WHITEPAPER

Geräteintegration mit MQTT

März 2022

AXIS COMMUNICATIONS

Zusammenfassung

MQTT ist ein Standardprotokoll, das den effizienten und zuverlässigen Austausch von Daten zwischen IoT-Geräten und Cloud-Anwendungen erleichtert. Damit können Geräte (über ihre MQTT-Clients) Nachrichten an einen gemeinsamen MQTT-Broker (Server) veröffentlichen, der die Kommunikation mit anderen Geräten vermittelt. Der Vermittler verfolgt, wer was veröffentlicht und wer die Daten sehen will. Er sendet Nachrichten nur an die Clients, die das richtige Thema abonnieren.

In einem typischen VMS-Ökosystem werden Axis Ereignisbenachrichtigungen traditionell mit dem RTSP-Streaming-Protokoll über die VAPIX/ONVIF-API-Schnittstelle zu einem einzigen Bestimmungsort gestreamt. Aber dieselben Benachrichtigungen lassen sich mit dem MQTT-Protokoll über den eingebauten MQTT-Client des Geräts verteilen (gilt für Geräte, die unter Axis OS 9.80 oder höher laufen). Das ist sowohl innerhalb von VMS-Ökosystemen als auch außerhalb möglich und besonders nützlich über das Internet. Mehrere abonnierte MQTT-Clients im Netzwerk können dann die vom Axis Gerät veröffentlichten Ereignisbenachrichtigungen nutzen und verarbeiten. Es gibt auch ACAP-Analyseanwendungen von Axis und Dritten mit eigenen MQTT-Clients, die für spezifische Systeme, Anwendungsfälle und Abonnenten ausgelegt sind.

Als Beispiel für einen Anwendungsfall in Verbindung mit Axis Produkten können Geräte zur Personenzählung Statistiken über MQTT zu Datenvisualisierungs-Software in der Cloud senden. In einem weiteren Beispiel kommuniziert der Türsensor eines Drittanbieters über MQTT mit einem Signalgerät und einer Kamera, die bei jedem Öffnen der Tür einen Alarm abgeben und eine Aufzeichnung starten.

