under license by Apple, Inc.

JavaScript is a registered trademark of Oracle Corporation in the United States.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

Microsoft, Windows, Microsoft Azure IoT, and Microsoft Power BI are registered trademarks of Microsoft Corporation.

Node.js and Node-RED are registered trademarks of the OpenJS Foundation in the United States and/or other countries.

ONVIF is a trademark of Onvif, Inc.

Python is a registered trademark of the Python Software Foundation.

T10162730/EN/M1.17/2203

About Axis Communications

Axis enables a smarter and safer world by creating network solutions that provide insights for improving security and new ways of doing business. As the industry leader in network video, Axis offers products and services for video surveillance and analytics, access control, intercom and audio systems. Axis has more than 3,800 dedicated employees in over 50 countries and collaborates with partners worldwide to deliver customer solutions. Axis was founded in 1984 and has its headquarters in Lund, Sweden.

For more information about Axis, please visit our website *axis.com*.