OMRON

Fleet Operations Workspace Core Integration Toolkit

MQTT API

User's Manual

M107-E-01

NOTE

Trademarks

Copyrights

Introduction

This document contains information that is necessary to use the MQTT API facilitating integration between the AMR, Fleet Manager, and the end user's client application.
This manual is OMRON's original instructions describing the MQTT API functionality provided with the Fleet Operations Workspace (FLOW) Core v4.1.0 software suite.
Please read this manual and make sure you understand the functionality and performance of the MQTT API before you attempt to use it with a fleet of AMRs. Read and understand all related manuals and safety guides before using the MQTT API.

Intended Audience

This document is intended for the following personnel.

- Personnel integrating the Omron AMR solution with manufacturing execution systems (MES), enterprise resource planning (ERP) solutions or other similar systems.
- Personnel familiar with Omron's fleet management software, AMRs, and the EM2100 appliance.
- Personnel familiar with the Advanced Robotics Command Language (ARCL), MQTT protocol, or RESTful Web Services.

System Requirements

The MQTT API has the following minimum system requirements.

- Fleet Manager device with Fleet Operations Workspace (FLOW) Core software version 4.1
- OMRON AMR with FLOW Core software 4.1
- Virtual Fleet Manager with FLOW Core software 4.1

Notations

Programming code and syntax examples are used throughout this document. This text will be indicated with the font shown below to distinguish it from other non-code text.

```
{
  "default_priority": false,
  "details": [
    {
```

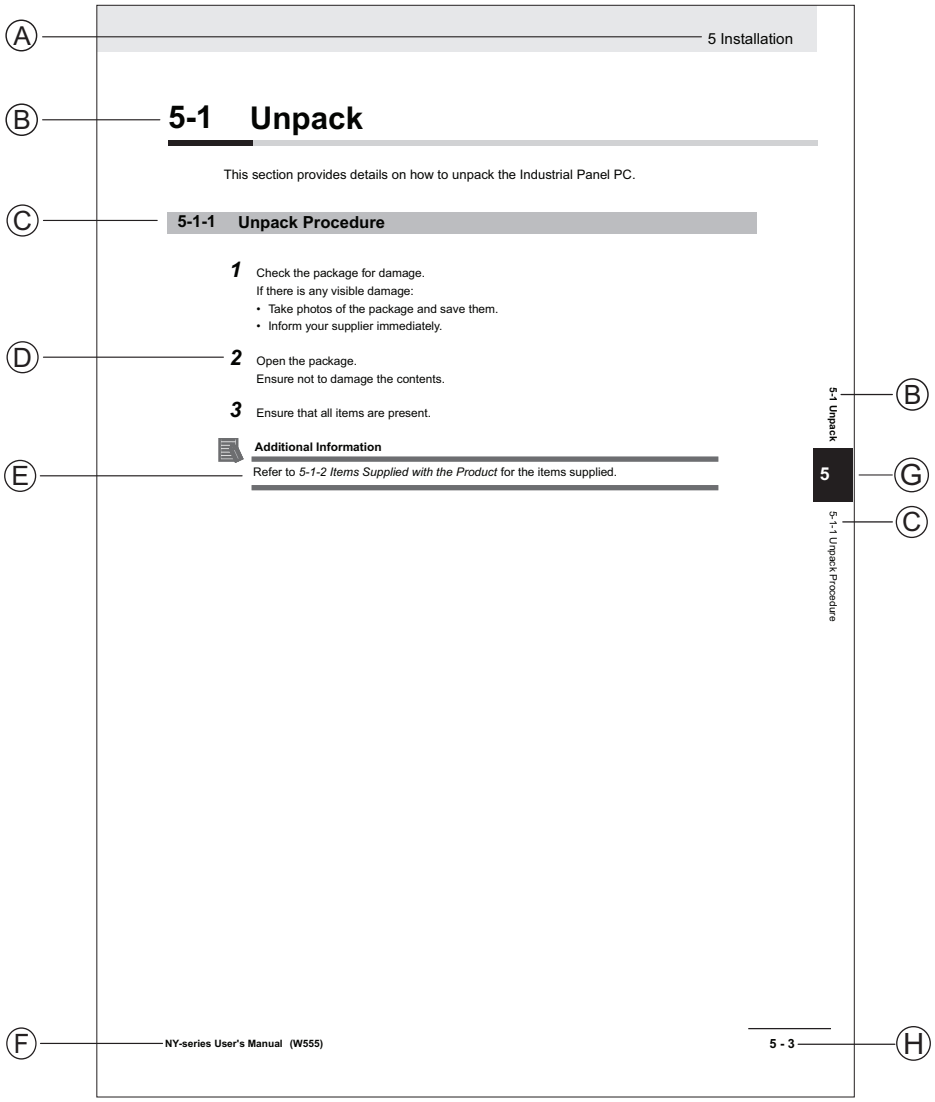
MQTT Explorer Copyright, License Information, and Disclaimer of Warranties

MQTT Explorer (version 0.4.0-beta1) images are reproduced subject to the Creative Commons Attribution-NoDerivatives 4.0 International open source, public license. https://github.com/thomasnordquist/MQTT-Explorer/blob/master/LICENSE.md
The Licensed Material is offered by the Licensor as-is and as-available, subject to the Disclaimer of Warranties and Limitation of Liability set forth in Section 5 of the linked Creative Commons license.
The licensed material has not been modified in this document.
MQTT Explorer is avilable at: https://gihttps//github.com/thomasnordquist/MQTT-Explorer

Manual Information

Page Structure

The following page structure is used in this manual.



Note: This illustration is provided as a sample. It will not literally appear in this manual.

Item	Explanation	Item	Explanation
A	Level 1 heading	E	Special Information
B	Level 2 heading	F	Manual name
C	Level 3 heading	G	Page tab with the number of the main section
D	Step in a procedure	H	Page number

Special Information

Special information in this manual is classified as follows:

Precautions for Safe Use

Precautions on what to do and what not to do to ensure safe usage of the product.

Precautions for Correct Use

Precautions on what to do and what not to do to ensure proper operation and performance.

Additional Information

Additional information to read as required.
This information is provided to increase understanding or make operation easier.

Version Information

Information on differences in specifications and functionality between different versions.

Sections in this Manual

CONTENTS

Section 1 Overview

Section 2 Setting Up and Connecting

Terms and Conditions Agreement

Warranty and Limitations of Liability

Warranty

- **Exclusive Warranty**
  Omron's exclusive warranty is that the Products will be free from defects in materials and workmanship for a period of twelve months from the date of sale by Omron (or such other period expressed in writing by Omron). Omron disclaims all other warranties, expressed or implied.
- **Limitations**
  OMRON MAKES NO WARRANTY OR REPRESENTATION, EXPRESS OR IMPLIED, ABOUT NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE OF THE PRODUCTS. BUYER ACKNOWLEDGES THAT IT ALONE HAS DETERMINED THAT THE PRODUCTS WILL SUITABLY MEET THE REQUIREMENTS OF THEIR INTENDED USE.
  Omron further disclaims all warranties and responsibility of any type for claims or expenses based on infringement by the Products or otherwise of any intellectual property right.
- **Buyer Remedy**
  Omron's sole obligation hereunder shall be, at Omron's election, to (i) replace (in the form originally shipped with Buyer responsible for labor charges for removal or replacement thereof) the non-complying Product, (ii) repair the non-complying Product, or (iii) repay or credit Buyer an amount equal to the purchase price of the non-complying Product; provided that in no event shall Omron be responsible for warranty, repair, indemnity or any other claims or expenses regarding the Products unless Omron's analysis confirms that the Products were properly handled, stored, installed and maintained and not subject to contamination, abuse, misuse or inappropriate modification. Return of any Products by Buyer must be approved in writing by Omron before shipment. Omron Companies shall not be liable for the suitability or unsuitability or the results from the use of Products in combination with any electrical or electronic components, circuits, system assemblies or any other materials or substances or environments. Any advice, recommendations or information given orally or in writing, are not to be construed as an amendment or addition to the above warranty.

See http://www.omron.com/global/ or contact your Omron representative for published information.

Limitations of Liability

OMRON COMPANIES SHALL NOT BE LIABLE FOR SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, LOSS OF PROFITS OR PRODUCTION OR COMMERCIAL LOSS IN ANY WAY CONNECTED WITH THE PRODUCTS, WHETHER SUCH CLAIM IS BASED IN CONTRACT, WARRANTY, NEGLIGENCE OR STRICT LIABILITY. Further, in no event shall liability of Omron Companies exceed the individual price of the Product on which liability is asserted.

Application Considerations

Suitability for Use

Omron Companies shall not be responsible for conformity with any standards, codes or regulations which apply to the combination of the Product in the Buyer's application or use of the Product. At Buyer's request, Omron will provide applicable third party certification documents identifying ratings and limitations of use which apply to the Product. This information by itself is not sufficient for a complete determination of the suitability of the Product in combination with the end product, machine, system, or other application or use. Buyer shall be solely responsible for determining appropriateness of the particular Product with respect to Buyer's application, product or system. Buyer shall take application responsibility in all cases.
NEVER USE THE PRODUCT FOR AN APPLICATION INVOLVING SERIOUS RISK TO LIFE OR PROPERTY WITHOUT ENSURING THAT THE SYSTEM AS A WHOLE HAS BEEN DESIGNED TO ADDRESS THE RISKS, AND THAT THE OMRON PRODUCT(S) IS PROPERLY RATED AND INSTALLED FOR THE INTENDED USE WITHIN THE OVERALL EQUIPMENT OR SYSTEM.

Programmable Products

- Omron Companies shall not be responsible for the user's programming of a programmable Product, or any consequence thereof.
- Omron Companies shall not be responsible for the operation of the user accessible operating system (e.g. Windows, Linux), or any consequence thereof.

Disclaimers

Performance Data

Data presented in Omron Company websites, catalogs and other materials is provided as a guide for the user in determining suitability and does not constitute a warranty. It may represent the result of Omron's test conditions, and the user must correlate it to actual application requirements. Actual performance is subject to the Omron's Warranty and Limitations of Liability.

Change in Specifications

Product specifications and accessories may be changed at any time based on improvements and other reasons. It is our practice to change part numbers when published ratings or features are changed, or when significant construction changes are made. However, some specifications of the Product may be changed without any notice. When in doubt, special part numbers may be assigned to fix or establish key specifications for your application. Please consult with your Omron's representative at any time to confirm actual specifications of purchased Product.

Errors and Omissions

Information presented by Omron Companies has been checked and is believed to be accurate; however, no responsibility is assumed for clerical, typographical or proofreading errors or omissions.

Safety Precautions

Definition of Precautionary Information

The following notation is used in this manual to provide precautions required to ensure safe usage of the product. The safety precautions that are provided are extremely important to safety.
Always read and heed the information provided in all safety precautions.
The following notation is used.

⚠ **DANGER**	Identifies an imminently hazardous situation which, if not avoided, is likely to result in serious injury, and might result in fatality or severe property damage.
⚠ **WARNING**	Indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury. Additionally, there may be severe property damage.
⚠ **CAUTION**	Indicates a potentially hazardous situation which, if not avoided, may result in minor or moderate injury, or property damage.

Symbols

	The circle and slash symbol indicates operations that you must not do. The specific operation is shown in the circle and explained in text. This example indicates prohibiting disassembly.
	The triangle symbol indicates precautions (including warnings). The specific operation is shown in the triangle and explained in text. This example indicates a precaution for electric shock.
	The triangle symbol indicates precautions (including warnings). The specific operation is shown in the triangle and explained in text. This example indicates a general precaution.
	The filled circle symbol indicates operations that you must do. The specific operation is shown in the circle and explained in text. This example shows a general precaution for something that you must do.
	The triangle symbol indicates precautions (including warnings). The specific operation is shown in the triangle and explained in text. This example indicates a precaution for high temperatures.

Warnings

⚠ WARNING

Cybersecurity

To maintain the security and reliability of the system, a robust cybersecurity defense program should be implemented, which may include some or all of the following:

Anti-virus protection

- Install the latest commercial-quality anti-virus software on the computer connected to the control system and keep the software and virus definitions up-to-date.
- Scan USB drives or other external storage devices before connecting them to control systems and equipment.

Security measures to prevent unauthorized network access

- Install physical controls so that only authorized personnel can access control systems and equipment.
- Reduce connections to control systems and equipment via networks to prevent access from untrusted devices.
- Install firewalls to block unused communications ports and limit communication between systems. Limit access between control systems and systems from the IT network.
- Control remote access and adopt multifactor authentication to devices with remote access to control systems and equipment.
- Set strong password policies and monitor for compliance frequently.

Data input and output protection

- Backup data and keep the data up-to-date periodically to prepare for data loss.
- Validate backups and retention policies to cope with unintentional modification of input/output data to control systems and equipment.
- Validate the scope of data protection regularly to accommodate changes.
- Check validity of backups by scheduling test restores to ensure successful recovery from incidents.
- Safety design, such as emergency shutdown and fail-soft operations in case of data tampering and incidents.