Inhalt

1 Einführung 4

2 MQTT-Protokoll 4

3 Vorteile 5

4 Einschränkungen 6

5 Infrastruktur 6

6 Sicherheit 6

7 MQTT-Client in Axis Geräten 7

8 MQTT-Clients in ACAP-Analyseanwendungen 7

9 Weitere MQTT-Clients 7

10 Beispiel für Anwendungsfälle der Geräteintegration mit MQTT 8

10.1 Analysefunktionsdaten zur Personenzählung an das Plattform-Dashboard der Cloud 8

10.2 Türsensordaten lösen über MQTT den Alarm des Signalgeräts und die Kameraaufzeichnung aus 8

11 Glossar 10

12 Markenzuordnungen 11

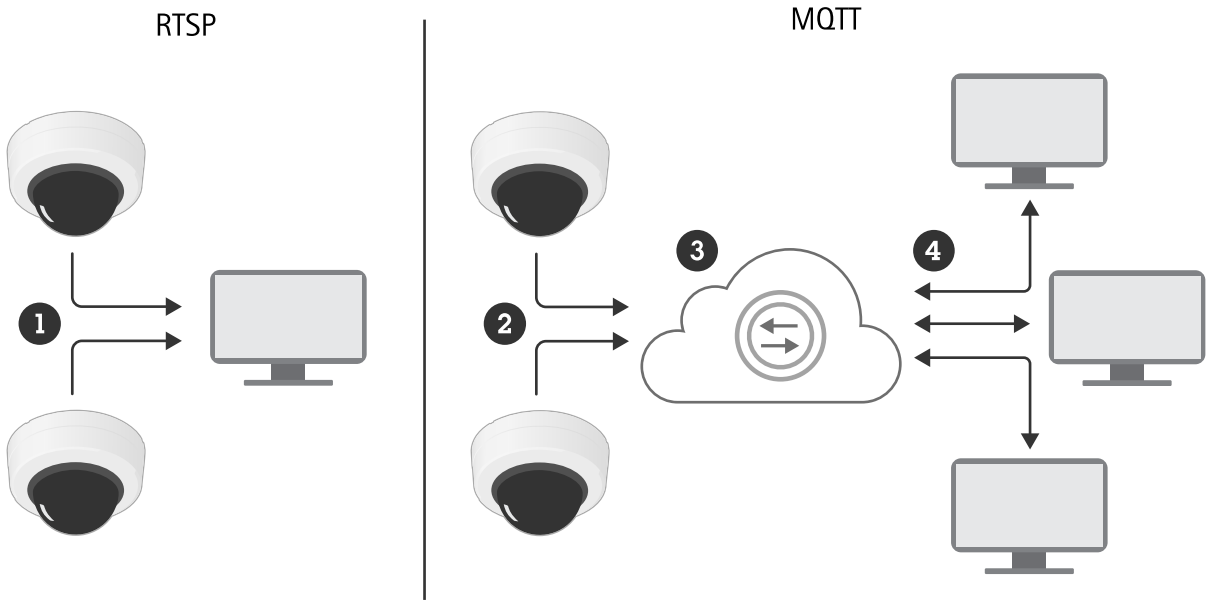
1 Einführung

MQTT (Message Queuing Telemetry Transport) ist ein Standardprotokoll für das Internet der Dinge (IoT). Es wurde für eine vereinfachte IoT-Integration entwickelt und wird in einer Vielzahl von Branchen zum Anschließen von Remote-Geräten mit kleinem Code-Footprint und minimaler Netzwerk-Bandbreite verwendet. Der MQTT-Client im Axis Betriebssystem kann die Integration der im Gerät erzeugten Daten und Ereignisse in Systeme vereinfachen, bei denen es sich nicht um Video Management Systeme (VMS) handelt.

Dieses Whitepaper bietet einen technischen Hintergrund zu MQTT einschließlich typischer Anwendungsfälle, Vorteile und Einschränkungen. Außerdem enthält es Einzelheiten zu MQTT-Clients in Axis Geräten und ACAP-Analysefunktions-Apps.

2 MQTT-Protokoll

MQTT ist ein Publish-Subcribe-Protokoll. Das bedeutet, es hat eine andere Messaging-Struktur als RTSP oder HTTP, die Anforderungs-/Antwort-Protokolle sind. Bei RTSP stellt eine Seite eine Anforderung, und die andere Seite antwortet. Viele mobile Messaging Apps basieren tatsächlich auf MQTT oder ähnlichen Publish-Subscribe-Konzepten. Es gibt auch Publish-Subscribe-Protokolle, die für geschlossene oder spezifische Systeme optimiert sind.



Während RTSP nur eins-zu-eins-Kommunikation zulässt, ist mit MQTT über den Broker auch eins-zu-viele- oder viele-zu-viele-Kommunikation möglich.

1 *Ereignis-Stream*
2 *Publishing*
3 *MQTT-Broker*
4 *Subscribing*

Das MQTT-Konzept besteht darin, dass sich alle Clients mit einem gemeinsamen MQTT-Broker (Server) verbinden, der verfolgt, wer was veröffentlicht und wer die Daten sehen will. Die Verbindung ist typischerweise eine TCP Session an Port 1883. Ein Client kann sich auch per TLS (typischerweise Port 8883) oder per WebSocket (typischerweise Port 1884/8884) verbinden.

Clients veröffentlichen Nachrichten mit einem Thema. Ein anderer Client kann dieses spezifische Thema abonnieren oder Wildcards verwenden, um alle Unterthemen zu erhalten. Eine Nachricht umfasst auch eine Payload, die in der Regel aus einer JSON-Datenstruktur, einer Zeichenkette oder auch kurzen Binärdaten besteht. Der Publisher weiß nicht, ob andere Clients abonnieren. Der Vermittler leitet nur Nachrichten an Kunden weiter, die ein Themenabonnement haben.

