MQTT Essentials

N° de la lecture individuelle :	1
Étudiant	Laurent Térence
Sujet	MQTT essentials
Source	https://www.hivemq.com/downloads/hivemq-ebook-mqtt-essentials.pdf?_hsmi=278147580

MQTT Essentials

Table des matières

Table des illustrations

1 Introduction

2 Introduction MQTT

2.1 MQTT Protocol et son invention

Le protocole MQTT, créé en 1999 par Andy Stanford-Clark (IBM) et Arlen Nipper (Arcom, aujourd'hui Cirrus Link).

L'invention avait pour objectif de minimiser la perte de batterie et l'utilisation de bande passante pour la connexion avec les pipelines pétroliers via satellite.

Les inventeurs ont défini des exigences spécifiques pour le protocole, notamment :

- Implémentation simple
- Livraison des données avec un niveau de service de qualité
- Léger et efficient en bande passante
- Agnostique aux données
- Conscience continue de la session

Ces exigences demeurent au cœur du protocole MQTT.

2.2 Evolution et changement de focalisation

Au fil du temps, la focalisation principale de MQTT a évolué des systèmes embarqués propriétaires vers des cas d'utilisation ouverts de l'IdO (Internet des Objets).

Ce changement a entraîné une confusion quant à la signification de l'acronyme MQTT, qui n'est plus considéré comme un acronyme. L'acronyme initial signifiait MQ Telemetry Transport, faisant référence à la série MQ d'IBM.

Malgré les idées préconçues, MQTT n'est pas une solution de file d'attente de messages traditionnelle. IBM a utilisé le protocole en interne pendant dix ans avant de publier la version sans redevance MQTT 3.1 en 2010.

Depuis lors, le protocole est ouvert à la mise en œuvre et à l'utilisation par n'importe qui.

3 Architecture Publish/Subscribe

Dans ce chapitre, nous explorons l'architecture Publish/Subscribe (Pub/Sub) et la comparons au modèle traditionnel Client-Serveur.
Dans le modèle **Client-Serveur**, la communication est directe entre un client et un point final. En revanche, le modèle Pub/Sub désaccouple l'expéditeur (client) du ou des destinataires (abonnés) grâce à l'intervention d'un courtier.

Un des principaux avantages du modèle Pub/Sub réside dans le **désaccouplement spatial**. Contrairement au modèle Client-Serveur, il n'est pas nécessaire que les éditeurs et les abonnés soient conscients les uns des autres, évitant ainsi l'échange d'adresses IP et de ports.

Le désaccouplement temporel est un autre aspect essentiel du modèle Pub/Sub. Les éditeurs et les abonnés n'ont pas besoin de fonctionner simultanément, offrant ainsi une flexibilité temporelle dans les opérations.

De plus, le modèle Pub/Sub propose un **désaccouplement de la synchronisation**. Les opérations sur les deux composants ne sont pas interrompues pendant la publication ou la réception, assurant un flux continu.

En résumé, l'architecture Pub/Sub présente des avantages significatifs, offrant un contrôle précis sur la distribution des messages aux abonnés et fournissant un désaccouplement spatial, temporel et de synchronisation.

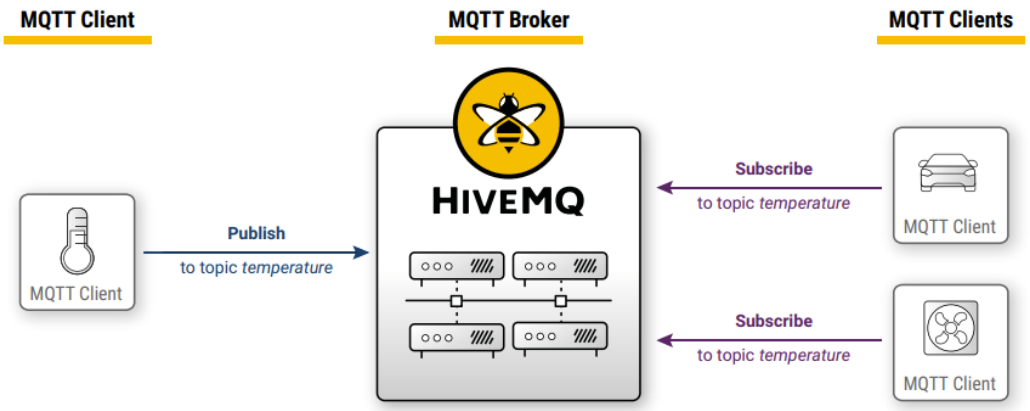


Figure 1 : MQTT Clients communicating through MQTT Brocker,

src:"https://www.hivemq.com/downloads/hivemq-ebook-mqtt-essentials.pdf?_hsmi=278147580"