Additional recommendations

- When using an external network environment to connect to an unauthorized terminal such as a SCADA, HMI or to an unauthorized server may result in network security issues such as spoofing and tampering.
- You must take sufficient measures such as restricting access to the terminal, using a terminal equipped with a secure function, and locking the installation area by yourself.
- When constructing network infrastructure, communication failure may occur due to cable disconnection or the influence of unauthorized network equipment.
- Take adequate measures, such as restricting physical access to network devices, by means such as locking the installation area.
- When using devices equipped with an SD Memory Card, there is a security risk that a third party may acquire, alter, or replace the files and data in the removable media by removing or unmounting the media.
- Please take sufficient measures, such as restricting physical access to the Controller or taking appropriate management measures for removable media, by means of locking and controlling access to the installation area.
- Educate employees to help them identify phishing scams received via email on systems that will connect to the control network.

Related Manuals

Use the following related manuals for reference.

Manual Title	Description
AMR User's Manual(s)	Describes the installation, start-up, operation, and maintenance of the AMR.
Fleet Operations Workspace Core User's Manual (Cat. No. I635)	Describes Fleet management, MobilePlanner software, the SetNetGo OS, and most of the configuration procedures for an AMR.
Fleet Operation Workspace Core Integration Toolkit User’s Manual (Cat. No. I637)	Describes the features and functionality of the Integration Toolkit interface application with REST, SQL, and RabbitMQ communication channels.
Advanced Robotics Command Language AMR Reference Guide (Cat. No. I617)	Describes how to use the Advanced Robotics Command Language (ARCL), a text-based command line operating language. Use ARCL to integrate a fleet of AMRs with an external automation system.
Enterprise Manager 2100 User's Guide (Cat. No. I631)	Describes the installation of an EM2100 appliance, which runs the Fleet Operations Workspace software to manage a fleet of AMRs.

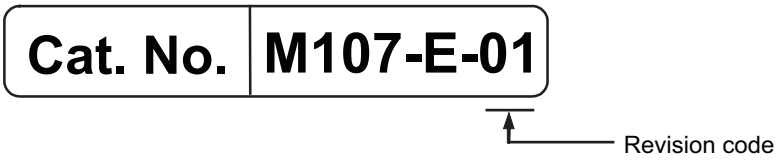
Glossary

Term / Abbreviation	Description
AMR	Autonomous Mobile Robot.
API	Application Programming Interface
ARCL	Advanced Robotics Command Language.
Client Application	A warehouse management system, manufacturing execution system, enterprise resource planning system, or similar application that interacts with the MQTT API.
Fleet Manager Device / EM2100 Appliance	Hardware appliance which connects all Omron AMRs and runs the fleet management software.
FA	Factory Automation
Fleet	Two or more AMRs operating in the same workspace controlled by a single Fleet Manager.
Fleet Manager	The operational mode of the computing appliance (Fleet Manger Device) that runs the FLOW Core software to control a fleet of AMRs.
Virtual Fleet Manager	The operational mode of the computing appliance that runs the FLOW Core software to control a fleet of AMRs when hosted by a hypervisor.
Fleet Operations Workspace (FLOW) Core software	Omron's software suite that is used to set up, integrate, and manage a fleet of AMRs within a factory environment.
Goal	A map-defined virtual destination for mobile robots (e.g., pickup or dropoff points).
Job	An activity typically consisting of one or two segments that instruct the AMR to drive to a goal for material pickup or dropoff.
JSON	JavaScript Object Notation is a lightweight data interchange format for storing and transporting data.
MobilePlanner	The primary software application for programming AMR actions. It provides the tools for all major AMR activities, such as observing a fleet of AMRs, commanding individual AMRs to drive, creating and editing map files, goals, and tasks, and modifying AMR configurations. MobilePlanner is part of the FLOW software suite.
MQTT	Message Queuing Telemetry Transport. It is an open source lightweight, publish-subscribe network protocol.
Payload	API/software context - the data that is sent in a request or received in a response. Hardware context - Material that is picked up or dropped off by an AMR.
Tasks	Instructions for the AMR to perform certain actions like reading inputs, setting outputs, movement commands, talking, waiting, and other functions.
DataStore Value	Data received at a specific interval about an AMR or Fleet Manager.
QoS	Quality of Service (QoS) determines the level of assurance for message delivery between the client and the broker. There are three levels of QoS: 0, 1, and 2. Higher levels of QoS are more reliable but with higher impact on the network.
Inflight Messages	The messages that are in the process of being transmitted simultaneously. Inflight messages are sent messages but unacknowledged by the broker as yet. The maximum number of inflight messages allowed by the MQTT broker is 20. The limit applies to messages with QoS 1 or 2.

Term / Abbreviation	Description
Message Queue	In-memory queue that holds inflight messages and subsequent incoming messages (when inflight maximum is reached). The maximum number of queued messages allowed by the broker is 1000. The limit applies to messages with QoS 1 or 2.

Revision History

A manual revision code appears as a suffix to the catalog number on the front and back covers of the manual.



Revision code	Date	Revised content
01	June 2024	Original production

1

Overview

This section describes MQTT API functions, features, and concepts.

1-1 Introduction

Omron's MQTT API is an interface application that enables integration between the Fleet Manager, AMR, and the end user's client application. The MQTT API is part of the Fleet Operations Workspace (FLOW) Core software suite.
Both the MQTT API and the Integration Toolkit (also part of the FLOW Core software) are integration applications. MQTT API uses the industry standard MQTT network protocol for system integration while the Integration Toolkit application offers RESTful, SQL, and RabbitMQ communication channels.

1-2 Functions and Features

The MQTT API provides a secure integration layer between an AMR, a fleet manager device, and the user's client application using a broker based subscribe-publish architecture. The MQTT API facilitates full management and monitoring of all AMR operations such as pickup, dropoff, digital I/O, dataStore values, and multi-segment. It also allows tracking of AMR data directly.

Additional Information

- The MQTT API can operate in parallel with existing ARCL communication. The MQTT API does not replace ARCL for direct AMR control (once it has reached a goal). Refer to *Advanced Robotics Command Language AMR Reference Guide (Cat. No. I617)* for more information.
- The MQTT API utilizes TLS encryption to establish secure connections between MQTT clients and the broker. However, no certificate validation is implemented.

1-2-1 MQTT API Advantages

The MQTT API offers the following advantages:

- Industry standard MQTT protocol allows for interoperability with diverse devices.
- A single point of connection to interact with the entire fleet, simplifying integration architecture.
- Simple, efficient, and flexible message queuing.
- Enhanced security with TLS encryption to ensure secure communications between Fleet Manager, AMRs, and other devices.
- MQTT API minimizes network transmission overhead, making it ideal for limited bandwidth environments.
- Low-latency interaction with the AMR or Fleet Manager.
- Ad hoc access to DataStore values.
- Simple format to create batch/bulk or single requests.
- Asynchronous messaging with read/write capabilities.
- Simple monitoring for the AMR or Fleet Manager.

1-2-2 MQTT API Limitations

Though there are several advantages to using the MQTT API over the three integration communication channels such as REST, SQL, or RabbitMQ, there are still a few limitations.

- The MQTT API does not provide any database interaction (no data persistence).
- No search history feature.
- Limited queuing abilities.

1-2-3 MQTT API Considerations

Make the following considerations when using the MQTT API:

- The MQTT API and the broker are hosted in both the Fleet manager and AMR using port 8883.
- MQTT API uses self-signed certificates with TLS 1.2 (or above) encryption protocol.
- Valid user name and password are required for connecting to the broker.
- For command interaction, use correct topic syntax and JSON format for payload schema.
- Ensure each client ID is unique if connecting multiple devices to the same broker.

- Ensure Quality of Service (QoS) is adjusted properly for efficient messaging.

1-2-4 MQTT API Basics

The MQTT protocol is fundamental to MQTT API's functionality. It is a lightweight, publish-subscribe network protocol enabling efficient data transmission in limited bandwidth networks. Two key components in establishing an MQTT connection for publishing and subscribing of messages are an MQTT client and MQTT broker. The following MQTT concepts help in understanding MQTT API functionality.

Broker

An MQTT broker is a server-based network that is the central hub in the publish/subscribe messaging system, managing the communication between different clients. The broker receives, filters, and distributes messages to the appropriate clients efficiently and reliably. The ExternalComms application (part of the FLOW Core software) contains the MQTT broker. Refer to *Section 2 Setting Up and Connecting* on page 2-1 for more information.

Client

The MQTT client is a device that connects to an MQTT broker over a network to send and receive (publish and subscribe to) messages. The MQTT client functionality resides in the MQTT API.

Topic

A topic refers to a UTF-8 string that is structured in a hierarchical manner used for filtering and routing messages to a connected client. A topic syntax consists of one or more levels separated by a forward slash (topic level separator).

```
itk/cmd/dropoff/req
```

Refer to *3-1 MQTT API Topics* on page 3-2 for more information.

Wildcards

MQTT supports two types of wildcards for topic subscription.

Name	Symbol	Description	Example
Single-Level	+	Replaces a single topic level and matches any string in that level	`itk/+` or `itk/+/req`
Multi-Level	#	Matches multiple levels of a topic and must be placed at the end of the topic string	`itk/#`

Quality of Service (QoS)

Quality of Service (QoS) is a key feature in the MQTT protocol that determines the level of assurance for message delivery between the client and the broker. This value can be set for every message published or subscribed to for a topic.

Name	Value	Description
At Most Once	0	• Also known as “fire and forget” • There is no guarantee of delivery since receiver does not return message acknowledgement to sender • Used for non-critical or high-rate messages • Low impact on network
At Least Once	1	• Message delivery is guaranteed since receiver sends acknowledgment back to sender • Can result in duplicated messages if acknowledgments are lost • Used for cases when message duplication is not an issue • Generally low impact on network
Exactly Once	2	• It guarantees message is delivered exactly once • Used only for critical message delivery • High impact on network

Retained Topic

A retained topic is an MQTT feature that allows storing the last message for a particular topic on the broker and delivering the message to a client whenever a client subscribes to the matching topic. The user sets a topic as retained in the MQTT client.
Example:

1. MQTT API client publishes a message with the current list of AMRs. The topic
   `itk/dt/robot/list`
   is marked as a retained topic with payload *AMR1* (AMR name).
2. A new subscriber connects, and subscribes to the topic
   `itk/dt/robot/list`
3. The MQTT broker immediately publishes the last retained message for that topic with payload *AMR1*, to any new subscriber ensuring they have the latest message (payload) for
   `itk/dt/robot/list`

Additional Information

Retained messages are stored in device memory as the MQTT broker does not provide data persistence. QoS, retained messages, and sessions will be cleared if you restart or reset the ExternalComms application.

Message Governor

The message governor feature in MQTT API limits the rate of incoming messages processed by the MQTT client. This feature conserves CPU and memory usage and alerts users about unusual message traffic.
Refer to *2-7 Messaging Limits* on page 2-18 for more information.

2

Setting Up and Connecting

This section describes setting up the MQTT API and establishing a connection.

2-1 Installation

The MQTT API and the ExternalComms applications are part of the the FLOW Core software suite version 4.1 (or above). These application packages are automatically installed and enabled when the FLOW Core software is installed.
If your AMR or Fleet Manager device has an older version of FLOW Core installed, you will need to upgrade to FLOW Core 4.1 (or above) in order to use the MQTT API. Refer to *2-2 Software Management* on page 2-3 for more information.
The ExternalComms package contains the broker functionality and the MQTT API contains the client.

Additional Information

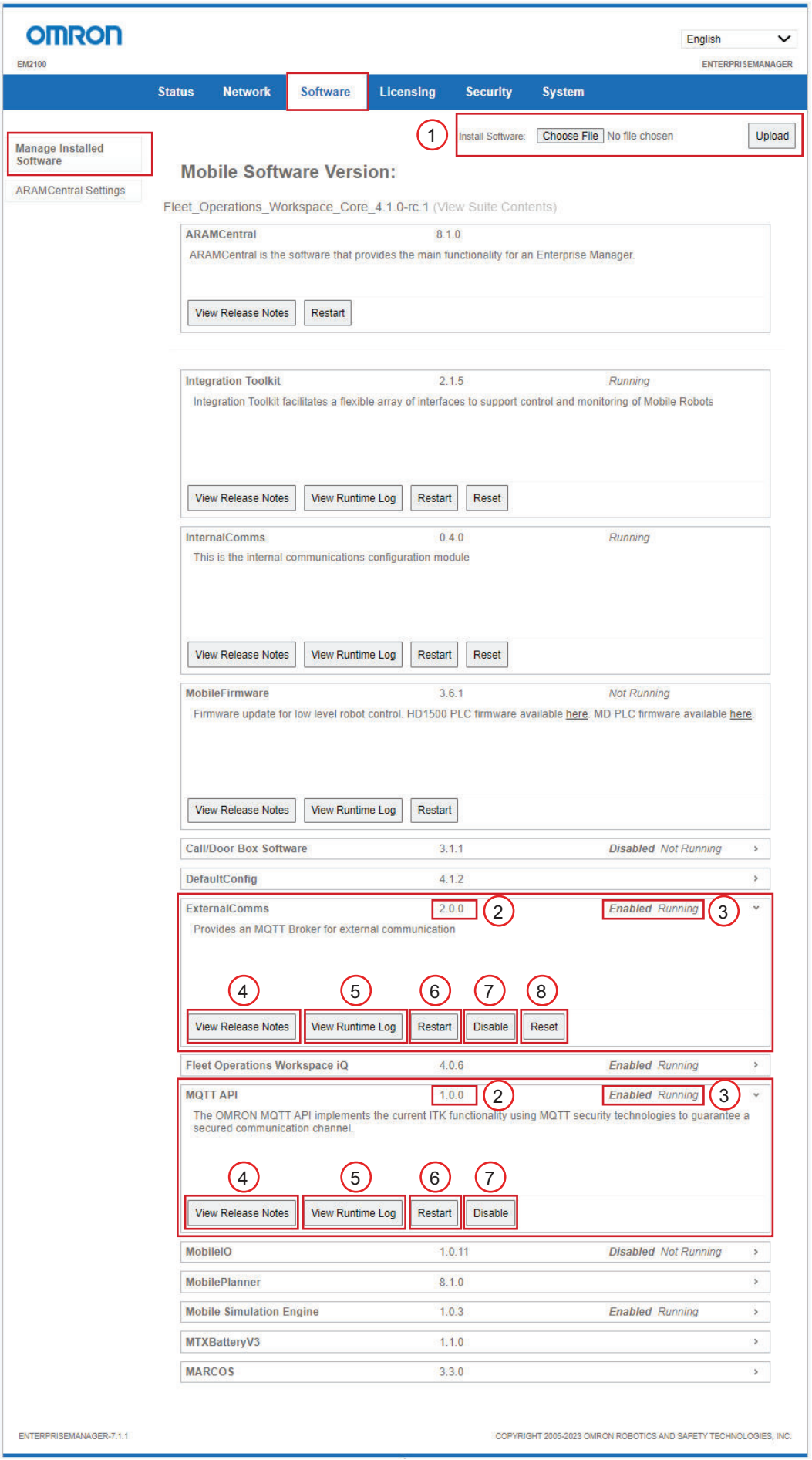
If you have a simulated AMR setup, the MQTT API and ExternalComms do not get enabled automatically. You will need to enable both applications in the SetNetGo interface Software Tab window.