Bei MQTT funktioniert es in etwa wie das Senden eines Artikels an ein Magazin. Wer das Magazin abonniert, kann den Artikel lesen, und es kann entweder Eins-zu-eins- oder Eins-zu-viele-Kommunikation sein (und MQTT lässt sogar Viele-zu-viele-Kommunikation zu). Der Artikel kann auch lange nach seiner ursprünglichen Veröffentlichung gelesen werden.

Im Vergleich dazu wäre die Nutzung von RTSP eher wie ein Telefongespräch. Es gibt eine Quelle und ein Ziel für Befehle, und die Kommunikation erfolgt immer eins zu eins. Wenn das Ziel das Gespräch nicht annimmt, verpasst es die Nachricht.

Bei Nutzung von MQTT zur Verbreitung von Axis Ereignis-Benachrichtigungen von Geräten können viele MQTT-Clients im Netzwerk, die Abonnenten sind, die Benachrichtigungen nutzen und verarbeiten. Das ist ein großer Vorteil im Vergleich zum konventionellen Verfahren (der Verwendung der VAPIX®/ONVIF®-API-Schnittstelle (Application Programming Interface) und von RTSP), wobei die Ereignis-Benachrichtigungen stattdessen zu einem einzigen Bestimmungsort gestreamt würden.

3 Vorteile

Die Nutzung von MQTT hat mehrere Vorteile. Im Vergleich zum Einsatz eines Anforderungs-/Antwortprotokolls wie beispielsweise RTSP gehören folgende zu den Hauptvorteilen von MQTT:

- **Geringeres Risiko exponierter Gerätekennwörter.** Ein Client muss nicht auf ein Gerät oder einen Server zugreifen, um die Daten zu holen. Das bedeutet, dass der Client weder das Kennwort noch die Funktionsweise der API kennen muss. Das reduziert die Gefahr der Exposition von Gerätekennwörtern gegenüber Clients und Benutzern und damit auch das Risiko absichtlichen oder zufälligen Missbrauchs.
- **Einzelner Integrationspunkt.** Bei entsprechender Berechtigung können alle Clients die veröffentlichten Nachrichten aller anderen Clients mit einer einzigen Verbindung zu einem Vermittler erhalten. In RTSP muss sich der Client mit jedem Client verbinden, von dem er Daten holen will. Das bedeutet, dass der MQTT-Nachrichtenstrom ohne Zusatzbelastung für jeden Client eins zu eins, eins zu viele oder viele zu eins verlaufen kann.
- **Mit intakter Firewall veröffentlichen und abonnieren** In RTSP muss die Geräte-/Server-API für den Client zugänglich sein. Wenn sich das Gerät hinter einer Firewall befindet und der Client remote ist, muss die Firewall so konfiguriert sein, dass eingehende Anforderungen zugelassen werden, wodurch die Geräte-API exponiert ist. Mit einem Public MQTT-Broker dazwischen können Clients hinter einer Firewall spezifische Daten veröffentlichen / abonnieren, ohne die Firewall zu durchlöchern (wenn die Firewall ausgehende Verbindungen zulässt).
- **Quality of Service.** Beim Posten einer kritischen Nachricht kann ein Publisher überwachen, ob diese Nachricht von einem anderen Client erhalten wurde, und ansonsten alternative Aktionen durchführen.
- **Gespeicherte Nachrichten.** Publisher können eine Nachricht als gespeichert markieren, was bedeutet, dass der Vermittler eine Kopie der Nachricht behält und diese Nachricht an neu verbundene Clients schickt, die dieses Thema abonnieren.
- **Verfügbarkeit des IoT Client.** MQTT Client-Pakete sind für alle üblichen Software-Entwicklungsumgebungen erhältlich, einschließlich Windows®, Linux®, Android™, iOS, Node.js®, PHP

und Python®. Es gibt viele weitere Clients, die sich mit einem Vermittler verbinden können, im Gegensatz zur Einrichtung eines RTSP-Datenstroms zu einem Gerät.

- **Vereinfachte Nachrichtenüberwachung und Debugging** Es gibt mehrere MQTT Tools, mit denen alle veröffentlichten Nachrichten überwacht und auch Nachrichten zur Fehlersuche veröffentlicht werden können, wenn und wie Abonnenten reagieren.
- **Vereinfachte Datenstruktur.** Da MQTT oft auf unbekannte Clients zielt, wird dies zur Vereinfachung für den Abonnenten in der Regel von der Nachrichten-Payload berücksichtigt.

