Best Practices Integrating MQTT

Guides to best practices for implementing MQTT based solutions using the Reekoh integration platform and components.

REEKOH
The Industrial IoT Integration Cloud

November 2020

MQTT Capabilities in Reekoh

MQTT has become a de-facto standard protocol for many IoT and Industrial IoT solutions. When working with MQTT capabilities in Reekoh, we have devised a series of best practices that you can follow to best handle your data and scaling requirements.

Reekoh Plugins and Modules that enabled MQTT

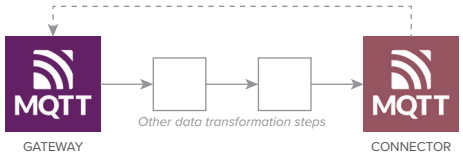
Gateway Plugin	MQTT Gateway is a hosted broker facilitating rapid data ingest for an integration solution
Stream Plugin	MQTT Stream is an MQTT client that subscribes to an external broker or the Standalone Cloud Broker. Achieves scale through pipeline design
Connector Plugin	MQTT Connector is an MQTT client optimised for high-volume data egress
Channel Plugin	MQTT Channel is a hosted broker facilitating lower-volume data egress to subscribing clients
Standalone Cloud Broker Module	The standalone Reekoh Cloud Broker is a high-volume, clustered MQTT broker, which can be deployed purely to facilitate site to site connectivity or permitting selected data to be emitted into an integration pipeline

Reekoh MQTT Plugin and Module Attributes

	Gateway Plugin	Stream Plugin	Connector Plugin	Channel Plugin	Standalone Broker
Data Ingress	✓	✓	●	●	✓
Data Egress	●	●	✓	✓	✓
Auto-Scale	✓	●	✓	●	✓
MQTT Behaviour	Broker	Client	Client	Broker	Broker
Suitable Volumes	High	Low	High	Low	ALL
Reekoh Usage	Pipeline	Pipeline	Pipeline	Pipeline	Module

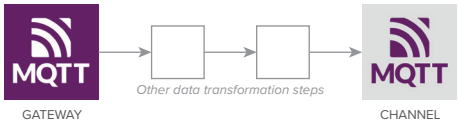
MQTT Request/Response Patterns

Gateway to Connector



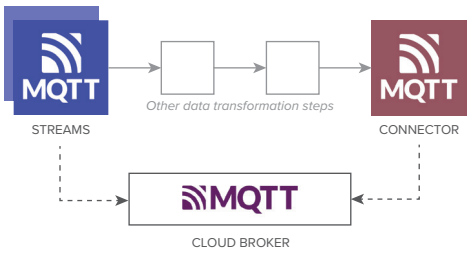
This is a naive implementation of a request response pattern over MQTT. Data is published to a Topic on the Gateway, and the Connector is publishing to other Topics on the Gateway, to which the consumer is subscribing. Auto-scaling on the MQTT Gateway can mean that some messages may be published on different replicas and therefore may not be able to be reliably subscribed to.

Gateway to Channel



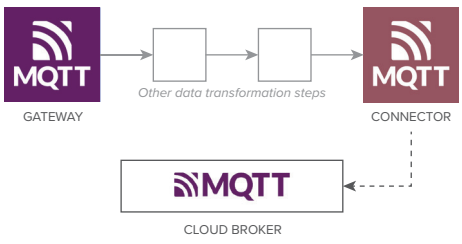
This approach permits a request/response pattern through separating data ingress from data egress, allowing the use of optimisations for each stage of the pipeline. Best suited to patterns where outbound only communication is desired and lower volumes are required (e.g. bridging OT systems to IT applications)
The lack of auto-scaling support on the Channel, which is required to permit reliable subscription, limits applications to those with lower volumes.

Topic Segregated Flow



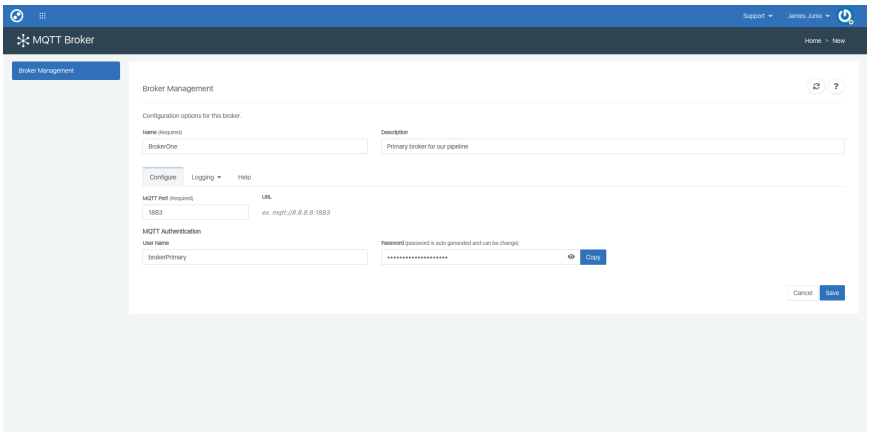
This approach permits a high volume request response pattern, but only if topic segregation is possible (for example with Sparkplug B). It permits the use of a single MQTT endpoint for both request and response.

Gateway to Cloud Broker



This approach permits a high volume request response pattern, but requires the use of two different MQTT endpoints. Very similar to Gateway to Channel approach, except facilitates high volume egress as well.

Integration-less MQTT with Standalone Cloud Broker Module



The standalone MQTT Cloud Broker module can be used to create high-volume, clustered MQTT brokers from within the Reekoh platform This broker can be used on its own, and not tied to an integration pipeline, for purposes where existing applications may require a broker to communicate directly with each other. The broker can also be used to emit selected data to an integration pipeline

Using Sparkplug

Standalone MQTT Cloud Broker can be used for all Sparkplug interactions. Gateways permit registration of multiple topics through wild-carding as data ingress. Streams can subscribe to subsets of data only.

It's planned that Sparkplug B Birth/Death Certificates will integrate with the Reekoh Device Registry module, and that Payload Definitions can be loaded as Data Schemas and used within the Reekoh Data Management module.

Sparkplug
MQTT Topic & Payload Definition