2-2 Software Management

The SetNetGo interface provides an area to manage the MQTT API and ExternalComms applications. Access the SetNetGo interface through MobilePlanner or a web browser. Refer to the *Fleet Operations Workspace Core User's Manual (Cat. No. I635)* for more information about accessing SetNetGo.

In the **Software** tab of SetNetGo, click the **Manage Installed Software** option in the left.

The following software management functions are available for the MQTT API and the ExternalComms applications:


OMRON
English
EM2100
ENTERPRISEMANAGER
Status
Network
Software
Licensing
Security
System
Manage Installed Software
ARAMCentral Settings
1
Install Software:
Choose File
No file chosen
Upload
Mobile Software Version:
Fleet_Operations_Workspace_Core_4.1.0-rc.1 (View Suite Contents)
ARAMCentral 8.1.0
ARAMCentral is the software that provides the main functionality for an Enterprise Manager.
View Release Notes
Restart
Integration Toolkit 2.1.5 Running
Integration Toolkit facilitates a flexible array of interfaces to support control and monitoring of Mobile Robots
View Release Notes
View Runtime Log
Restart
Reset
InternalComms 0.4.0 Running
This is the internal communications configuration module
View Release Notes
View Runtime Log
Restart
Reset
MobileFirmware 3.6.1 Not Running
Firmware update for low level robot control. HD1500 PLC firmware available here. MD PLC firmware available here.
View Release Notes
View Runtime Log
Restart
Call/Door Box Software 3.1.1 Disabled Not Running
DefaultConfig 4.1.2
ExternalComms 2.0.0 2 Enabled Running 3
Provides an MQTT Broker for external communication
4
5
6
7
8
View Release Notes
View Runtime Log
Restart
Disable
Reset
Fleet Operations Workspace iQ 4.0.6 Enabled Running
MQTT API 1.0.0 2 Enabled Running 3
The OMRON MQTT API implements the current ITK functionality using MQTT security technologies to guarantee a secured communication channel.
4
5
6
7
View Release Notes
View Runtime Log
Restart
Disable
MobileIO 1.0.11 Disabled Not Running
MobilePlanner 8.1.0
Mobile Simulation Engine 1.0.3 Enabled Running
MTXBatteryV3 1.1.0
MARCOS 3.3.0
ENTERPRISEMANAGER-7.1.1
COPYRIGHT 2005-2023 OMRON ROBOTICS AND SAFETY TECHNOLOGIES, INC.

Item	Description
1	Select and upload a new FLOW Core software file.*1
2	The current version of the installed application.
3	The operational status of the application (Running or Not running).
4	Opens a dialog box to display the release notes.
5	Opens a dialog box to display the application's RunTime Log for diagnostic purposes.
6	Restarts the application.*2
7	Disables or stops the application from running in the system.
8	Resets ExternalComms to refresh user credentials.*3

*1. Refer to Additional Information below.

*2. Refer to Additional Information below.

*3. Click the Reset button to refresh user credentials if broker to broker communication fails or if user credential authentication with broker fails.

Additional Information

MQTT API functionality relies on other applications for operation. Any action that stops the MQTT API or other applications may have an impact on data integrity and functionality. Pausing fleet activity during a planned software stoppage is recommended.

2-3 Security

MQTT API security is implemented using a self-signed certificate to establish a TLS encrypted (TLS 1.2) connection between the clients and the broker. A username and password are used for authentication. It is not possible to configure communications without this encrypted connection.

Additional Information

If you are concerned about the secure transport of the self-signed certificate (since the client is not authenticating the certificate), then this certificate can be moved, loaded, and trusted manually.

2-4 User Management and Access Control

To interact with the MQTT API, you need to create users and specify a password in every system including each AMR and the Fleet Manager. Only two usernames are allowed for accessing the MQTT API - apiControl and apiMonitor. Use apiControl for read-write operations with all privileges. Use apiMonitor for monitoring or read-only operations.
The table below lists the usernames and their topic permissions.

Username	Write Permission	Read permission
apiControl	All Command topics	All MQTT API topics*1
apiMonitor	DataStore Value Request topics	

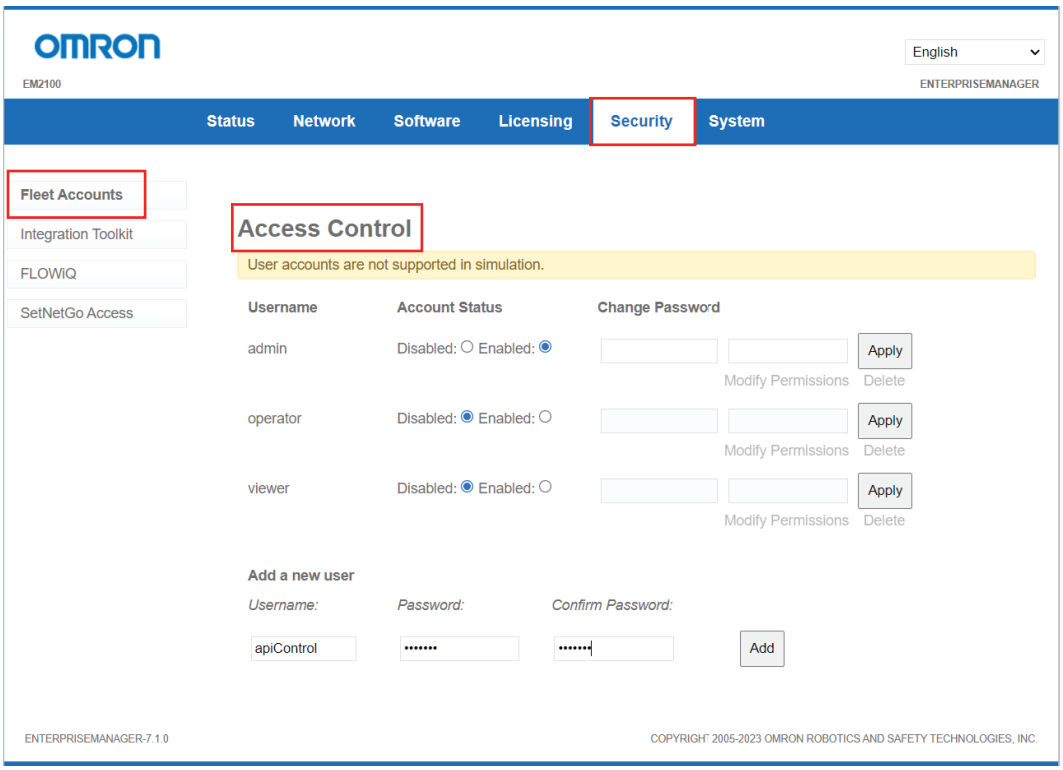
*1. Refer to *3-1 MQTT API Topics* on page 3-2 for more information.

Passwords are user specified. Passwords must be 1 - 20 alphanumeric characters only and are case sensitive.

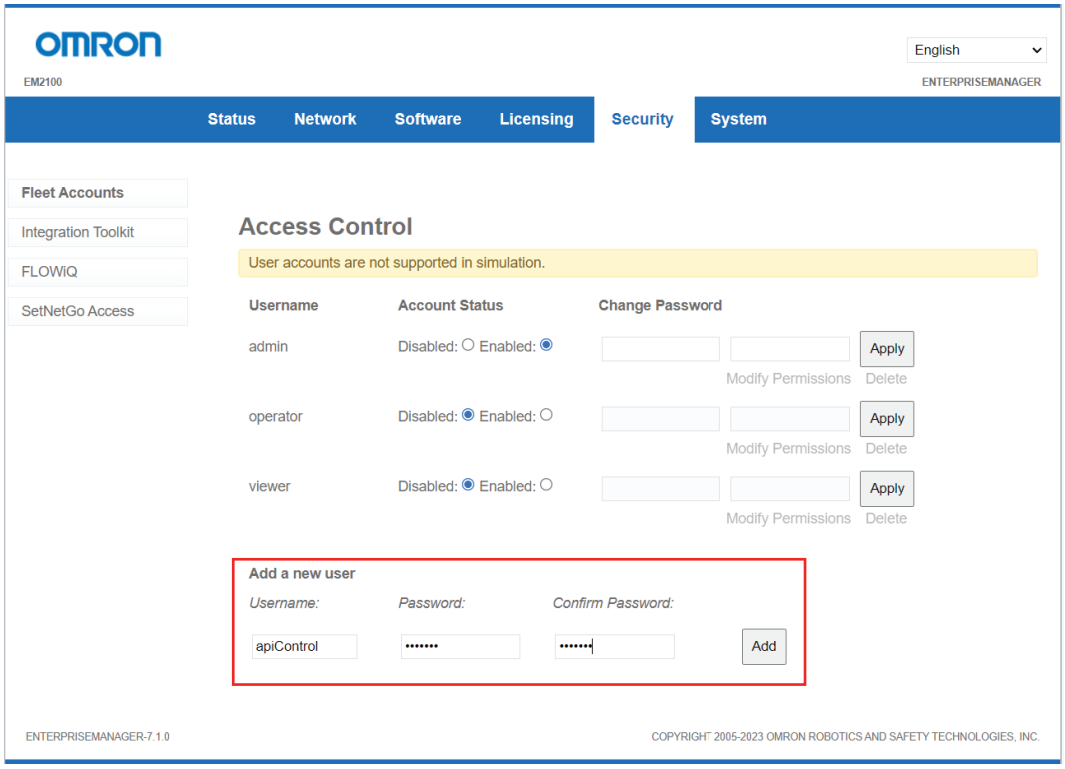
2-4-1 Set Username and Password

To create MQTT API user accounts on an AMR or a Fleet Manager, access the SetNetGo interface. Refer to the *Fleet Operations Workspace Core User's Manual (Cat. No. I635)* for more information. Follow the instructions below to set up users:

1 Click the **Security** tab in SetNetGo.
The **Access Control** window for the **Fleet Accounts** option opens.

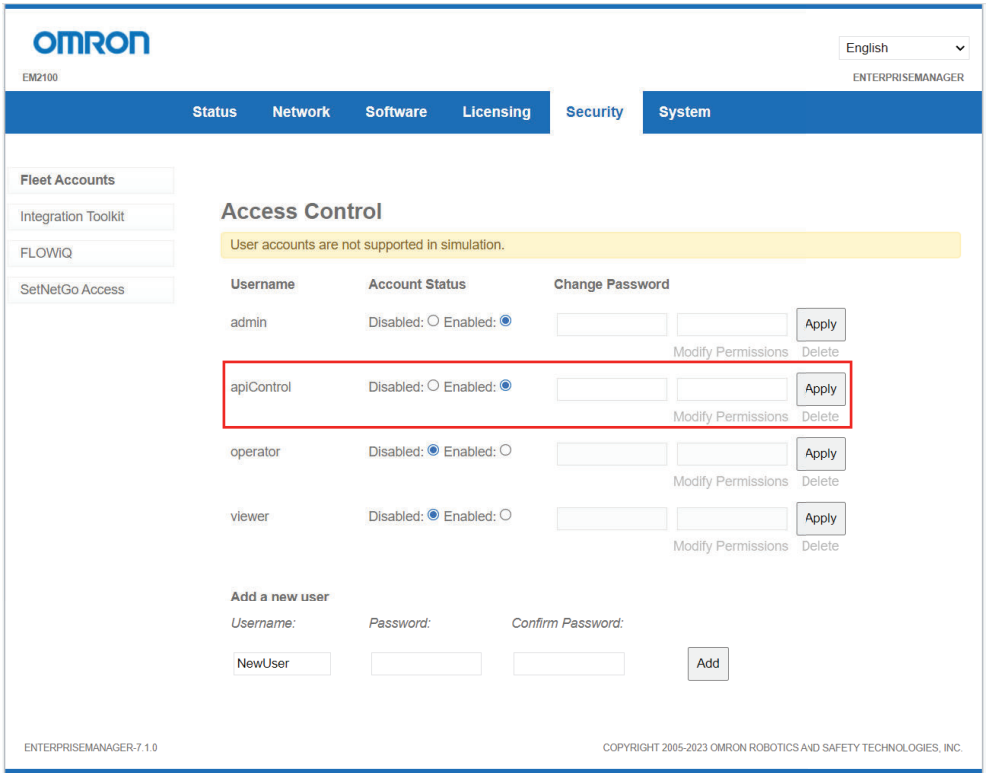


2 In the **Add a new user** section, enter a valid username.
Only two usernames *apiMonitor* and *apiControl* are allowed. The username is case sensitive.