4 Einschränkungen

Im Vergleich zu alternativen Protokollen hat MQTT einige Nachteile.

- **Einzelne Fehlerstelle.** Wenn der Vermittler nicht verfügbar ist, funktioniert keine Nachrichtenübermittlung mehr. Die Infrastruktur kann jedoch mit Redundanz-Vermittlern ausgelegt werden.
- **Wer hat eine Nachricht veröffentlicht?** Seiner Auslegung nach ist MQTT auf das Thema und nicht darauf fokussiert, wer die Nachricht veröffentlicht hat. Soweit der Publisher keine ID als Teil des Themas oder der Payload umfasst, muss auf das Protokoll des Vermittlers zugegriffen werden, um festzustellen, wer die Nachricht veröffentlicht hat. Es ist gängige Praxis, dass Publisher je nach Anwendungsfall im Thema oder in der Payload eine Client-ID umfassen.
- **Ein bösartiger Client,** der mit dem Vermittler verbunden ist, kann jedes Thema veröffentlichen/abonnieren, für das er eine Berechtigung hat. Der Vermittler muss geschützt werden (vgl. Abschnitt zur MQTT-Sicherheit).
- Er ist **nicht zum ständigen Streamen von Video/Audio ausgelegt.**

Und wie bei jedem Server muss der Gesamt-Bandbreitendurchsatz berücksichtigt werden. Bei sehr großen Systemen mit vielen Clients kann dynamische Skalierung erforderlich sein.

5 Infrastruktur

Es ist recht einfach, einen lokalen Eclipse Mosquitto™ Broker einzurichten oder Node-RED® für die Funktion als lokaler Vermittler wie beispielsweise Aedes zu aktivieren. Außerdem gibt es eine Reihe von Internetdienstanbietern und anderen, die Managed MQTT Broker anbieten, wie etwa Microsoft® Azure® IoT, HiveMQ™, CloudMQTT und IBM® Cloud®.

Wenn ein System keine Remote Clients hat, wird der Einsatz eines lokalen Vermittlers empfohlen. Ein lokaler Vermittler kann auch als Proxy für einen öffentlichen Vermittler (Public Broker) fungieren oder so konfiguriert werden, dass er für ausgewählte Nachrichten des lokalen Brokers und des Public Brokers als Proxy fungiert.

6 Sicherheit

Der Vermittler (= Broker) benötigt geeigneten Schutz je nachdem, wie kritisch Nachrichten sind und welchen Bedrohungen ein bestimmtes System ausgesetzt sein kann. MQTT bietet verschiedene Authentifizierungssysteme einschließlich Keine Authentifizierung, Benutzer/Kennwort und TLS-Client-Zertifikats-Authentifizierung. Verschiedene Benutzer können unterschiedliche Berechtigungen

dafür haben, zu welchem Thema sie veröffentlichen oder abonnieren können. Der Broker kann Clients erlauben, sich über unverschlüsseltes TCP oder über verschlüsseltes TLS (wie HTTPS) zu verbinden.

- **Keine Authentifizierung** Ein lokaler Vermittler kann Authentifizierung deaktivieren, wenn die Nachrichten unkritisch sind und der Broker nicht gegenüber Internet-Clients exponiert ist. Es wird empfohlen, diese Option nur für Tests, Sandbox-Entwicklung und Demonstrationen zu verwenden.
- **Benutzer / Kennwort.** Dies ist die häufigste Variante. Je nachdem, welche Risiken das System birgt, kann der Systemadministrator allen MQTT-Clients denselben Benutzer und dasselbe Passwort zuweisen oder Benutzer mit eingeschränktem Zugriff auf das Thema erstellen.
- **TLS-Zertifikate.** Vermittler im Internet, deren Nachrichten als sensibel eingestuft sind, sollten so konfiguriert werden, dass sie nur Clients mit einem gültigen TLS-Zertifikat zulassen. Dieses Schema setzt eine PKI (Public Key Infrastructure) und eine Zertifizierungsstelle voraus, die für den Vermittler vertrauenswürdige Client-Zertifikate ausstellen kann. Public-MQTT-Internetdienstleister bieten dies in der Regel an.