Scalabilité dans Pub/Sub

Ce chapitre explore la scalabilité dans le contexte de l'architecture Publish/Subscribe (Pub/Sub) par rapport au modèle traditionnel client-serveur. L'architecture Pub/Sub offre une meilleure évolutivité grâce à la parallélisation intensive des opérations du courtier. La mise en cache des messages et le routage intelligent améliorent considérablement la capacité à évoluer.

Les défis liés à la gestion de millions de connexions sont résolus grâce à l'utilisation de « clustered broker » regroupés et d'équilibreurs de charge (« load balancers »).

Filtrage des messages

Un élément clé de l'architecture Pub/Sub réside dans le filtrage des messages. Le courtier joue un rôle central avec diverses options de filtrage, notamment le filtrage basé sur le sujet (topics), le filtrage basé sur le contenu grâce à un langage de filtrage, et le filtrage basé sur le type ou la classe du message, souvent utilisé dans les langages orientés objet.

L'intégration de MQTT dans l'architecture Pub/Sub est également explorée. MQTT incarne les aspects de publication/abonnement, se désaccouple spatialement et temporellement, et fonctionne de manière asynchrone. Il s'avère convivial du côté client, particulièrement pour les petits appareils contraints. L'utilisation de MQTT comprend le filtrage basé sur le sujet (topics), et les niveaux de qualité de service (QoS) résolvent les problèmes de livraison des messages.

Distinction: MQTT & Message queue

Enfin, une distinction est établie entre MQTT et les files d'attente de messages. Contrairement à une file d'attente, chaque message entrant dans MQTT est stocké jusqu'à ce qu'il soit récupéré par un client. Dans une file d'attente traditionnelle, un message ne peut être traité que par un seul consommateur, les files d'attente sont nommées et doivent être créées explicitement.

4 Etablissement de la connexion

Ce chapitre examine la relation étroite entre le protocole MQTT et TCP/IP. Le protocole MQTT repose entièrement sur la suite de protocoles TCP/IP, nécessitant la présence d'une pile TCP/IP tant du côté du client que du courtier.

L'établissement de la connexion est toujours unidirectionnel, entre un client et le courtier. Il est impératif de souligner que les clients ne se connectent jamais directement les uns aux autres. Le processus démarre par l'initiative du client, qui envoie un message CONNECT au courtier. Ce dernier répond avec un message CONNACK (Accusé de réception) et un code d'état. La connexion reste ouverte jusqu'à ce que le client envoie une commande de déconnexion ou que la connexion soit interrompue.

La section sur le message CONNECT met en lumière l'importance de messages CONNECT bien formés et sans délai pour éviter que le courtier ne ferme la connexion, ce qui dissuaderait les clients malveillants. Les clients bien intentionnés envoient un message CONNECT avec un contenu essentiel, incluant le ClientId.

La réponse du broker, matérialisée par le message CONNACK, est une obligation à la réception d'un message CONNECT. Son contenu comprend deux aspects cruciaux. Tout d'abord, le flag "Session Present" indique si le broker détient une session persistante de précédentes interactions avec le client. Ensuite, le flag "Connect Acknowledge" contient un code de retour indiquant le succès ou l'échec de la tentative de connexion.