3 Enter a password and confirm the password.
Passwords are case sensitive and must be 1 - 20 alphanumeric characters only.

4 Click *Add*.
The new user is created and automatically enabled. This completes the procedure.



You can also control user access to the MQTT broker by clicking the **Enabled** or **Disabled** button above.

2-5 Establish Connection

To utilize the MQTT API, you need to establish a connection with the broker. There are different ways to interact with the broker, you can either implement your own MQTT client programmatically, or use a GUI client tool such as MQTT Explorer or MQTTX.

2-5-1 Connect Programmatically

Use the example code below to implement your MQTT API client and establish connection with the broker.

Collect the following information before implementing the client:

- Username and password - refer to *2-4 User Management and Access Control* on page 2-7
- Host - IP address of your AMR or Fleet Manger
- Port - 8883

The following is an example Python code for implementing the MQTT client programmatically:

```
# Third party library that implements MQTT client
import paho.mqtt.client as mqtt
# Standard library to establish secure communication
import ssl

USERNAME = "apiControl"
PASSWORD = "control"
HOST = "xx.xxx.xx.xxx"
PORT = 8883

# Create MQTT client instance
client = mqtt.Client()
client.username_pw_set(username=USERNAME, password=PASSWORD)
context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2)
client.tls_set_context(context)
client.connect(host=HOST, port=PORT, keepalive=60)

# Start networking loop
client.loop_start()
```

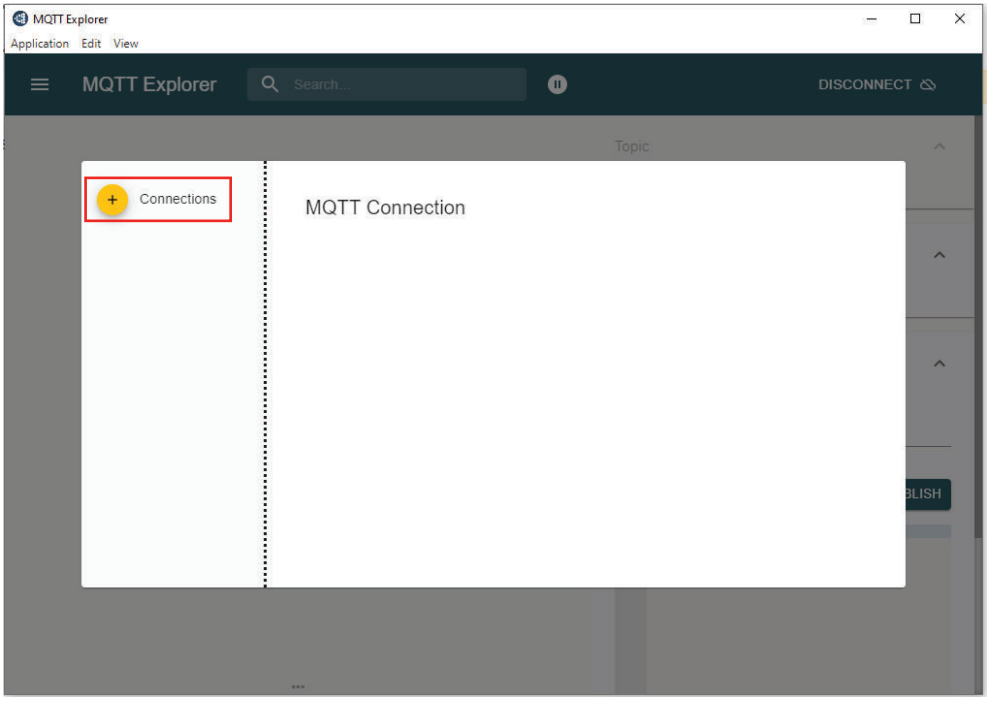
2-5-2 Connect Using MQTT Client GUI

MQTT Explorer is used as a client GUI example for establishing connection with the broker. Before beginning the procedure, download and install the MQTT Explorer client tool.

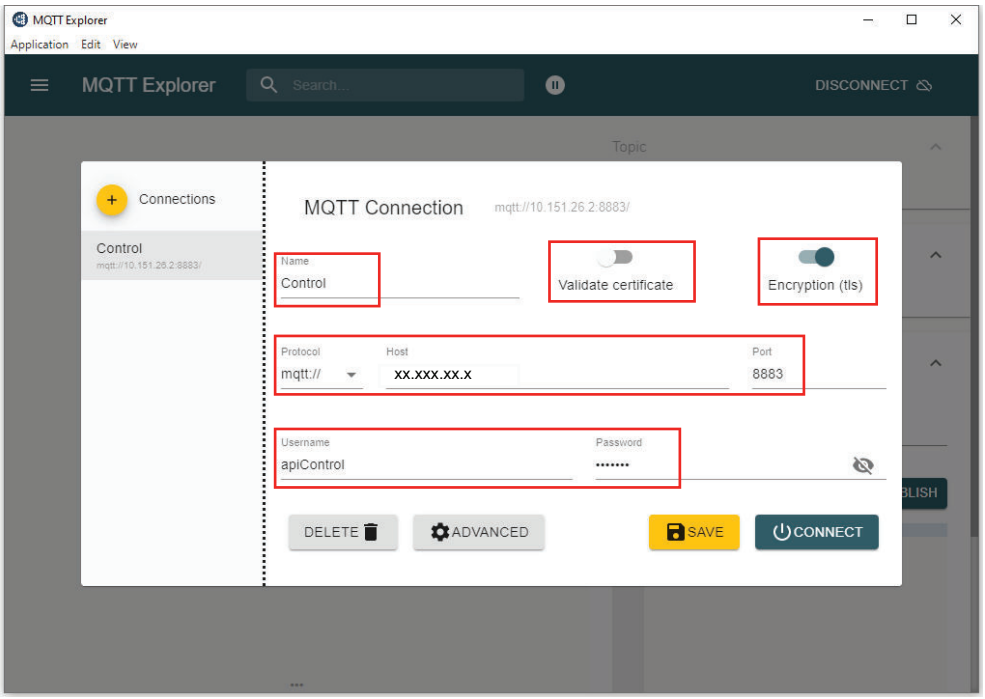
Additional Information

MQTT Explorer version 0.4.0-beta1 was used for the example procedure below. Other versions may appear differently.

1 Open the application and click **Connections**.



The *MQTT Connection* window opens.



2 Enter a profile name in the *Name* field.

3 Disable the **Valid certificate** button since certificate validation is not supported.

4 Enable **Encryption (tls)**.

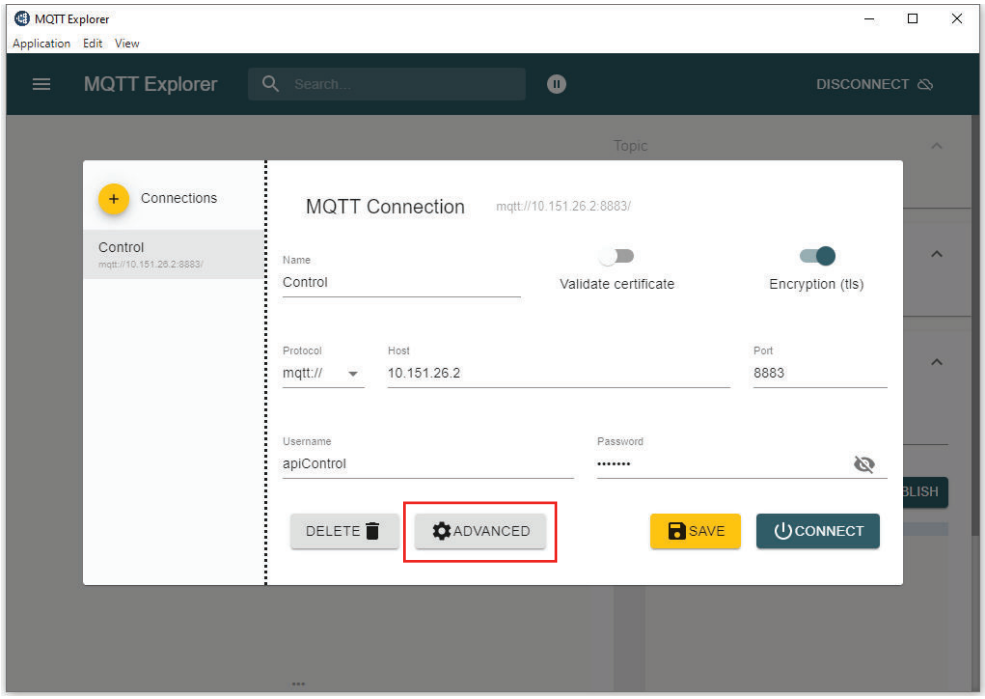
5 In the *Protocol* field, choose **MQTT** from the drop down.

6 Enter the AMR's or Fleet Manager's IP address in the *Host* field.

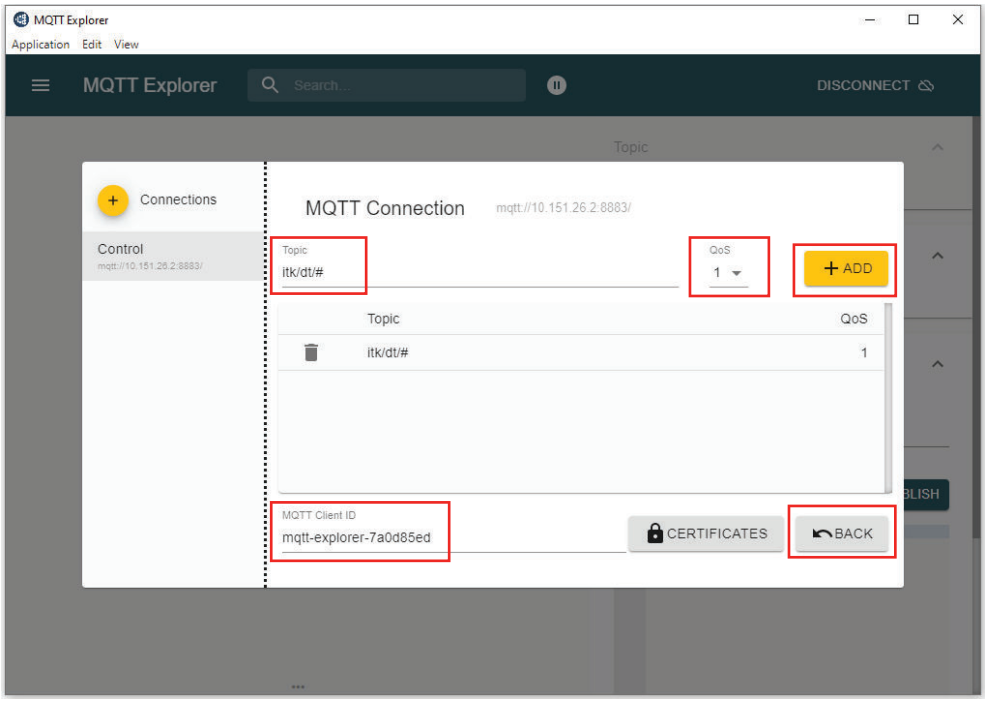
7 Enter **8883** for the *Port*.

8 In the *Username* and *Password* fields, enter the same username and password that you defined in the SetNetGo Fleet Accounts page when you created user account name and password. Refer to *2-4 User Management and Access Control* on page 2-7.

9 After you set the basic network configuration in the steps above, click **Advanced** to configure initial subscriptions and QoS.



The following window opens.