In einigen Fällen kann es angebracht sein, verschiedene Anwendungsfälle mit mehreren Vermittlern, entweder lokale und/oder Public Broker, zu segmentieren. Die Segmentierung kritischer und unkritischer Nachrichten ist eine Sicherheitskontrolle. Mehrere Vermittler reduzieren außerdem das Risiko eines „Single Point of Failure" (einzelne Fehlerstelle) und verbessern die Überwachung und Fehlersuche. Die Kosten bestehen in der zusätzlichen Bereitstellung und Wartung zusätzlicher Vermittlers.

7 MQTT-Client in Axis Geräten

In einem Standard-VMS-Ökosystem werden Axis Ereignisbenachrichtigungen von Geräten traditionell mit dem RTSP-Streaming-Protokoll über die VAPIX/ONVIF-API-Schnittstelle zu einem einzigen Bestimmungsort gestreamt.

Dieselben Ereignisbenachrichtigungen können mit dem MQTT-Protokoll über den eingebauten MQTT-Client eines Axis Geräts (mit AXIS OS 9.80 oder höher) verteilt werden. Das gilt sowohl innerhalb von VMS-Ökosystemen als auch außerhalb und ist besonders nützlich über das Internet. Mit MQTT können mehrere abonnierte MQTT-Clients im Netzwerk die vom Axis Gerät veröffentlichten Ereignisbenachrichtigungen nutzen und verarbeiten.

8 MQTT-Clients in ACAP-Analyseanwendungen

Es gibt ACAP Apps von Axis und Dritten mit eigenen MQTT-Clients, die für spezifische Systeme, Anwendungsfälle und Teilnehmer ausgelegt sind. Axis Publisher ist ein Beispiel, das zusätzliche Funktionen, Strukturen und Verhaltensweisen ergänzt, die von einigen Systemen benötigt werden.