CONNECT Message Client initiates connection

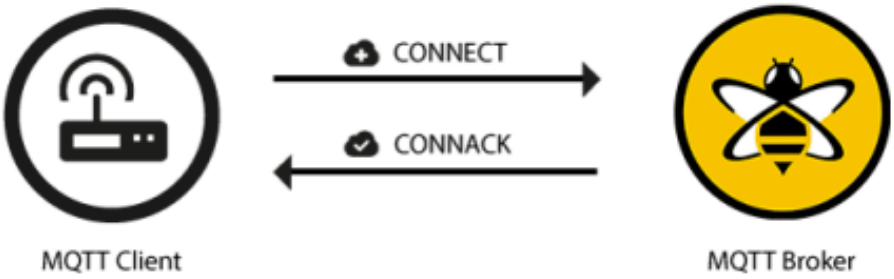


Figure 2: Connection communication message,
src : « https://www.hivemq.com/downloads/hivemq-ebook-mqtt-essentials.pdf?_hsmi=278147580 »

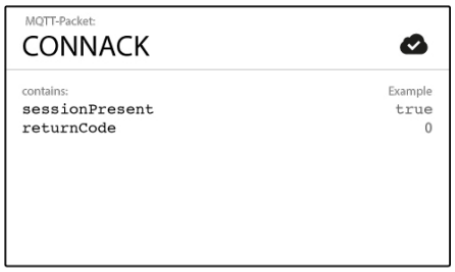
MQTT-Packet:

CONNECT

contains:	Example
clientId	"client-1"
cleanSession	true
username (optional)	"hans"
password (optional)	"letmein"
lastWillTopic (optional)	"/hans/will"
lastWillQos (optional)	2
lastWillMessage (optional)	"unexpected exit"
lastWillRetain (optional)	false
keepAlive	60

Figure 3 Connect Message,
src: "https://www.hivemq.com/downloads/hivemq-ebook-mqtt-essentials.pdf?_hsmi=278147580"

CONNACK Message - Brocker Response



Return Code	Return Code Response
0	Connection accepted
1	Connection refused, unacceptable protocol version
2	Connection refused, identifier rejected
3	Connection refused, server unavailable
4	Connection refused, bad user name or password
5	Connection refused, not authorized

5 Publish, Subscribe & Unsubscribe

5.1 Publish

Ce chapitre explore le processus de publication dans MQTT, mettant l'accent sur le filtrage basé sur les sujets et le rôle essentiel du courtier dans le traitement des messages.

Le filtrage basé sur les sujets est une obligation du courtier MQTT. Chaque message doit comporter un « topic », qui est utilisé pour le transférer aux clients intéressés. Cette approche démontre la flexibilité et la puissance du modèle de publication/abonnement, permettant aux clients de choisir les messages pertinents en fonction des « topics ».

La structure des messages MQTT comprend un payload, qui contient les données au format binaire. Cette agnosticisme vis-à-vis des données permet une variété de structures de payload, telles que binaire, texte, XML ou JSON, déterminées par le client émetteur (l'éditeur).

Les attributs d'un message PUBLISH, tels que le nom du sujet, sont structurés de manière hiérarchique à l'aide de barres obliques. Par exemple, "device/mobile12121/getAllDevices" ou "Germany/Munich/Octoberfest/people".

Lorsqu'un client envoie un message au courtier pour publication, le courtier lit le message, le reconnaît en fonction du niveau de service de qualité (QoS) spécifié par le client, puis le traite. Le traitement consiste à déterminer quels clients sont abonnés au sujet associé au message et à transférer le message à tous les clients abonnés au sujet pertinent. Ce processus garantit que seuls les clients intéressés reçoivent les messages pertinents, offrant ainsi une communication ciblée et efficace.

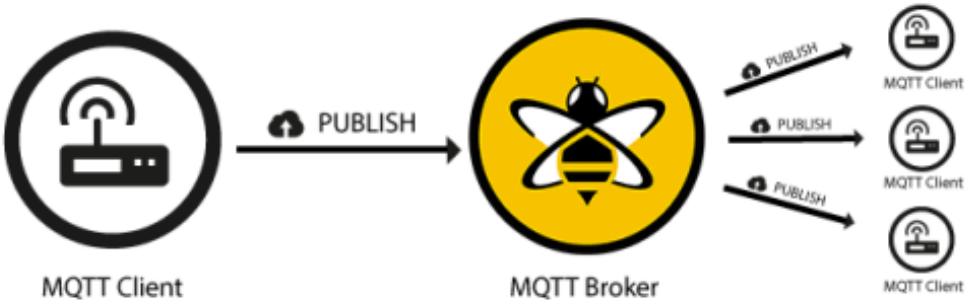


Figure 4 Publish communication,
src : https://www.hivemq.com/downloads/hivemq-ebook-mqtt-essentials.pdf?_hsmi=278147580

Figure 5 PUBLISH Message
src : « https://www.hivemq.com/downloads/hivemq-ebook-mqtt-essentials.pdf?_hsmi=278147580 »

5.2 Subscribe

Ce chapitre explore le processus de souscription dans MQTT, mettant en lumière des éléments clés tels que l'identifiant de paquet et la liste des abonnements.