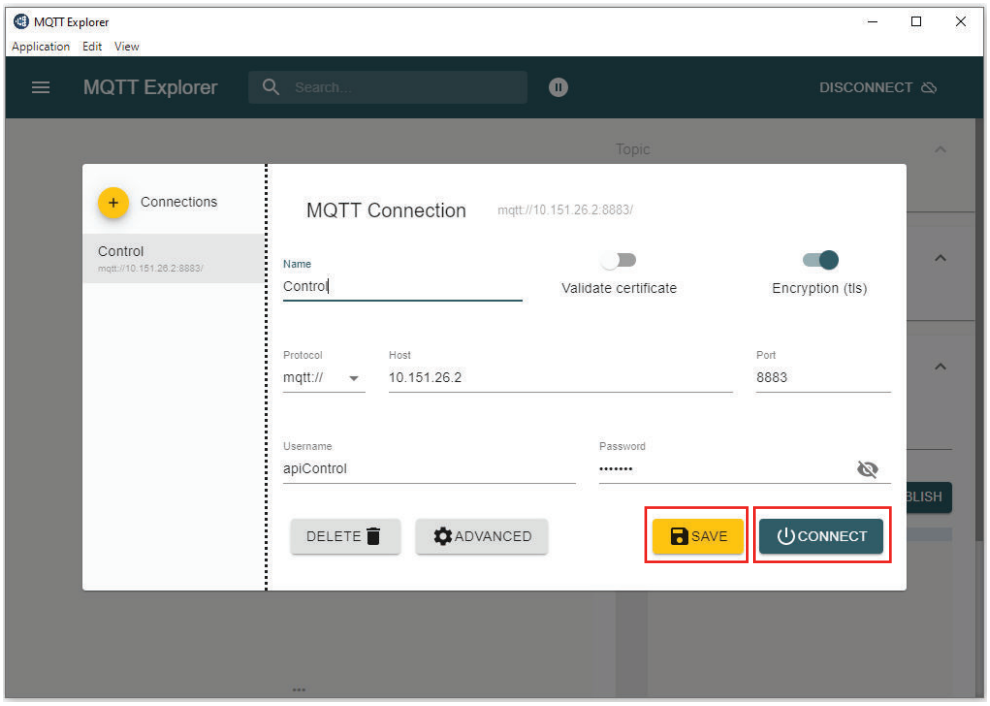
10 In the *Topic* field, enter a topic. Refer to *3-1 MQTT API Topics* on page 3-2 for more information.

11 Ensure your *MQTT Client ID* is unique. Reusing this name will cause frequent disconnections.

12 Ensure you set *QoS* for your specific need and according to your network bandwidth. Refer to *1-2-4 MQTT API Basics* on page 1-4 for more information about QoS.

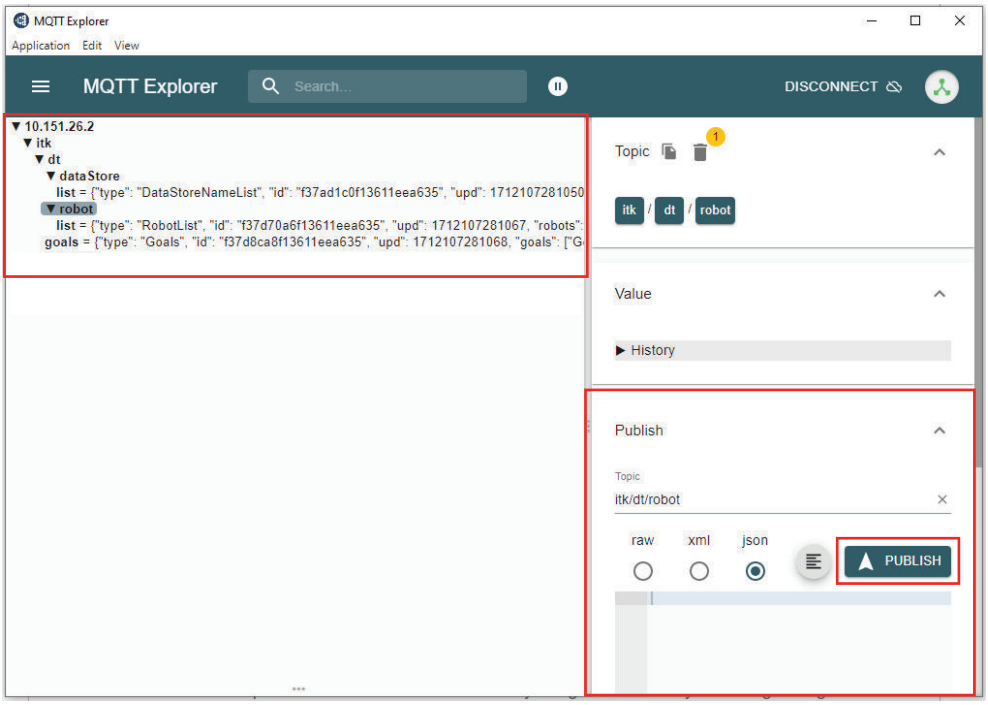
13 Click **Add** to add the topic entered in step 10.

14 Click the **Back** button to go back to the basic MQTT Connection window.
The following window opens.



15 Click **Save** to store network and subscription settings.

16 Click **Connect**.
The following window opens to indicate that connection has been established with the broker.



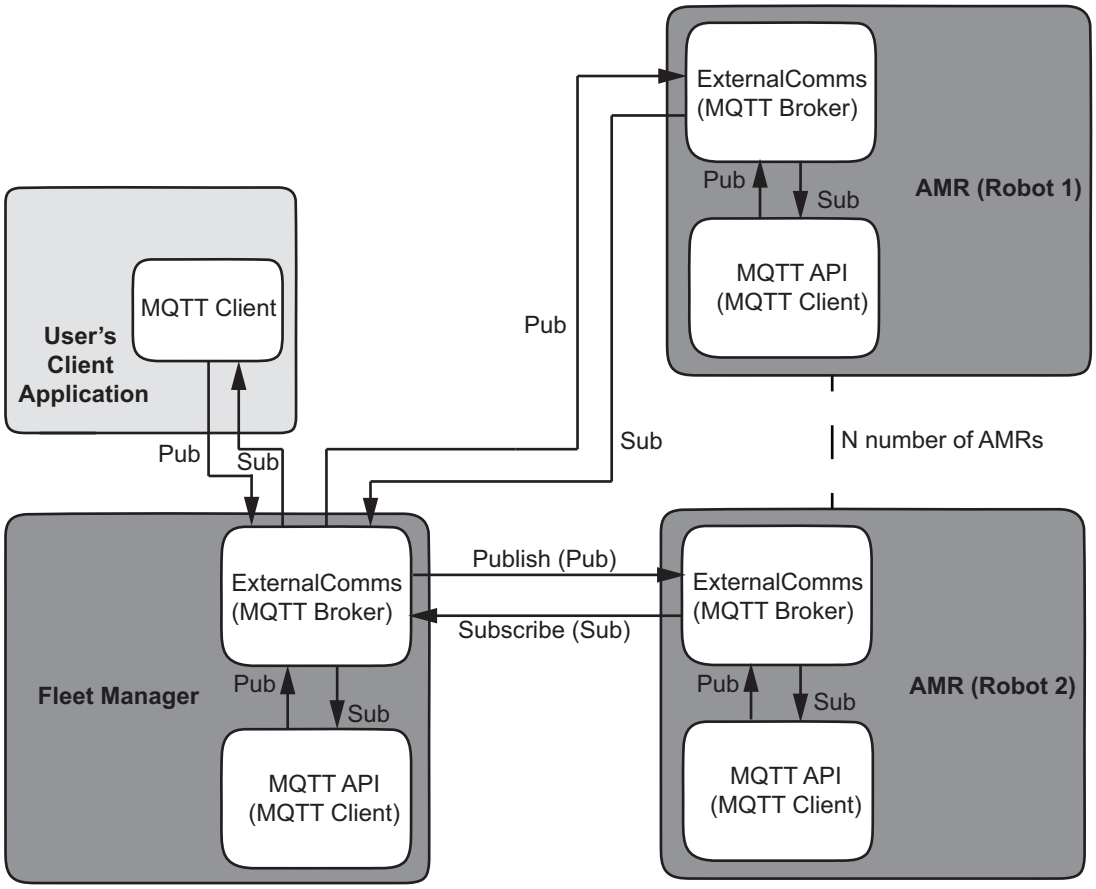
Topic structure with a set of drop downs are displayed on the left side of the window.

17 Use the *Publish* panel on the right to publish messages.
This completes the connection procedure.
Retained messages are immediately visible and other messages are received and displayed with system updates. Refer to *1-2-4 MQTT API Basics* on page 1-4 for more information about Retained messages/topics.

2-6 Broker to Broker Communication

In typical deployments, each AMR and Fleet Manager has its own MQTT broker (ExternalComms) and client (MQTT API). This set up facilitates the capability for broker-to-broker (B2B) communication enhancing connectivity and control across your fleet. With B2B communication, you can manage all the AMRs in the fleet from a single point through the Fleet Manager broker. This architecture simplifies complex network setups, synchronization of operations, execution of coordinated tasks, and provides a more efficient and flexible way to manage data exchange and command distribution across your fleet.

The B2B functionality gets automatically enabled when an AMR is connected to a Fleet Manager. The following illustration represents the access to the entire fleet from a single point of connection.



2-6-1 MQTT Bridge

An MQTT bridge is set up to share messages between diverse systems by connecting two MQTT brokers together. A local broker is configured to act as a bridge between a remote broker and the clients connected to both brokers.

When you configure a broker to act as a bridge, it becomes a client to the remote broker and subscribes/publishes to topics on the remote broker just like any other MQTT client. The bridge (broker) is referred to as a bridge client.

Additional Information

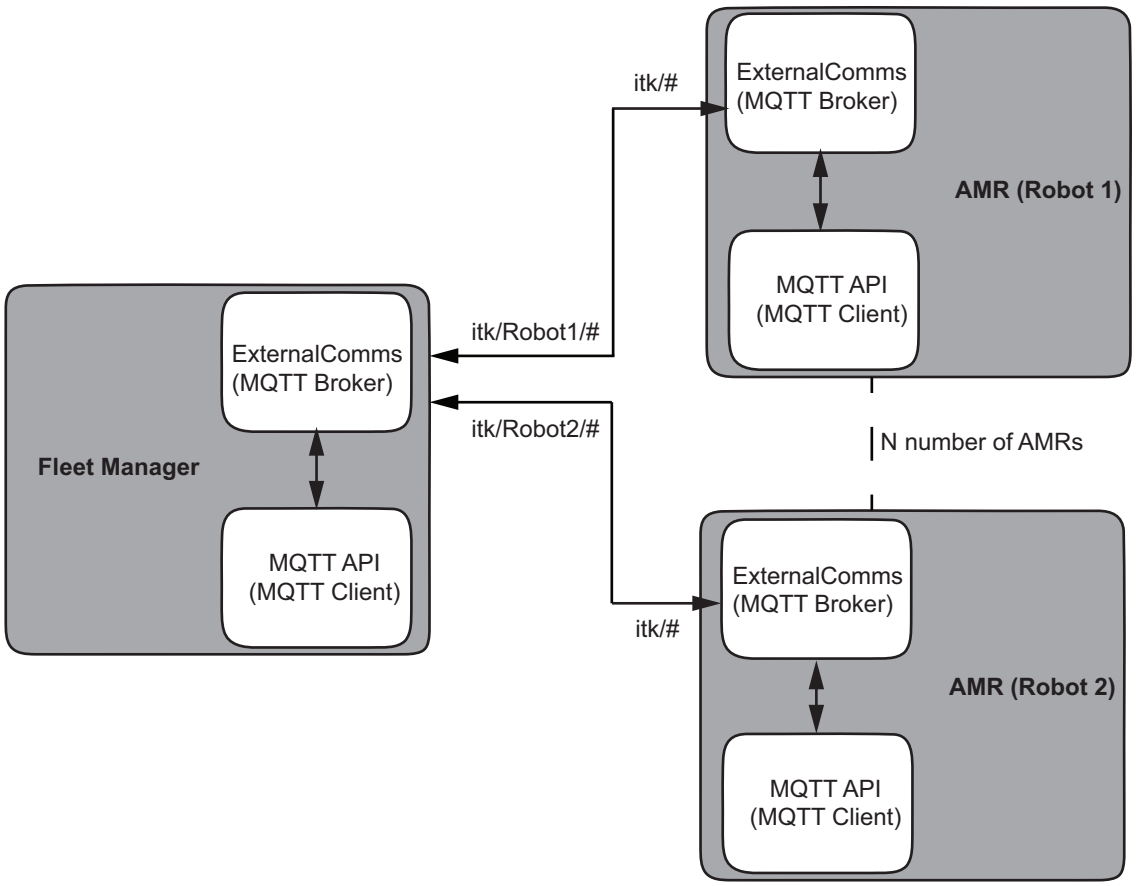
To enable a bridge connection between an AMR and a Fleet Manager, you must use the same password for MQTT API user accounts on both systems.

2-6-2 Topic Remapping

The topic remapping feature allows the alteration of MQTT topic names as messages are exchanged between brokers, facilitating organized and efficient data routing within a network.
The following remapping rule is implemented:

Original Topic (At Source)	Source Device	Destination	Remapped Topic (Destination)
`itk/<amr-name>/#`	from Fleet Manager	to AMR	`itk/#`
`itk/#`	from AMR	to Fleet Manager	`itk/amr_name/#`

In the above table, both rows represent the same rule but from different connection perspectives. The Original Topic represents the topic published/subscribed at source, and the Remapped Topic, the topic published/subscribed at destination.
Refer to the illustration below for a schematic representation of topic remapping architecture in B2B communication.



For example, assume you have a Fleet Manager with a Robot named "Robot 1" and you need to switch its digital output device on or off. The topic for publishing if connected from Robot 1 is as follows:

`itk/cmd/digOutputSwitch/req`

If connected to the Fleet Manager broker, you must include the AMR name in the topic after the "itk" topic level:

`"itk/Robot1/cmd/digOutputSwitch/req"`

The same rule applies to subscription topics as well:

`itk/Robot1/cmd/digOutputSwitch/res/+`

Additional Information

- Refer to all the command topics in the section *3-1-1 Command Topics* on page 3-2 for more information about the bridge topics to use if connected from the Fleet Manager.
- Refer to *Turn Digital Output On or Off* on page 3-29 for more information about digital output switch on/off procedure.