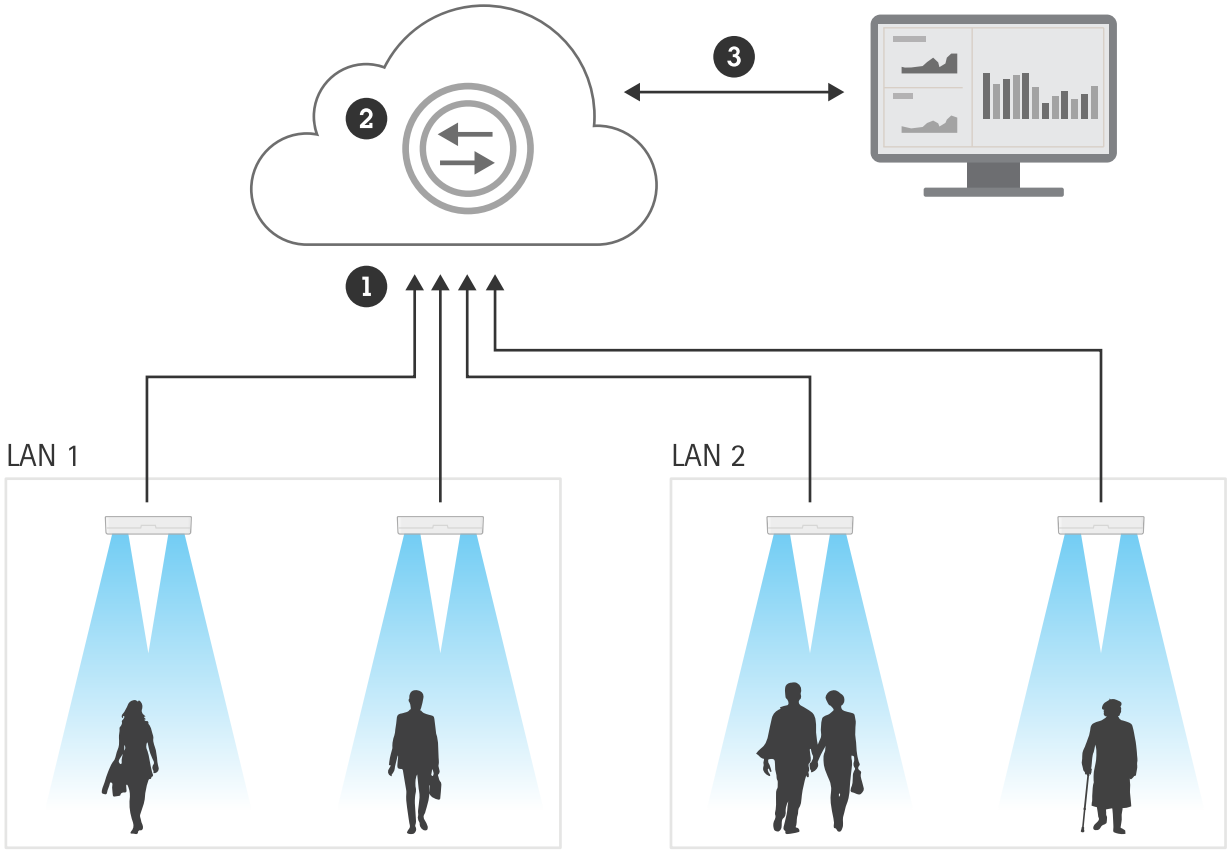
9 Weitere MQTT-Clients

Es gibt eine breite Palette von MQTT-Clients, die sich unter Linux, Windows, Android und iOS installieren lassen und als Tools oder Services für spezifische Anwendungsfälle vorgesehen sind. MQTT eignet sich sehr gut zum Skripten und für Middleware wie beispielsweise Node-RED/Node.js, Python und PHP. Die meisten Service-Plattformen im Internet wie etwa Microsoft Azure IoT, AWS™, und Google Cloud Platform™ bieten MQTT-Broker zur Integration in Services, die auf der Plattform laufen. Es gibt Sensoren, mobile Apps und (Haus-)Automatisierungssysteme, die alle einen MQTT-Client haben.

10 Beispiel für Anwendungsfälle der Geräteintegration mit MQTT

10.1 Analysefunktionsdaten zur Personenzählung an das Plattform-Dashboard der Cloud

Ein Gerät mit Analysefunktionen zur Personenzählung generiert jedes Mal eine Ereignisbenachrichtigung, wenn eine Person erfasst wird, die einen definierten Bereich betritt oder verlässt. Die Benachrichtigung wird an den MQTT-Client weitergeleitet, der sie in Echtzeit zur Cloud-Plattform veröffentlicht. Auf der Cloud-Plattform wird eine Verbindung zu Datenvisualisierungs-Software hergestellt (zum Beispiel einem Microsoft® Power BI® Dashboard), um die Echtzeitstatistiken der Personenzählung anzuzeigen.

