Persistent sessions in MQTT allow a client to maintain its subscription and message state across multiple connections. When a client establishes a persistent session with an MQTT broker, the broker stores the client's subscription information and any undelivered messages intended for the client. This way, if the client disconnects and reconnects later, it can resume communication seamlessly.

Here's Part 7 of **MQTT Essentials**, a ten-part blog series on the core features and concepts of the **MQTT protocol**, where we explore the topics of persistent sessions, clean sessions and message queueing in detail. We will dive deep into how persistent sessions in MQTT enhance QoS by enabling reliable message delivery, providing QoS level guarantees, and facilitating efficient reconnection for clients, even in the presence of intermittent connectivity or client disconnections.

If you are new to **MQTT Quality of Service (QoS) 0,1, & 2**, check out Part 6 of this series. Else, let's dive in.

What are Persistent Sessions?

To receive messages from an **MQTT broker**, a client establishes a connection and creates **subscriptions to the desired topics**. In a non-persistent session, if the connection between the client and broker is interrupted, the client loses its subscriptions and needs to re-subscribe upon reconnection. This can be burdensome for resource-constrained clients. **To address this issue, clients can request a persistent session when connecting to the broker.**

> Persistent sessions store all relevant client information on the broker, ensuring that subscriptions and messages are retained even when the client is offline. The session is identified by the clientId provided by the client during the connection establishment process. To learn more, read our article **MQTT Client, MQTT Broker, and MQTT Server Connection Establishment Explained.**

What's Stored in a Persistent Session?

In a persistent session, the broker stores the following information (even if the client is offline). This information becomes immediately available to the client upon reconnection:

- **Session existence (even if there are no subscriptions)**: The broker retains information about the existence of the session, allowing the client to resume its previous state upon reconnection.
- **All client's subscriptions**: The broker stores the list of topics to which the client has subscribed. This ensures the client does not need to re-subscribe to the same topics every time it reconnects, saving valuable time and resources.
- **Flow of all messages in a Quality of Service (QoS) 1 or 2 where the client has not yet confirmed**: The broker keeps track of unacknowledged messages sent to the client at QoS 1 or QoS 2 levels. These messages are stored in the

broker's message queue and will be delivered to the client upon reconnection, ensuring reliable message delivery.

- **All new QoS 1 or 2 messages that the client missed while offline**: If the client was offline when QoS 1 or QoS 2 messages were published to subscribed topics, the broker stores these missed messages. Once the client reconnects, it receives the queued messages, preventing any loss of important information.
- **All QoS 2 messages received from the client that are awaiting complete acknowledgment**: For QoS 2 messages sent by the client, the broker keeps track of their acknowledgment status. If any of these messages are not fully acknowledged, the broker holds them until the acknowledgment is complete.

What are Clean Sessions in MQTT and How to Use Them to Start or End a Persistent Session?

When establishing a connection to the broker, clients can enable or disable a persistent session by setting the value of the `cleanSession` flag. Here's how it works: **When the cleanSession flag is set to true**, the client explicitly requests a non-persistent session. In this scenario, if the client disconnects from the broker, all queued information and messages from the previous persistent session are discarded. The client starts with a clean slate upon reconnection.

> On the other hand, **when the cleanSession flag is set to false**, the broker creates a persistent session for the client. This means that the broker preserves all relevant information and messages even if the client goes offline. The session remains intact until the client explicitly requests a clean session. If the cleanSession flag is

set to false and the broker already has a session available for the client, it will utilize the existing session and deliver any previously queued messages to the client upon reconnection.

For more detailed information on establishing a connection between the client and broker, refer to our article, **MQTT Client, MQTT Broker, and MQTT Server Connection Establishment Explained.**

How Does A Client Know if a Session is Already Stored?

In MQTT version 3.1.1 and beyond, the CONNACK message sent by the broker in response to the client's connection request includes a session present flag. This flag serves as an indicator for the client, informing it whether a previously established session is still available on the broker. By examining the session present flag, the client can determine whether to establish a new session or reconnect to an existing one. To gain a comprehensive understanding of the connection establishment process, we recommend referring to **MQTT Essentials Part 3.**

Persistent Session on the Client Side: Ensuring Local Message Persistence and Acknowledgment

In addition to the broker storing a persistent session, each MQTT client also plays a role in maintaining session continuity. When a client requests the server to retain

a role in maintaining session continuity. When a client requests the server to retain session data, it assumes the responsibility and must store the following information:

- **Flow of all messages in a QoS 1 or 2 where the broker has not yet confirmed**: The client keeps track of messages it has sent to the broker at QoS 1 or QoS 2 levels. These messages are stored locally until the broker acknowledges their receipt or completion. By maintaining these unconfirmed messages, the client ensures that it can retransmit any messages if necessary and achieve the desired level of reliability.

- **All QoS 2 messages received from the broker awaiting complete acknowledgment**: When the broker sends QoS 2 messages to the client, they are received and processed by the client. However, until the client acknowledges the successful processing of these messages, the client stores them locally. This ensures that the client can handle any interruptions or disconnections while maintaining the reliability and integrity of message delivery.

- By storing these message-related details on the client side, MQTT clients can actively maintain session persistence and ensure the successful processing of messages even in challenging network conditions.

MQTT Session Management Best Practices: Enhancing Message Delivery and Resource Efficiency

When working with MQTT, it's essential to consider the best practices for session management to optimize your implementation. Here are some guidelines to help you determine when to use a persistent session or a clean session:

Best Practices for MQTT Persistent Sessions

- **Ensure Message Reliability**: If your client needs to receive all messages from a specific topic, even when they are offline, a persistent session is the way to go. This ensures that the broker queues the messages for the client and delivers them promptly when the client reconnects.
- **Resource Optimization**: If your client has limited resources, leveraging a persistent session is beneficial. Storing the client's subscription information on the broker facilitates a quick restoration of interrupted communication, reducing the burden on constrained clients.
- **Resume Publishes**: A persistent session is necessary if your client needs to resume publishing Quality of Service (QoS) 1 and 2 messages after reconnecting. The broker retains these messages until the client returns online, ensuring their reliable delivery.

Best Practices for MQTT Clean Session

- **Publish-Only Clients**: A clean session is suitable if your client only needs to publish messages and does not require subscriptions to topics. In such cases, the broker does not need to store session information or attempt to transmit QoS 1 and 2 messages, simplifying the session management process.
- **Avoid Offline Message Retrieval**: A clean session suffices if your client does not need to receive missed messages while offline. It eliminates the overhead of storing and delivering messages that the client did not subscribe to during its offline period.