2-7 Messaging Limits

Messaging limits are imposed on the MQTT broker and the client.
The MQTT broker in the ExternalComms application is pre-configured for messaging limits. If the following limits are exceeded, the messages are dropped.

- Inflight messages: 20 (maximum)
- Queued messages: 1000 (maximum)
- Packet size: 2 MB (maximum)

The message governor limits the rate of incoming messages processed by the MQTT client. The message governor sets different message limits on different devices based on their hardware capabilities:

- Fleet Managers: 200 (maximum) incoming messages per second
- AMRs: 100 (maximum) incoming messages per second
- Inflight messages: 20 (maximum)

If the message limit is exceeded, an error message appears as topic response.
Error message payload example:

```
{
  "type": "GovernorError",
  "id": "207baeda117e11efa481",
  "upd": 1715667088030,
  "status": "Error",
  "description": "Payload was not processed due to message rate limit"
}
```

3

Communication

This section describes using MQTT API topics for communication.

3-1 MQTT API Topics

MQTT topic strings are essential for establishing communication between MQTT clients and brokers. Publishers (clients) send messages to specific topics, while subscribers (clients) can subscribe to those topics to receive the messages. The broker uses topics to filter messages for all connected clients according to their subscriptions and forwards to those subscribers.
The MQTT API allows the following topics for communication and integration between your AMRs, Fleet Manger, and other third party applications:

- Command Topics
- Data Topics
- Custom Topic

The following sections provide details about each category of MQTT API topics using a tabular format. The table properties are explained below:

Table Properties	Description
Topic	Topic name (hierarchical string) to which a client is subscribed or publishes data.
QoS	Quality of service to which a client is subscribed or publishes data.
Retain	These topics are marked to be retained in the broker when the client publishes.
System	Device (Fleet manager/AMR) that can only implement such topic functionalities natively.

3-1-1 Command Topics

MQTT API command topics implement functionality that trigger actions such as request a job, cancel a job or request datastore values. The command topics are characterized by the prefix `itk/cmd` and either the suffix `req` for request or `res` for response. All command topics use JSON format and follow a schema.
The following is the request format:
Publish to topic:
`itk/cmd/<req-context>/req`
Use the payload for providing details of the request.

Additional Information

Refer to each command topic section below for request payload format examples.

The following is the response format:
Subscribe to topics:
To verify if the request was successful:
`itk/cmd/<req-context>/res/<req-context-id>`
To verify there were no errors:
`itk/cmd/<req-context>/res/error`
To use a single level wildcard to subscribe to both:
`itk/cmd/<req-context>/res/+`
The payload is generated by the MQTT API since a response topic is for subscription only.
Response payload format example:

```
{
  "type": "JobRequest",
  "id": "JOBe3d7dd9",
  "upd": 1714509569778,
  "status": "Success",
  "description": "Successfully requested job with id JOBe3d7dd9",
  "job_id": "JOBe3d7dd9",
  "default_priority": false,
  "details": [
    {
      "goal": "Goal101",
      "segment_type": "Pickup",
      "priority": 10
    }
  ]
}
```

- type (string): describes the message type.
  Example:
  - "JobRequest"
  - "SchemaError"
  - "ArclRequest"
- id (string): unique identifier for the message. If you did not specify an ID in the request payload, the MQTT API client will automatically generate an ID in the response payload.
  Example:
  - "5e208bea0ccb11ef9858"
  - "CPUUse" (more concrete IDs are used for dataStore values).
- upd (integer): timestamp update.
  Example:
  - 1715150506886 (represents Wednesday, May 8, 2024 6:41:46.886 AM.
- status: indicates the status of the requests.
  Example:
  - Success: The request was valid and processed.
  - Error: The request failed and was not processed.
  - Forward: The request was forwarded to another server and whether valid or processed is unknown.
- description: short description of the message.
  Example:
  - Payload is not in JSON format
- arguments: depending on the message type, these are the arguments used during the request.

Additional Information

When using command topics for sending a request, it is important to subscribe to the corresponding response and data topics before publishing the request. This ensures messages are not missed.

Job Request

Use this topic to request a single or multi-segment job.

Topic	QoS	System
itk/cmd/job/request/req	2	Fleet Manager

Payload schema: JSON format

- default_priority (required): boolean
- details (required): array of objects (1 min. to 100 max. items)
  - Object schema
    - goal: string
    - priority: integer
    - segment_type: string
      - Values allowed: “Pickup”, “Dropoff”, “pickup”, and “dropoff”
- job_id: string

  You can specify an ID in the request payload. If you do not, then the MQTT API client will automatically generate an ID in the response payload.

Example:

```
{
"default_priority": false,
"details": [
    {
     "goal": "Goal117",
     "priority": 17,
     "segment_type": "Pickup"
    },
    {
     "goal": "Goal101",
     "priority": 27,
     "segment_type": "Dropoff"
    }
    ],
    "job_id": "JOB1"
}
```

For topic response, subscribe to the following:

- For success confirmation:

  itk/cmd/job/request/res/<job-id>

  Payload example:

```
{
  "type": "JobRequest",
  "id": "JOBe3d7dd9",
  "upd": 1714509569778,
  "status": "Success",
  "description": "Successfully requested job with id JOBe3d7dd9",
  "job_id": "JOBe3d7dd9",
  "default_priority": false,
  "details": [
```

```
    {
      "goal": "Goal101",
      "segment_type": "Pickup",
      "priority": 10
    }
  ]
}
```

- To check if request failed:

  `itk/cmd/job/request/res/error`

  Payload example:

```
{
  "type": "JobRequest",
  "id": "JOB43200fd",
  "upd": 1714509587035,
  "status": "Error",
  "description": "no such goal: NotInMapGoal.",
  "job_id": "JOB43200fd",
  "default_priority": false,
  "details": [
    {
      "goal": "NotInMapGoal",
      "segment_type": "Pickup",
      "priority": 10
    }
  ]
}
```

Additional Information

- Refer to *Job Status* on page 3-12 for more information about receiving job updates.
- Refer to *Request a Job* on page 3-21 for a job request usage example.

Job Modify

Use this topic to modify a segment in a job.

Topic	QoS	System
`itk/cmd/job/modify/req`	2	Fleet Manager

Payload schema: JSON format

- segment_id (required): string
- modify_type (required): integer
  - values allowed
    - 5: Modify based on goal name
    - 6: Modify based on priority
- modify_value (required): string if “modify_type” value is 5, or integer if it is 6

Payload example:

```
{
    "segment_id": "PICKUP1",
    "modify_type": 5,
    "modify_value": "Goal102"
}
```

For topic response, subscribe to the following:

- For success confirmation:

  `itk/cmd/job/modify/res/<modify-value>`

  Payload example:

```
{
  "type": "JobModify",
  "id": "9400459e086b11efbb60",
  "upd": 1714658761347,
  "status": "Success",
  "description": "Successfully modified segment PICKUP2018 with value 85",
  "segment_id": "PICKUP2018",
  "modify_type": 6,
  "modify_value": 85
}
```

- To check if request failed:

  `itk/cmd/job/modify/res/error`

  Payload example

```
{
  "type": "JobModify",
  "id": "005d088a086c11efbb60",
  "upd": 1714658943173,
  "status": "Error",
  "description": "id not found: NotExistsSegment",
  "segment_id": "NotExistsSegment",
  "modify_type": 6,
  "modify_value": 85
}
```

Additional Information

Refer to *Modify Job* on page 3-24 for a job modification usage example.

Job Cancel

Use this topic to cancel a job.

Topic	QoS	System
`itk/cmd/job/cancel/req`	2	Fleet Manager

Payload schema: JSON format

- cancel_type (required): integer
  - values allowed
    - 1: Cancel by segment id

  - 2: Cancel by job id
  - 3: Cancel by robot name
  - 4: Cancel by state
- cancel_value (required): string
- echo_msg (optional): string
- cancel_reason (optional): string

Additional Information

Ensure that the values entered for "cancel reason" and "echo message" do not contain white (blank) spaces.

Example:

```
{
    "cancel_type": 1,
    "cancel_value": "PICKUP1",
    "echo_msg": "cancelled",
    "cancel_reason": "timeout"
}
```

For topic response, subscribe to the following:

- For success confirmation:

  `itk/cmd/job/cancel/res/<cancel-value>`

  Payload example:

```
{
  "type": "JobCancel",
  "id": "96657c28086b11efbb60",
  "upd": 1714658765393,
  "status": "Success",
  "description": "Successfully cancelled job with value JOBa38f02d",
  "cancel_type": 2,
  "cancel_value": "JOBa38f02d",
  "echo_msg": "",
  "cancel_reason": "None"
}
```

- To check if request failed:

  `itk/cmd/job/cancel/res/error`

  Payload example:

```
{
  "type": "JobCancel",
  "id": "38ef000a121411efbe47",
  "upd": 1715731554209,
  "status": "Error",
  "description": "jobId error: JOBa38f02d",
  "cancel_type": 2,
  "cancel_value": "JOBa38f02d",
  "echo_msg": "",
  "cancel_reason": ""
}
```

Additional Information

Refer to *Cancel Job* on page 3-26 for a job cancel usage example.

Drop Off

Use this topic to create a drop off job for an AMR.

Topic	QoS	System
itk/cmd/job/dropoff/req	2	AMR
itk/<amr-name>/cmd/job/dropoff/req	2	Fleet Manager (bridge)

Payload schema: JSON format

- job_id: string
- goal (required): string
- priority: integer

Payload example:

```
{
  "goal": "Goal123",
  "job_id": "AMR-Dropoff-Goal123",
  "priority": 10
}
```

For topic response, subscribe to the following:

- For success confirmation:

  itk/cmd/job/dropoff/res/<job-id>

  Payload example:

```
{
  "type": "Dropoff",
  "id": "7358a002071211efbfa0",
  "upd": 1714510530006,
  "status": "Success",
  "description": "Successfully requested dropoff to Goal123",
  "job_id": "AMR-Dropoff-Goal123",
  "priority": 10,
  "goal": "Goal123"
}
```

- To check if request failed:

  itk/cmd/job/dropoff/res/error

  Payload example:

```
{
  "type": "Dropoff",
  "id": "b81048e4071211efbfa0",
  "upd": 1714510645326,
  "status": "Error",
  "description": "no such goal: GoalNotInMap",
  "job_id": "AMR-Dropoff-Goal123",
  "priority": 10,
```

```
  "goal": "GoalNotInMap"
}
```

DataStore Value Request

Use this topic to configure the value update interval of a data store item.

Topic	QoS	System
itk/cmd/dataStore/subscribe/req/{dataStore-name}	1	Fleet Manager, AMR
itk/<amr-name>/cmd/dataStore/subscribe/req/<dataStore-name>	1	Fleet Manager (bridge)

Payload schema: JSON format

- interval (required): string

Example:

```
itk/cmd/dataStore/subscribe/req/ARAM_Uptime
{
    "interval": "1s"
}
```

For topic response, subscribe to the following:

- For topic forward confirmation:
  `itk/cmd/dataStore/subscribe/res/<dataStore-name>`
  Payload example:

```
{
  "type": "SubscriptionConfig",
  "id": "TripCompletedJobSegments",
  "upd": 1714658690834,
  "status": "Forwarded",
  "description": "Forwarded subscription request for TripCompletedJobSegments with
 interval 2s to ARAM server.",
  "interval": "2s"
}
```

- To check if request failed:
  `itk/cmd/dataStore/subscribe/res/error`

Additional Information

After a datastore value has been requested, the values are updated at specific intervals. Refer to *DataStore Values* on page 3-16 to receive datastore value updates and *DataStore List* on page 3-15 to receive a list of available datastore values.

ARCL Command Request

Use these topics to request MQTT API permitted ARCL commands.

Additional Information

Refer to *A-1 ARCL Commands* on page A-2 for a complete list of permitted ARCL commands.

Topics	QoS	System
itk/cmd/arcl/req	2	Fleet Manager, AMR
itk/<amr-name>/cmd/arcl/req	2	Fleet Manager (bridge)

Payload schema: JSON format

- command (required): string

Example:

```
{
    "command": "log"
}
```

For topic response, subscribe to the following:

- For success confirmation:

  itk/cmd/arcl/res/<message-id>

  Payload example:

```
{
  "type": "ArclRequest",
  "id": "d9a54724071311ef80be",
  "upd": 1714511130903,
  "status": "Forwarded",
  "description": "Command was forwarded to ARCL server. Subscribe to `itk/dt/arcl/update` for updates.",
  "command": "say Hello!"
}
```

- To check if request failed:

  itk/cmd/arcl/res/error

  Payload example:

```
{
  "type": "ArclRequest",
  "id": "bcd081f3121311ef8f22",
  "upd": 1715731345323,
  "status": "Error",
  "description": "Failed to write to ARCL server.",
  "command": "odometer"
}
```

Additional Information

Refer to *ARCL Update* on page 3-18 for more information about getting updates on requested ARCL commands.

Digital Output Switch

Use this topic to turn a specific digital IO on or off in an AMR.