L'identifiant de paquetest un élément essentiel, identifiant de manière unique chaque message circulant entre le client et le courtier. Cet identifiant MQTT interne est défini par la bibliothèque client et/ou le courtier.

La liste des abonnements dans un message de type SUBSCRIBE peut regrouper plusieurs abonnements pour un client. Chaque abonnement est constitué d'un sujet et d'un niveau de service de qualité (QoS). Le sujet dans le message de souscription peut inclure des caractères génériques, offrant la possibilité de s'abonner à un motif de sujet plutôt qu'à un sujet spécifique. En cas de chevauchement d'abonnements pour un client, le courtier délivre le message ayant le niveau de QoS le plus élevé pour ce sujet.

Ce processus de souscription permet aux clients de définir avec précision les sujets qui les intéressent et de spécifier le niveau de qualité de service souhaité pour chaque abonnement.

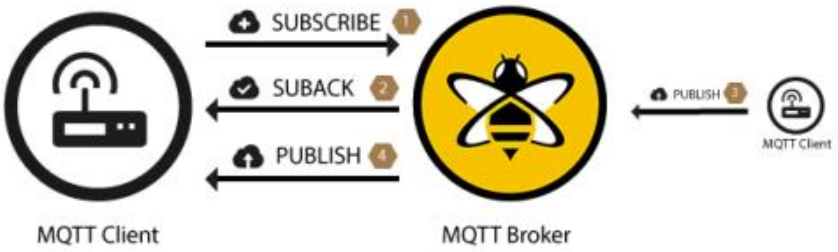


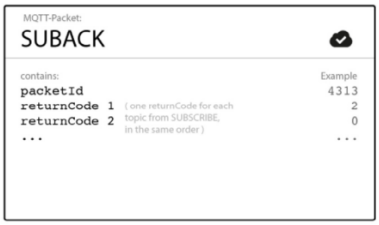
Figure 6 Subscribe communication
src : https://www.hivemq.com/downloads/hivemq-ebook-mqtt-essentials.pdf?_hsmi=278147580

Le SUBACK est un message de confirmation de souscription essentiel dans MQTT, comprenant des éléments clés tels que l'identifiant de paquet et les codes de retour.

L'identifiant de paquet dans le message SUBACK est identique à celui du message SUBSCRIBE. Il joue un rôle crucial dans le suivi des messages entre le client et le courtier.

Le code de retourconstitue une partie intégrante du SUBACK, où le courtier envoie un code de retour pour chaque paire sujet/QoS reçue dans le message SUBSCRIBE. Si le message SUBSCRIBE comporte cinq abonnements, le message SUBACK contient cinq codes de retour. Ces codes de retour confirment chaque sujet et indiquent le niveau de qualité de service (QoS) accordé par le courtier. En cas de refus d'un abonnement, le SUBACK contient un code d'erreur spécifique pour ce sujet particulier. Par exemple, si le client n'a pas les autorisations nécessaires ou si le sujet est mal formé, le SUBACK affiche un code d'erreur correspondant.

Ainsi, le SUBACK joue un rôle crucial dans le processus de souscription en fournissant des confirmations spécifiques sur chaque abonnement demandé par le client.



Return Code	Return Code Response
0	Success - Maximum QoS 0
1	Success - Maximum QoS 1
2	Success - Maximum QoS 2
128	Failure

Figure 7 Suback message
src : https://www.hivemq.com/downloads/hivemq-ebook-mqtt-essentials.pdf?_hsmi=278147580"

5.3 Unsubscribe (UNSUBACK)

L'UNSUBACK est un message essentiel dans MQTT qui confirme la réception du message de désabonnement (UNSUBSCRIBE) et joue un rôle crucial dans le processus de gestion des abonnements.

