influxdata®

MQTT Cheat Sheet

MQTT is a messaging protocol designed for the Internet of Things (IoT) with the purpose of providing lightweight and reliable messaging between remote devices. This guide will cover some of the most important concepts, features, and tools of MQTT and the surrounding ecosystem.

InfluxDB tools for working with MQTT

Native MQTT Collector
InfluxDB Cloud provides a native integration for MQTT which allows you to ingest data directly from MQTT brokers

www.influxdata.com/products/data-collection/cloud-native/

Telegraf
Easily send your MQTT data to InfluxDB or other data stores with the Telegraf MQTT plugin.

https://www.influxdata.com/blog/mqtt-topic-payload-parsing-telegraf/

Client Libraries
If you prefer working with MQTT using a language-specific client library, you can also use the InfluxDB client libraries to store data in InfluxDB.

https://docs.influxdata.com/influxdb/cloud/api-guide/client-libraries/

MQTT Basics

Why was MQTT created?
MQTT was created by IBM in the 90's as a bespoke network protocol used to connect sensors on oil pipelines via satellite. Because of the nature of the sensors and the cost of bandwidth via satellite link at the time, this protocol needed to be extremely bandwidth efficient, lightweight, and able to handle intermittent connectivity. These features made MQTT ideal for many IoT use cases and in 2013 IBM submitted the MQTT specification to OASIS. OASIS took over maintenance of the standard and MQTT rapidly gained adoption in the IoT community.

MQTT Concepts

Broker
MQTT brokers receive and pass along messages created by connected client devices. These messages are organized by topics that clients can subscribe to. There are many different choices for MQTT brokers, which are all implementations of the OASIS specification for MQTT.

Some popular brokers are Mosquitto, Aedes, and HiveMQ. You have the option of hosting your own MQTT broker or using a hosted solution to abstract the complexity of managing your own broker.

Client
MQTT clients are anything that connects to the MQTT broker. Client devices can publish messages as well as subscribe to listen for messages being created by other clients. MQTT is optimized for devices with limited computing resources and bandwidth.

Topics
Topics are UTF-8 strings that the MQTT broker uses to filter messages. A topic can have numerous subtopics by using forward slashes. Topics do not need to be initialized, the broker will simply accept each topic and dispatch to any clients listening. Topics are case sensitive, allow spaces, and require at least 1 character.

For subscribing to topics MQTT brokers allow for wildcard operators to give users more flexibility for defining which messages client devices receive. A single-level wildcard is defined with a plus sign and will match for any arbitrary string on that level. A multi-level wildcard is defined with a hashtag and will match for all strings for multiple levels after the hashtag.

Quality of service
MQTT allows the operator to define quality of service levels for messages depending on use case requirements. The three service levels are:

- At most once
- At least once
- Exactly once

At most once is the lowest service level and doesn't guarantee delivery to client devices. The client sends a single message to the broker and doesn't check if it was delivered. This could result in data being lost if the network isn't stable. The second level of service is at least once, which may result in multiple copies of the same message being received by clients. The third level of service is exactly once. This is the most resource-intensive service level because it requires multiple requests and responses between the client and broker.

MQTT Use Cases

MQTT is a versatile network protocol that is ideal for any situation where your network might be unreliable, you want to minimize bandwidth consumption, have low-powered hardware, or you have an architecture where you have many client devices that will need access to the same data in near real-time.

Consumer IoT
MQTT is used for communication between many consumer IoT devices. This could include everything from smart home type devices designed for automating things and making your home more efficient.

Industrial IoT
MQTT was designed for industrial use in the oil and gas industry and has rapidly spread to other areas like manufacturing due to the many benefits it has over competing network protocols. MQTT helps companies become more efficient by allowing them to collect more data and act on that data faster.

Logistics
MQTT is ideal for real-time monitoring of assets as they move all around the world. Even if internet connectivity is lost, MQTT is designed to ensure delivery of data once it is regained. In a world with increasingly complex supply chains, MQTT is becoming even more important.