Topic	QoS	System
itk/cmd/digOutputSwitch/req	2	AMR
itk/<amr-name>/cmd/digOutputSwitch/req	2	Fleet Manager (bridge)

Payload schema: JSON format

- dig_output_name (required): string
- switch (required): string
  - values allowed: "on" or "off" (case insensitive)

Example:

```
{
    "dig_output_name":"o1",
    "switch": "on"
}
```

For topic response, subscribe to the following:

- For success confirmation:
  itk/cmd/digOutputSwitch/res/<dig-Output-Switch-name>
  Payload example:

```
{
   "type": "DigitalOutputSwitch",
   "id": "3aecbf46065011ee983900",
   "upd": 1686290415080,
   "status": "Success",
   "description": "Digital output switch o1 is on",
   "dig_output_name": "o1",
   "switch": "on"
}
```

- To check if request failed:
  itk/cmd/digOutputSwitch/res/error

Additional Information

Refer to *Turn Digital Output On or Off* on page 3-29 for usage example on turning a digital output device on or off.

WaitTaskFail

Use this topic to trigger a failure of a wait task.

Topic	QoS	System
itk/cmd/waitTaskFail/req	0	Fleet Manager, AMR
itk/<amr-name>/cmd/waitTaskFail/req	0	Fleet Manager (bridge)

Payload schema: any format

Example:

Use any content in the payload or leave it empty to trigger a failure.

{}

For topic response:

- Subscribe to the topic below and in the response payload check for wait_state.

`itk/dt/robot/status`

WaitTaskCancel

Use this topic to cancel a wait task.

Topic	QoS	System
`itk/cmd/waitTaskCancel/req`	2	Fleet Manager, AMR

Payload schema: any format
Example:
Use any content in the payload or leave it empty to cancel a wait task.
{}
For topic response:

- Subscribe to the topic below and in the response payload check for wait_state.
  `itk/dt/robot/status`

3-1-2 Data Topics

The MQTT API client publishes messages to a set of data topics to update or display data about the Fleet Manager or the AMRs. Subscribe to the data topics to ensure you receive updates or information about the system. Data topics are characterized by the prefix `itk/dt`.
Subscribe to topic:
`itk/dt/<dt-context>`
The payload will be generated by the MQTT API since all data topics are subscribe only.
The payload in data topics has the following format:

- type (string): describes the message type.
  Example:
  - "QueueUpdate"
  - "RobotQMState"
- id (string): unique identifier for the message.
  Example:
  - "5e208bea0ccb11ef9858"
  - "CPUUse" (more concrete ids are used for dataStore values).
- upd (integer): timestamp update.
  Example:
  - 1715150506886 (represents Wednesday, May 8, 2024 6:41:46.886 AM.
- arguments: depending on the message type, one or multiple arguments will be used to provide data.

The following sections describe each data topic with an example.

Job Status

Use this topic to obtain status updates about ongoing jobs. Updates are provided when there is a change in the job state.

Additional Information

Refer to *Job Request* on page 3-4 to request a job and trigger updates.

Topic	QoS	Retain	System
itk/dt/job/status/<job-id>	1	False	Fleet Manager

Example:


```
{
    "type": "JobStatus",
    "id": "5b57ae72c1de11ee8519",
    "upd": 1706905125443,
    "segment_id": "DROPOFF4",
    "job_id": "JOB1",
    "priority": 27,
    "segment_state": "Pending",
    "segment_sub_state": "ContainsLinkedReason",
    "goal": "Goal101",
    "robot": "",
    "queued_time": "2024-02-02T10:18:45-05:00",
    "completed_time": "",
    "fail_count": 0,
    "job_type": "Multi",
    "linked_segment_id": "PICKUP3",
    "job_state": "Pending",
    "linked_job_id": "",
    "segment_state_str": "PICKUP3"
}
```


Robot Pose

Use this topic to obtain AMR's x, y, and theta coordinates. Updates are provided when robot position or orientation changes.

Topic	QoS	Retain	System
itk/dt/robot/pose	0	True	AMR
itk/<amr-name>/dt/robot/pose	0	True	Fleet Manager (bridge)

Example:


```
{
    "type": "RobotPosition",
    "id": "971229f2c08911ee970200",
    "upd": 1706758767266,
    "x": 58855,
    "y": 9134,
    "th": 1
}
```

QM State

Use this topic to obtain information about the AMR queue manager state and sub- state. Updates are provided when there is a change in the robot state.

Topic	QoS	Retain	System
itk/dt/robot/qmState	0	True	AMR
itk/<amr-name>/dt/robot/qmState	0	True	Fleet Manager (bridge)

Example:

```
{
    "type": "RobotQMState",
    "id": "971231fec08911ee970200",
    "upd": 1706758767266,
    "state": "Available",
    "sub_state": "Docked"
}
```

Robot Status

Use this topic to obtain information about the AMR status, temperature (in °C), and battery state of charge.

Topic	QoS	Retain	System
itk/dt/robot/status	0	True	AMR
itk/<amr-name>/dt/robot/status	0	True	Fleet Manager (bridge)

Example:

```
{
  "type": "RobotStatus",
  "id": "ea4fdb1812a111efb9bb",
  "upd": 1715781610191,
  "status": "Parked",
  "mode": "Goal seeking",
  "extended": "Parked\nParked at Standby1\nDone driving",
  "wait_state": "Not waiting",
  "state_of_charge": 50,
  "temperature": 36
}
}
```

Map Goals

Use this topic to obtain a list of map goals. Updates are provided when goals are added or removed from the map.

Topic	QoS	Retain	System
itk/dt/goals	2	True	Fleet Manager

Example:

```
{
    "type": "Goals",
    "id": "9712989cc08911ee970200",
    "upd": 1706758767269,
    "goals": [
        "Goal1",
        "Goal2"
        ]
    }
}
```

Robot List

Use this topic to obtain a list of AMRs in the fleet. Updates are provided when an AMR is added to the fleet or removed from it.

Topic	QoS	Retain	System
itk/dt/robot/list	2	True	Fleet Manager

Example:

```
{
    "type": "RobotList",
   "id": "d0491e2a5fbc11edabe60",
   "upd": 1667950007129,
   "robots": [
      {
         "name": "Sim248",
         "ip": "xxx.xxx.23.248"
      },
      {
         "name": "Sim247",
         "ip": "xx.xxx.23.247"
     }
    ]
}
```

DataStore List

Use this topic to obtain a list of datastore values.

Topic	QoS	Retain	System
itk/dt/dataStore/list	1	True	AMR, Fleet Manager
itk/<amr-name>/dt/dataStore/list	1	True	Fleet Manager (bridge)

Example:


```
{
    "type": "DataStoreNameList",
    "id": "2ff69f1cbf6411ee8519",
    "upd": 1706632751739,
    "data_store_list": [
      "RobotIP",
      "SNG",
      "ARAM",
      "MARC",
      "Odometer(KM)",
      "OdometerKM_MM",
      "HourMeter",
      "Laser_1_Filtered_v2",
      "Idle",
      "DockingState",
      "IsForced",
      "TipAngle",
      "IsTipped",
      "Queue_ID",
      "Queue_Job_ID",
      "RobotState",
      "RobotSubState",
      "JobSegmentState",
      "CancelledInProgressJobSegments",
      "FailedJobSegments",
      "RobotRotVel",
      "RobotHeading",
      "GyroOffset",
      "TimeSinceGyroCentered",
      "Mem_megs",
      "Mem_rss_megs",
      "Mem_v_megs"
    ]
}
```


Additional Information

Refer to *DataStore Value Request* on page 3-9 to subscribe to a specific datastore value and *DataStore Values* on page 3-16 to receive datastore value updates.

DataStore Values

Use this topic to obtain datastore value updates. Updates are provided at requested intervals.
JSON Format:

Topic	QoS	Retain	System
itk/dt/dataStore/value/<dataStore-name>	0	False	AMR, Fleet Manager
itk/<amr-name>dt/dataStore/value/<dataStore-name>	0	False	Fleet Manager (bridge)

Example:

```
{
    "type": "DataStoreValue",
    "id": "DateAndTime",
    "upd": 1706733866661,
    "value": "Wed Jan 31 10:44:26 2024"
}
```

PLC Format:

Topic	QoS	Retain	System
itk/dt/dataStore/value/<dataStore-name>/bytes	0	False	AMR, Fleet Manager

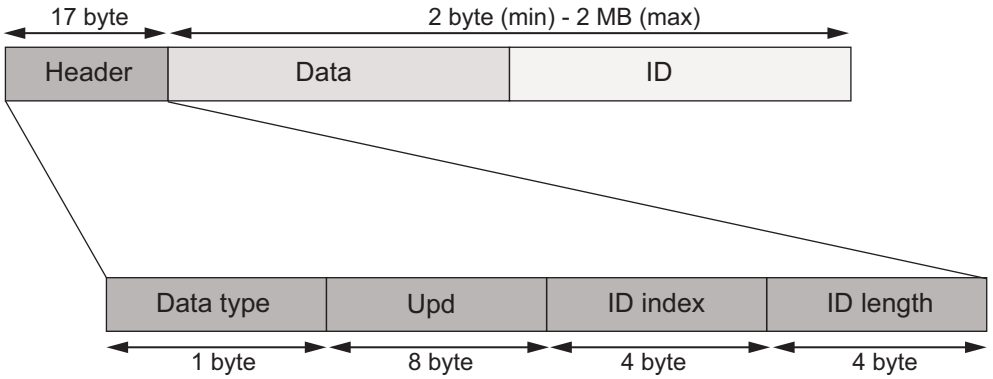
The following PLC types are supported. The format for PLC data type follows the IEC 61131-3 standard.

PLC Type	Data Type Code (hex)	Data Size
SINT	x01	1 byte
LINT	x02	8 bytes
REAL	x03	4 bytes
STRING	x04	Variable length

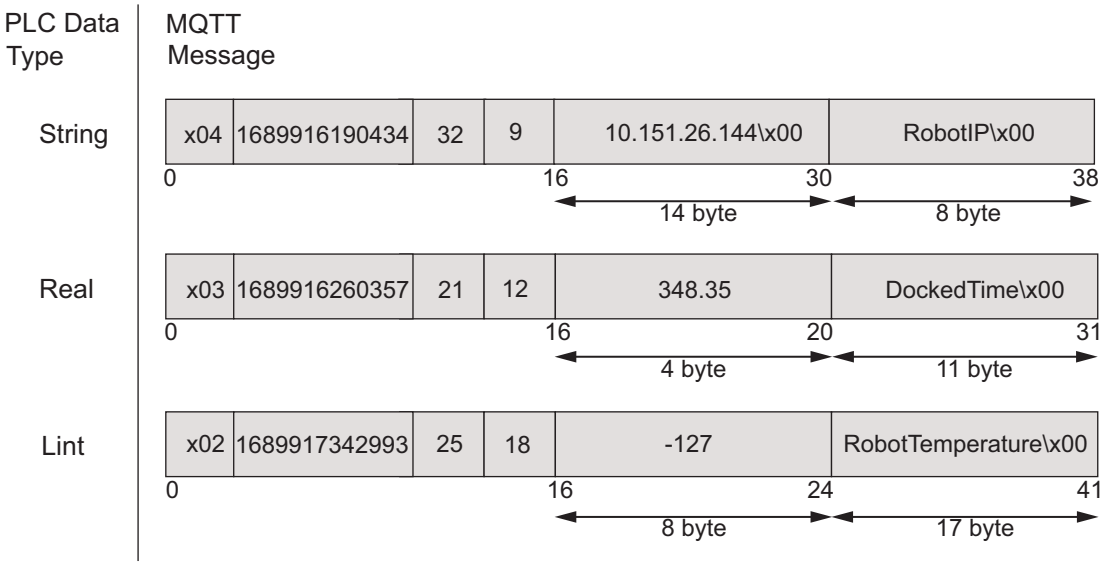
Payload Format:

- Header: Fixed 17 byte, provides details to decode payload.
  - Data type: Fixed 1 byte, provides code with expected Data type
  - Upd: Fixed 8 byte, provides upd time
  - ID index: Fixed 4 byte, provides start byte index value for ID
  - ID length: Fixed 4 byte, provides length of ID
- Data: DataStore value information
- ID: Variable length string with datastore value name

Schematic representation of payload format and header segments is shown below:



Payload Example:



Additional Information

Refer to *DataStore Value Request* on page 3-9 to subscribe to a specific datastore value and *DataStore List* on page 3-15 to receive a list of datastore values.

ARCL Update

Use this topic to obtain updates about MQTT permitted ARCL command requests.

Topic	QoS	Retain	System
itk/dt/arcl/update	1	False	Fleet Manager, AMR
itk/<amr-name>/dt/arcl/update	1	False	Fleet Manager (bridge)

Example:

```
{
    "type": "arclUpdate",
    "id": "5c2bc3ddfc3a11eebd81",
    "upd": 1713329007973,
    "message": "Temperature: -127"
}
```

Additional Information

Refer to *ARCL Command Request* on page 3-9 for more information.

3-1-3 Custom Topic

Use the userspace topic tree to implement custom solutions.

Topic	QoS	System
itk/userspace/#	2	Fleet Manager, AMR

Payload schema: No schema or data format is enforced.
Example topic:
`itk/userspace/controller/cmd/door`

Best Practices

Follow these best practices when implementing custom solutions using the *userspace* topic.