L'identifiant de paquet dans le message UNSUBACK est identique à celui du message UNSUBSCRIBE, permettant une identification unique du message tout au long de son parcours entre le client et le courtier. Après réception de l'UNSUBACK en provenance du courtier, le client peut en déduire que les désabonnements spécifiés dans le message UNSUBSCRIBE ont été effectués avec succès.

En résumé, l'UNSUBACK offre une confirmation fiable au client, assurant ainsi une gestion transparente et précise du processus de désabonnement dans le protocole MQTT.

Figure 8 Unsubscribe,
src : https://www.hivemq.com/downloads/hivemq-ebook-mqtt-essentials.pdf?_hsmi=278147580

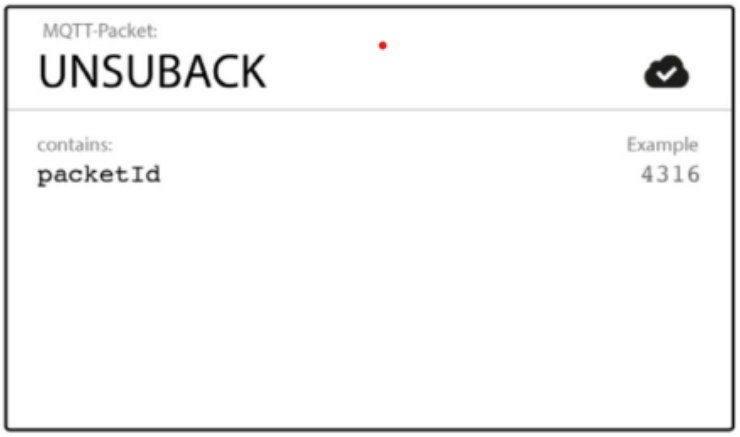


Figure 9 Unsuback,
src : : https://www.hivemq.com/downloads/hivemq-ebook-mqtt-essentials.pdf?_hsmi=278147580

6 Niveau de Qualité de Service (QoS)

Les niveaux de Qualité de Service (QoS) dans MQTT établissent un accord entre l'émetteur et le récepteur concernant la livraison des messages. Il existe trois niveaux de QoS dans MQTT, chacun offrant un équilibre entre la fiabilité de la livraison et l'efficacité du réseau.

1. QoS 0- Au plus une fois (0) :

- Niveau minimal avec une livraison « au mieux ».

- Aucune garantie, aucun accusé de réception, et aucun stockage de message.

- Souvent appelé "fire and forget", ce niveau privilégie l'efficacité à la garantie de livraison.

2. QoS 1- Au moins une fois (1) :

- Garantit la livraison du message au moins une fois.

- L'émetteur stocke le message jusqu'à réception d'un accusé de réception (PUBACK) du récepteur.

- Possibilité d'envoi ou de livraison multiple de messages.

- Utilise un identifiant de paquet pour faire correspondre les paquets PUBLISH et PUBACK.

3. QoS 2- Exactement une fois (2) :

- Niveau le plus élevé, garantit que chaque message est reçu exactement une fois.

- Assure la sécurité grâce à une poignée de main à quatre parties entre l'émetteur et le récepteur.

- Le niveau le plus lent, mais le plus sécurisé en termes de qualité de service.

La définition des niveaux de QoS se fait lors de l'envoi d'un message par le client émetteur et lors du processus d'abonnement pour les clients abonnés. Le courtier transmet les messages avec le niveau de QoS défini par chaque client abonné, et en cas de disparité entre les niveaux, le courtier utilise le niveau le plus bas pour assurer la cohérence du service.

Figure 10 QoS 0
src : « https://www.hivemq.com/downloads/hivemq-ebook-mqtt-essentials.pdf?_hsmi=278147580 »

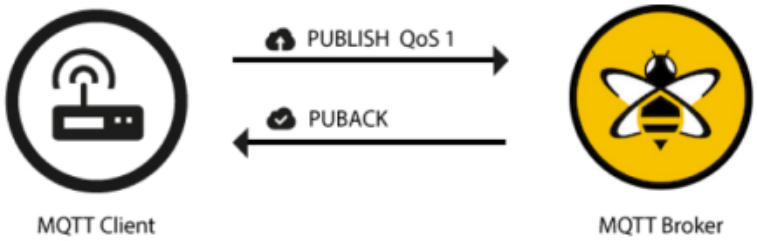


Figure 11 QoS1
src : « https://www.hivemq.com/downloads/hivemq-ebook-mqtt-essentials.pdf?_hsmi=278147580 »