Mobile application development
One app that uses MQTT that you might be familiar with is Facebook Messenger. The Facebook Messenger team chose MQTT over HTTP because mobile applications are fairly similar to IoT workloads and for group chats especially MQTT's pub/sub architecture was very natural and made it easy to add members, with each group chat being its own topic that clients could subscribe to. For general mobile apps, MQTT is ideal because it allows for persistent connections to receive new messages without consuming large amounts of bandwidth or burning through the phone's battery. If you are building any mobile app that will require network communication frequently, MQTT may be a solid choice.

MQTT Best Practices

Security
Security is a challenge with any type of software, but internet of things applications bring a whole other level of security challenges. Because IoT devices often have limitations when it comes to compute and memory resources, developers have to make tradeoffs with what security tools, like encryption algorithms, they can use. Another challenge is dealing with updates to devices in the field efficiently. If devices can't be remotely accessed, software updates become more expensive and will be less frequent.

At the application level MQTT can be set up to require a basic username and password for authentication. Different MQTT broker implementations may also provide additional security features. Because it is built on TCP, MQTT also gives the option of transport-layer level security using SSL/TLS to encrypt MQTT message data. At the network level, developers also have the option of trying to physically secure hardware devices and network components, as well as using Virtual Private Networks(VPNs) for communication between clients and brokers.

Topic structure and naming
MQTT topics can become messy if you don't enforce some kind of standard. Define consistent standards like how to handle case sensitivity, spaces, dashes, etc. General best practice is to only use lowercase letters and use dashes rather than space to make debugging easier. For level structuring it's recommended to move from a more general topic to more specific as you nest your levels.

Avoid Multi-level Wildcards for high-traffic applications
If you use multi-level wildcards for high traffic topics you may run into performance problems because every client device will receive every message coming through the topic level specified by the wildcard.

MQTT Ecosystem and Tools

MQTT brokers

Mosquitto
Mosquitto is the most popular open source MQTT broker and is supported by the Eclipse foundation. Mosquitto is probably the best MQTT broker available if you don't want to be reliant on a broker implementation supported by a vendor that has somewhat conflicting interests due to offering commercial alternatives to their open source implementations.

EMQX
EMQX is an open source, cloud native MQTT broker created by the company EMQ, which is a foundational sponsor of OASIS alongside IBM. EMQX was the first MQTT broker to fully support MQTT version 5 functionality and EMQ claims a cluster of EMQX brokers can handle 100 million subscribers. EMQ also provides a number of other useful open source tools for working with MQTT like NanoMQ and Neuron which work together with EMQX to help connect IoT devices from edge to cloud.

HiveMQ
HiveMQ is another popular vendor in the MQTT ecosystem. HiveMQ provides an open source broker that supports MQTT version 3 and 5. HiveMQ also provides a hosted MQTT broker that can be used for scaling your MQTT application and integrates well with InfluxDB.

MQTT client libraries
There are numerous client libraries available for working with MQTT in your language of choice. Most vendors provide their own implementations, but the most popular client libraries are the Paho MQTT libraries supported by the Eclipse Foundation.

Data storage for MQTT
MQTT messages often contain time series data. This data is streaming in very fast from potentially a large number of devices. For many IoT-related workloads, that data also needs to be able to be analyzed shortly after being ingested — long delays are unacceptable for monitoring situations where near real-time access to data is needed. A time series database like InfluxDB is designed for these requirements and provides a number of benefits in terms of pure performance when it comes to writing and querying data, as well as benefits like faster development time due to a number of built-in features that are provided out of the box like downsampling and common analysis and forecasting functions required for time series data.

Additional Resources

1. Explore the InfluxDB documentation
2. See how other companies are using InfluxDB for IoT
3. Data collection

Have additional questions about MQTT and InfluxDB?
Join the InfluxDB Slack community - influxdata.com/slack