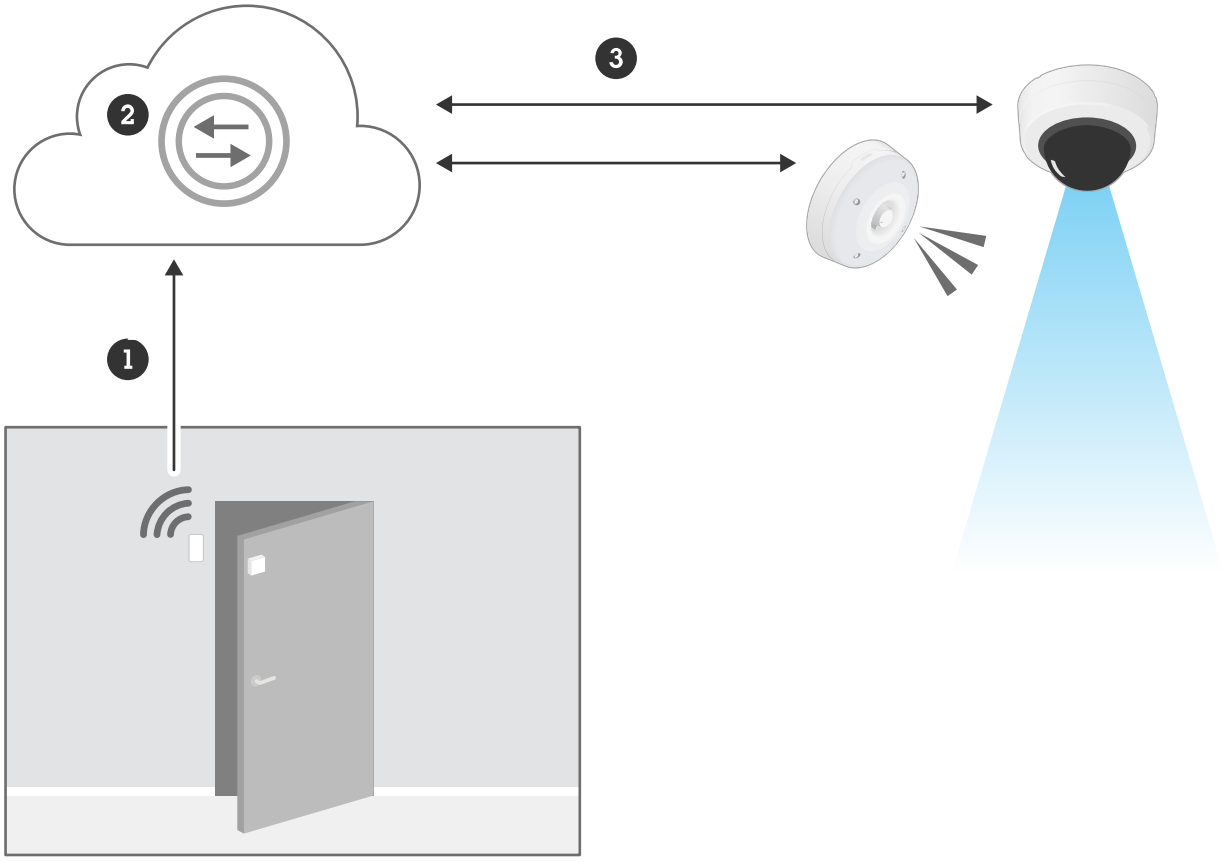


1 *Publishing*
2 *MQTT-Broker*
3 *Subscribing*

10.2 Türsensordaten lösen über MQTT den Alarm des Signalgeräts und die Kameraaufzeichnung aus

Der MQTT-Türsensor eines Drittanbieters wird eingesetzt, um beim Öffnen der Tür eine Ereignisbenachrichtigung auszulösen. Der Türsensor veröffentlicht eine MQTT-Nachricht an den

MQTT-Broker in der Cloud. Das Signalgerät und die Kamera abonnieren das Thema für den Türsensor; sie geben einen Alarm aus und starten eine Aufzeichnung, wenn die Tür geöffnet wird.