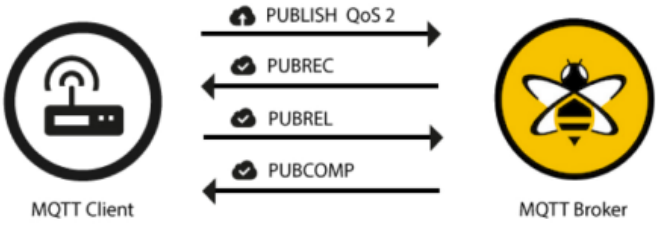


Figure 12 QoS2
src : « https://www.hivemq.com/downloads/hivemq-ebook-mqtt-essentials.pdf?_hsmi=278147580 »

Lorsque des niveaux de QoS plus élevés (1 et 2) sont utilisés dans MQTT, des paquets supplémentaires sont introduits pour garantir la livraison exacte des messages. Chaque paquet joue un rôle spécifique dans le processus, assurant une communication fiable entre l'émetteur et le récepteur.

1. Paquet PUBREC :

- Accusé de réception du paquet PUBLISH par le récepteur.

- En l'absence de PUBREC, l'émetteur renvoie le paquet PUBLISH avec le drapeau DUP jusqu'à réception de l'accusé de réception.

2. Paquet PUBREL :

- À la réception de PUBREC, l'émetteur ignore le paquet PUBLISH initial.

- Répond au récepteur avec un paquet PUBREL.

3. Paquet PUBCOMP :

- À la réception de PUBREL, le récepteur ignore tous les états stockés.

- Répond avec un paquet PUBCOMP.

- Étape cruciale pour éviter la retraitement des messages.

- Après la réception de PUBCOMP, l'identifiant de paquet du message publié est disponible pour une réutilisation ultérieure.

Le choix du niveau de QoS dépend du contexte d'utilisation :

Utiliser QoS 0 lorsque :

- La connexion entre l'émetteur et le récepteur est stable.

- La perte occasionnelle de quelques messages est acceptable.

- Aucune nécessité de mise en file d'attente des messages.

Utiliser QoS 1 lorsque :

- Chaque message doit être reçu, et les doublons peuvent être gérés.
- Le surcoût de QoS 2 est insupportable.
- Une livraison plus rapide des messages est essentielle.

Utiliser QoS 2 lorsque :

- Il est critique de recevoir tous les messages exactement une fois.
- La livraison en double peut nuire aux utilisateurs ou aux clients abonnés.
- Conscience du surcoût et volonté de tolérer le temps supplémentaire pour l'interaction QoS 2.

Les messages envoyés avec QoS 1 et 2 sont mis en file d'attente pour les clients hors ligne, avec la condition que le client dispose d'une session persistante.

7 Sessions persistantes et mise en file d'attente des messages

Overview

Pour recevoir des messages, un client se connecte au courtier et s'abonne à des sujets. Les sessions non persistantes perdent les sujets lors de coupures de connexion, nécessitant une nouvelle souscription. Les sessions persistantes stockent les informations pertinentes du client sur le courtier, identifiées par le clientId.

Informations Stockées dans une Session Persistante :

Existence de la session.

Abonnements du client.

Messages QoS 1 ou 2 non confirmés.

Nouveaux messages QoS 1 ou 2 manqués pendant une déconnexion.

Messages QoS 2 non confirmés par le client.

Initialisation/Fin d'une Session Persistante :

Le drapeau CleanSession dans la communication client-broker :

True : Aucune session persistante ; perte d'informations lors de la déconnexion.

False : Session persistante ; conserve les informations jusqu'à ce que le client demande une session propre.

Notification de la Présence de Session :

Le message CONNACK contient un drapeau de présence de session pour informer le client de l'existence d'une session. Aide le client à savoir si une session précédemment établie est disponible sur le courtier.

Côté Client de la Session Persistante :

Chaque client MQTT stocke une session persistante. Responsable du stockage des messages QoS 1 ou 2 non confirmés et des messages QoS 2 non confirmés.

Meilleures Pratiques pour le Choix de la Session :

Session Persistante :

Le client a besoin de tous les messages d'un sujet spécifique, même s'il est hors ligne.

Ressources limitées du client ; une restauration rapide de la communication est essentielle.

Le client doit reprendre tous les messages de publication QoS 1 et 2 après une reconnexion.

Session Propre :

Le client publie uniquement des messages ; pas besoin de s'abonner à des sujets.

Aucun stockage d'informations de session ou de nouvelle tentative de transmission de messages QoS 1 et 2.

Le client n'a pas besoin de messages manqués hors ligne.

8 Retained Messages

Overview

Les éditeurs n'ont aucune garantie de réception des messages par les abonnés. L'assurance de la livraison des messages se limite à une livraison sûre au courtier. Les clients abonnés n'ont aucune garantie de savoir quand un éditeur enverra un nouveau message.

Concept des Messages Conservés :

Les messages conservés répondent à l'incertitude quant au moment de la livraison des messages. Un message conservé est un message MQTT standard avec le drapeau conservé activé. Le courtier stocke le dernier message conservé et son QoS correspondant pour chaque sujet.

Scénario d'Abonnement :

Les abonnés reçoivent le dernier message conservé immédiatement lors de leur abonnement. Le courtier stocke uniquement un message conservé par sujet. Si des caractères génériques sont utilisés dans le modèle d'abonnement, même les correspondances non exactes déclenchent la livraison du message conservé.

Scénario Exemple :

Client A publie un message conservé sur myhome/livingroom/temperature. Client B s'abonne à myhome/#. Client B reçoit le message conservé myhome/livingroom/temperature lors de son abonnement.

Objectif et Cas d'Utilisation :

Les messages conservés sont bénéfiques lorsqu'il s'agit de :

- Fournir des mises à jour d'état aux abonnés nouvellement connectés.

- Transmettre des données telles que la température ou les coordonnées GPS.

- Offrir immédiatement la dernière valeur connue aux clients en cours de connexion.

Envoi et Suppression de Messages Conservés :

L'envoi d'un message conservé est simple : activez le drapeau conservé à true dans le message MQTT à publier. La suppression d'un message conservé implique l'envoi d'un message conservé avec une

charge utile de zéro octet sur le sujet spécifique. Le courtier supprime le message conservé, garantissant que les nouveaux abonnés ne le reçoivent pas.

9 Keep Alive and Client Take-Over

Vue d'ensemble sur le Maintien de la Connexion :

Le Maintien de la Connexion dans MQTT aborde les défis des connexions à moitié ouverte. Il garantit une conscience continue de l'état de la connexion entre le courtier et le client. Lorsqu'un client se connecte, il communique un intervalle de temps (l'intervalle de maintien) au courtier, définissant le temps maximum de non-communication.

Mécanisme de Maintien de la Connexion :

Le Maintien de la Connexion est le temps maximum entre la transmission de deux paquets de contrôle. Le client doit garantir que l'intervalle entre les paquets de contrôle ne dépasse pas la valeur de Maintien de la Connexion. En l'absence d'autres paquets de contrôle, le client doit envoyer un paquet PINGREQ.

Déroulement du Maintien de la Connexion :

Tant que des messages sont échangés fréquemment dans l'intervalle de maintien, aucune vérification supplémentaire n'est nécessaire. Si aucun message n'est envoyé pendant la période de maintien, le client envoie un PINGREQ pour confirmer sa disponibilité. Le courtier déconnecte le client s'il ne reçoit aucun message ou PINGREQ dans 1,5 fois l'intervalle de maintien.

Paquets du Maintien de la Connexion :

- Paquet PINGREQ : Envoyé par le client pour indiquer sa présence ; ne contient aucune charge utile.

- Paquet PINGRESP : Le courtier répond avec ce paquet pour confirmer sa disponibilité ; ne contient également aucune charge utile.

Points Importants :

- Le courtier ferme la connexion s'il ne reçoit aucun PINGREQ ou paquet.

- Le client MQTT définit une valeur de maintien de la connexion appropriée ; ajuste l'intervalle en fonction de la force du signal.

- Le maintien de la connexion maximum est de 18 heures 12 minutes 15 secondes ; un intervalle de maintien de 0 désactive le mécanisme.