1 *Publishing*
2 *MQTT-Broker*
3 *Subscribing*

11 Glossar

ACAP	*AXIS Camera Application Platform*, ein Rahmen für Anwendungen, die Funktionen und Intelligenz am Rande des Netzwerks (on the edge) hinzufügen
Aedes	Ein MQTT-Broker
API	*Application Programming Interface*, ein Code, mit dem zwei Softwareprogramme miteinander kommunizieren können
AWS™	Eine Cloud-Service-Plattform
AXIS OS	Das Betriebssystem für Edge-Geräte von Axis
CloudMQTT	Ein MQTT-Broker
Eclipse Mosquitto™	Ein Open-Source-Nachrichten-Broker, der MQTT-Protokolle implementiert
Google Cloud Platform™	Eine Cloud-Service-Plattform
HiveMQ™	Ein MQTT-Broker
HTTP	*Hypertext Transfer Protocol*, ein Datenübertragungsprotokoll, das im World Wide Web genutzt wird
IBM Cloud®	Eine Cloud-Service-Plattform
IoT	*Internet der Dinge*, die Vernetzung intelligenter Geräte in Alltags- und Haushaltsapparaten über das Internet
JSON	*JavaScript Object Notation*, ein kompaktes Dateiformat und Datenaustauschformat
Microsoft® Azure® IoT	Eine Cloud-Service-Plattform
Microsoft® Power BI®	Ein Softwareprogramm zur interaktiven Datenvisualisierung mit Fokus auf Unternehmensintelligenz
MQTT	*Message Queuing Telemetry Transport*, ein Nachrichtenprotokoll für das Internet der Dinge
Node.js®	Eine Open-Source-Entwicklungsplattform zur serverseitigen Ausführung des JavaScript-Codes
Node-RED®	Ein Programmier-Tool zum Erstellen von Verbindungen im Internet der Dinge
ONVIF®	Ein offenes Branchenforum, das standardisierte Schnittstellen für eine effektive Kompatibilität IP-basierter physischer Sicherheitsprodukte bereitstellt und fördert
PHP	Eine Skriptsprache zum allgemeinen Gebrauch, die auf Webentwicklung ausgerichtet ist
Python®	Eine Programmiersprache zum allgemeinen Gebrauch
RTSP	*Real-Time Streaming Protocol*, ein Netzwerkprotokoll zum Einrichten und Steuern von Mediensitzungen zwischen Endpunkten
TCP	*Transmission Control Protocol*, ein Datentransportprotokoll, das zu den wichtigsten Internetprotokollen gehört
TLS	*Transport Layer Security*, ein Protokoll, das für Datenschutz und Integrität bei der Kommunikation über Computernetzwerke sorgt

VAPIX®	Die offene Anwendungsprogrammierschnittstelle API (Application Programming Interface) für Axis Produkte
WebSocket	Ein Kommunikationsprotokoll, das bidirektionale Kommunikationskanäle über eine einzelne TCP-Verbindung bietet
VMS	*Video Management Software* oder *Video Management System*

12 Markenzuordnungen

Android und Google Cloud sind Marken von Google LLC.

AWS ist eine Marke von Amazon.com, Inc. oder ihren Tochtergesellschaften in den Vereinigten Staaten und/oder anderen Ländern.

Eclipse Mosquitto ist eine Marke von Eclipse Foundation, Inc.

HiveMQ ist eine Marke von HiveMQ GmbH.

IBM und IBM Cloud sind Marken von International Business Machines Corp. und in vielen Rechtsräumen weltweit eingetragen.

IOS ist eine Marke oder eingetragene Marke von Cisco Systems, Inc und/oder ihren Tochtergesellschaften in den Vereinigten Staaten und bestimmten anderen Ländern und wird unter Lizenz von Apple, Inc. genutzt.

JavaScript ist eine eingetragene Marke von Oracle Corporation in den Vereinigten Staaten.

Linux ist die eingetragene Marke von Linus Torvalds in den USA und anderen Ländern.

Microsoft, Windows, Microsoft Azure IoT, und Microsoft Power BI sind eingetragene Marken von Microsoft Corporation.

Node.js und Node-RED sind eingetragene Marken der OpenJS Foundation in den Vereinigten Staaten und/oder anderen Ländern.

ONVIF ist eine eingetragene Marke von Onvif, Inc.

Python ist eine eingetragene Marke der Python Software Foundation.

T10162730/DE/M1.17/2203

Über Axis Communications

Axis ermöglicht eine intelligente und sichere Welt durch Lösungen zur Verbesserung der Sicherheit und Geschäftsperformance. Als Unternehmen für Netzwerktechnologie und Branchenführer bietet Axis Lösungen in den Bereichen Videosicherheit, Zutrittskontrolle sowie Intercoms und Audiosysteme. Sie werden verstärkt durch intelligente Analyseanwendungen und unterstützt durch gute Schulungen.

Axis beschäftigt rund 4.000 engagierte Mitarbeiter in über 50 Ländern und arbeitet weltweit mit Technologie- und Systemintegrationspartnern zusammen, um den Kunden Lösungen anbieten zu können. Axis wurde 1984 gegründet und der Hauptsitz befindet sich in Lund, Schweden

AXIS COMMUNICATIONS