Reprise de Client dans MQTT :

Les clients déconnectés tentent généralement de se reconnecter. En cas de connexion à moitié ouverte, le courtier lance une 'reprise de client.' La connexion précédente est fermée, et une nouvelle connexion est établie avec le client, évitant les obstacles à la rétablissement de la connexion.

10 MQTT Over Websockets

Vue d'ensemble :

MQTT est adapté aux dispositifs contraints et aux réseaux peu fiables, idéal pour une transmission de messages à faible surcharge. L'intégration de MQTT dans les navigateurs, par exemple sur les téléphones mobiles, est réalisée grâce à MQTT sur WebSockets.

Cas d'utilisation de MQTT sur WebSockets :

Permet aux navigateurs d'utiliser directement les fonctionnalités MQTT, avec des cas d'utilisation tels que l'affichage d'informations en direct à partir de dispositifs ou de capteurs, la réception de notifications push (alertes, avertissements de conditions critiques), la vérification de l'état actuel des dispositifs avec LWT et les messages conservés, et la communication efficace avec les applications Web mobiles.

Présentation Technique :

Les navigateurs modernes prenant en charge les WebSockets peuvent fonctionner comme des clients MQTT complets. Un bibliothèque JavaScript pour MQTT sur WebSockets et un courtier prenant en charge cette fonctionnalité sont requis. Le courtier HiveMQ prend en charge MQTT sur WebSockets dès sa configuration.

Protocole WebSocket et MQTT sur WebSockets :

WebSocket est un protocole de communication bidirectionnel entre les navigateurs et les serveurs Web, standardisé en 2011, pris en charge par les navigateurs modernes, basé sur TCP. Les messages MQTT sont encapsulés par des trames WebSocket pour une communication bidirectionnelle, ordonnée et sans perte.

WebSockets Sécurisés :

Utilisez la sécurité de la couche de transport (TLS) pour des WebSockets sécurisés. TLS garantit le chiffrement de l'ensemble de la connexion.

TLS, ou Transport Layer Security, est un protocole de sécurité qui assure la confidentialité et l'intégrité des données lors de la transmission sur un réseau. Il est utilisé pour sécuriser les communications sur Internet, notamment dans des applications telles que la navigation Web, le courrier électronique, la messagerie instantanée et d'autres protocoles de communication.

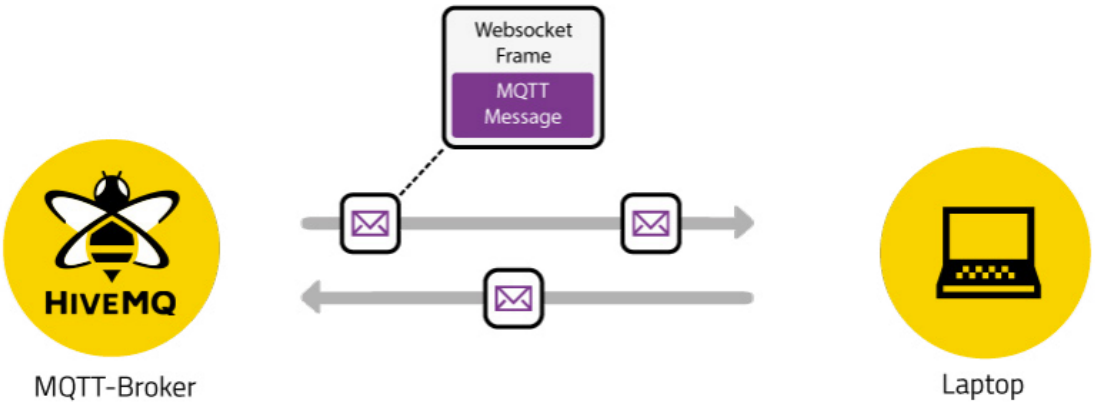


Figure 13 MQTT Over Websockets
src : « https://www.hivemq.com/downloads/hivemq-ebook-mqtt-essentials.pdf?_hsmi=278147580 »

Bibliographie

hivemq. (n.d.). *hivemq-ebook-mqtt-essentials.pdf.* Retrieved from https://www.hivemq.com/downloads/hivemq-ebook-mqtt-essentials.pdf?_hsmi=278147580