Best Practice	Details
Each topic must contain at least one character.	
Never start with a forward slash.	This introduces unnecessary topic level with the zero character. Example: `/some/topic`
Never end with a forward slash.	Example: `/some/topic/`
Follow a consistent naming scheme.	Topics are case sensitive.
Never use space character.	Poses readability issues for the programmer. UTF-8 has many different white space types. Example: `api/dataStore values`
Keep topics short and concise.	This optimizes network traffic and conserves resources, specifically with resource-constrained devices.
Never use non-ASCII characters, and avoid non-printable characters.	Non-ASCII UTF-8 characters may display incorrectly. Use lowercase letters, numbers, and dashes.
Include a unique identifier or the client id.	This enforces message identification and authorization. Identifies the message sender. Example: Client with ID *client1* can publish to `client1/status` but not to `client2/status`
Topics must embrace extensibility.	This facilitates scaling applications.
Be specific.	Promotes clarity and enables the utilization of advance MQTT features such as retained messages. Example: `hv100/bld1518/basement/hvac719` (goes from general to specific). `myhome/livingroom/humidity` instead of `myhome/livingroom`
Never expose sensitive information.	To avoid exposing security vulnerabilities of your application.

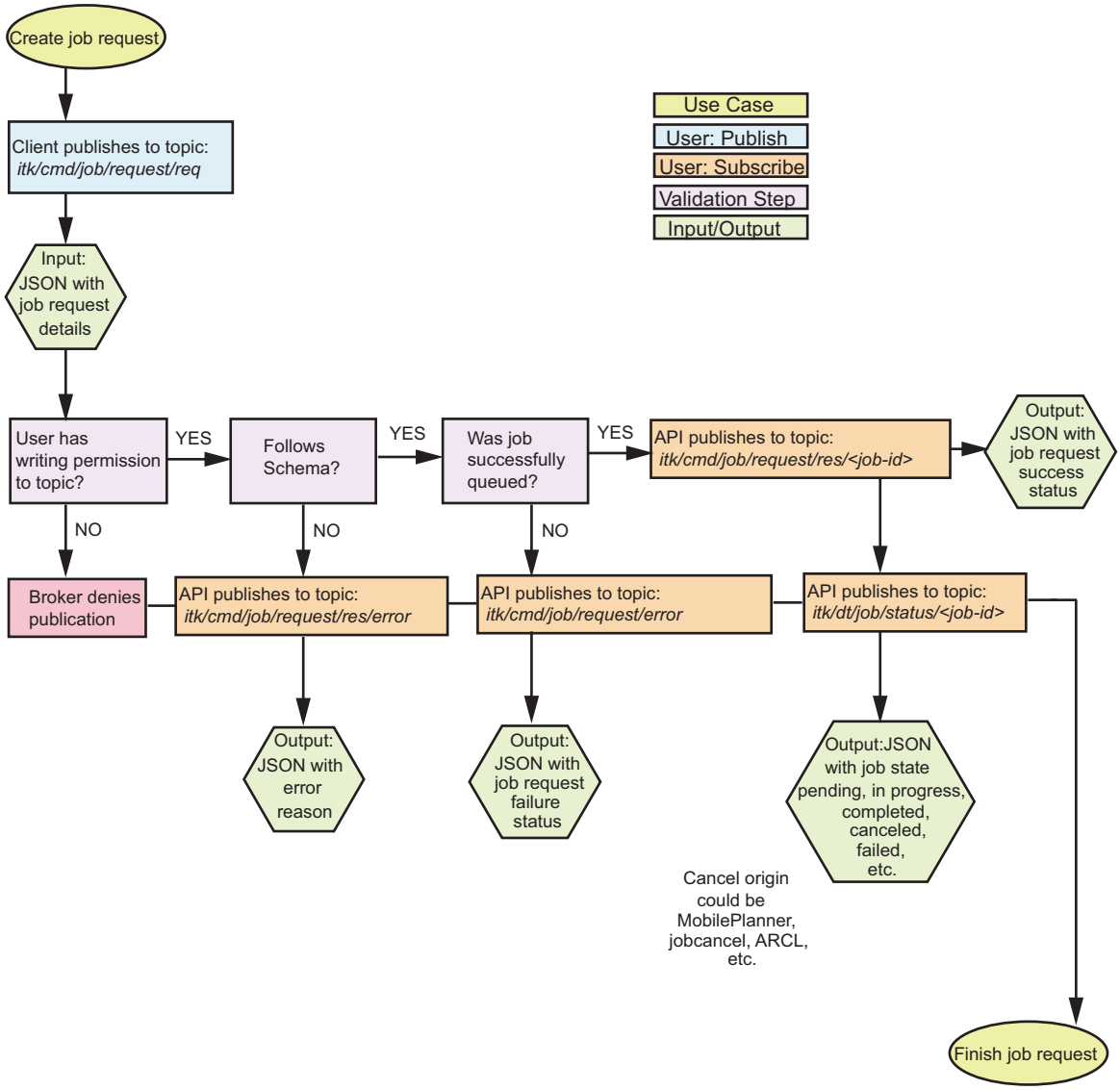
3-2 Use Case

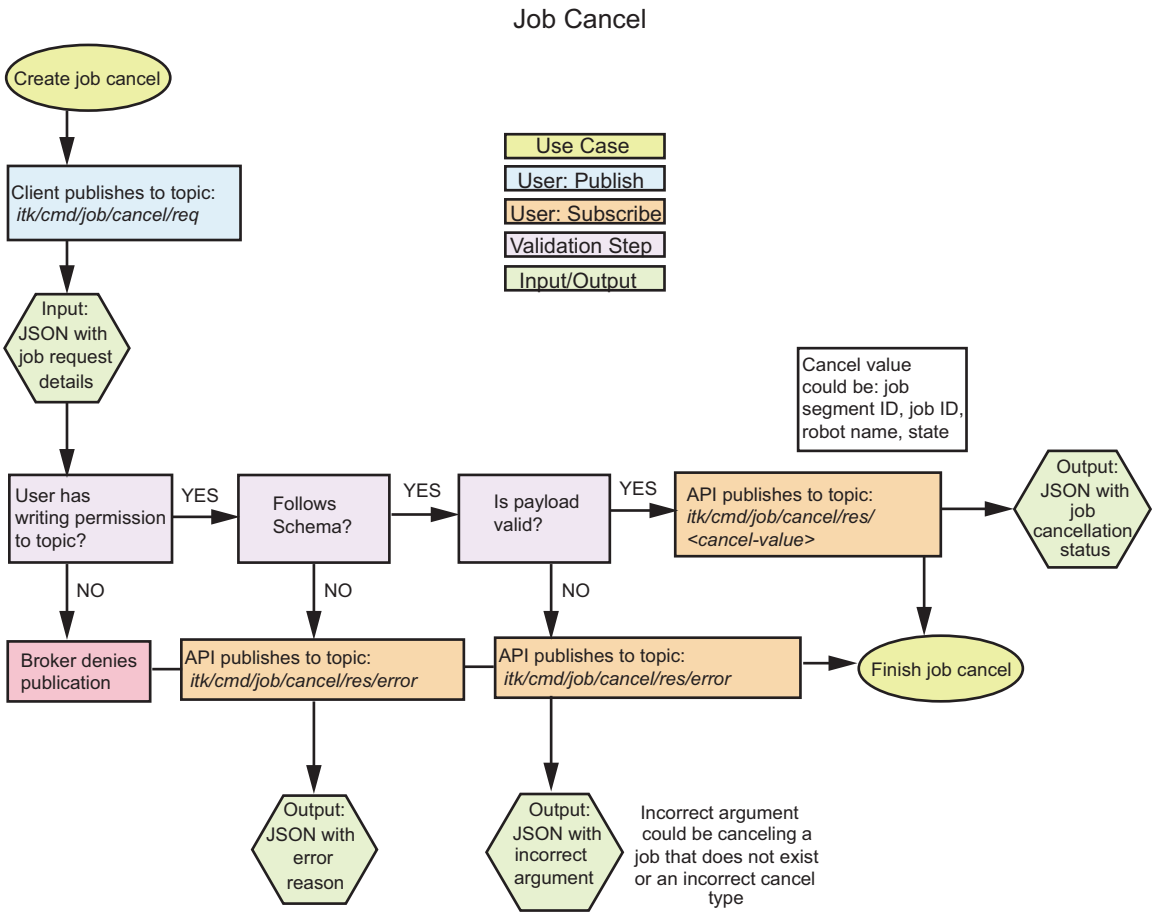
The following sections give details about topic flow charts and usage examples for the MQTT API.

3-2-1 Flow Charts

Use the following topic flow charts for understanding the general steps in processing a job request and job cancel tasks.

Job Request





3-2-2 Usage Examples

Topic usage examples for requesting, modyfying, and canceling a job, and switching digital output devices on or off are provided in the sections below.

Additional Information

When using command topics for sending a request, it is important to subscribe to the corresponding response and data topics before publishing the request. This ensures messages are not missed.

Request a Job

Follow the steps below to request a multi-segment (pickup and drop off) job.

1. Subscribe to the following topic to confirm success or failure of the job before publishing the job request.
   `itk/cmd/job/request/res/+`

2. Subscribe to the following topic to monitor the job (Job ID: JOB1) after it has been successfully queued to confirm completion.
   `itk/dt/job/status/<job-id>`

Complete the above steps before publishing the job request to ensure messages are not missed.

3 Create the payload for the job request (JOB1) following the correct schema for the topic below and publish. Use QoS 2 to ensure the job is reliably requested once.

`itk/cmd/job/request/req`

Payload example:

```
{
   "default_priority":false,
   "details":[
      {
         "goal":"Goal117",
         "priority":17,
         "segment_type":"Pickup"
      },
      {
         "goal":"Goal101",
         "priority":27,
         "segment_type":"Dropoff"
      }
   ],
   "job_id":"JOB1"
}
```

4 Check the messages received in the response topic:

`itk/cmd/job/request/res/+`

5 If the job request was successful, you will receive a message from the following topic.

`itk/cmd/job/request/res/JOB1`

Example response when you subscribe to the above topic:

```
{
  "type": "JobRequest",
  "id": "3efab6b20d5f11efa13d",
  "upd": 1715203220856,
  "status": "Success",
  "description": "Successfully requested job with id JOB1",
  "job_id": "JOB1",
  "default_priority": false,
  "details": [
    {
      "goal": "Goal117",
      "segment_type": "Pickup",
      "priority": 17
    },
    {
      "goal": "Goal101",
      "segment_type": "Dropoff",
      "priority": 27
```

```
    }
  ]
}
```

6 If there was an error in the job, you will receive a message from the topic below.

`itk/cmd/job/request/res/error`

Example response when you subscribe to the topic above:

```
{
  "type": "JobRequest",
  "id": "8a79888e0d5f11efa13d",
  "upd": 1715203347430,
  "status": "Error",
  "description": "jobid in use: JOB1.",
  "job_id": "JOB1",
  "default_priority": false,
  "details": [
    {
      "goal": "Goal1177",
      "segment_type": "Pickup",
      "priority": 17
    },
    {
      "goal": "Goal101",
      "segment_type": "Dropoff",
      "priority": 27
    }
  ]
}
```

7 After job success is confirmed, you will receive job status updates in the following data topic:

`itk/dt/job/status/JOB1`

Example job status when you subscribe to the above topic:

```
{
  "type": "JobStatus",
  "id": "8b4b66240d5f11efa8b9",
  "upd": 1715203348180,
  "segment_id": "DROPOFF58",
  "job_id": "JOB1",
  "priority": 27,
  "segment_state": "Completed",
  "segment_sub_state": "None",
  "goal": "Goal101",
  "robot": "Sim144",
  "queued_time": "2024-05-08T13:20:20-04:00",
  "completed_time": "2024-05-08T13:22:28-04:00",
  "fail_count": 0,
  "job_type": "Multi",
  "linked_segment_id": "PICKUP57",
```

```
  "job_state": "Completed",
  "linked_job_id": "",
  "segment_state_str": "None"
}
```


Modify Job

Modify a job (in progress or pending) by changing the goal or priority using the Job Modify topic.

Schema	Type	Value
segment_id (required)	String	Pickupxxx or Dropoffxxx
modify_type (required)	Integer	5: Goal 6: Priority
modify_value (required)	Integer or string	Integer if modify_type = 6 String if modify_type = 5

● Modify by Goal or Priority

Follow the steps below to modify a job by changing its goal name or its priority.

1 Subscribe to the following topic to confirm success or failure of the request before publishing the job modification.

`itk/cmd/job/modify/res/+`

2 Subscribe to the following topic to monitor the job (Job ID: JOB1) status after it has been successfully modified.

`itk/dt/job/status/+`

Complete the above steps before publishing the job modify request to ensure messages are not missed.

3 Assume a job has been requested (JOB1) and it is currently queued.
Take note of the segment ID. The segment ID gets generated when a new job is created. you need to identify the job segment that will be modified. In this example the segment ID is "PICKUP57".
Subscribe to:

`itk/cmd/job/status/JOB1`


```
{
  "type": "JobStatus",
  "id": "3f6de8f80d5f11efa8b9",
  "upd": 1715203220899,
  "segment_id": "PICKUP57",
  "job_id": "JOB1",
  "priority": 17,
  "segment_state": "Pending",
  "segment_sub_state": "None",
  "goal": "Goal117",
  "robot": "",
  "queued_time": "2024-05-08T13:20:20-04:00",
  "completed_time": "",
```

```
  "fail_count": 0,
  "job_type": "Multi",
  "linked_segment_id": "",
  "job_state": "Pending",
  "linked_job_id": "",
  "segment_state_str": "None"
}
```

4 Create the payload following the correct schema for the topic below and publish. Use QoS 2 to ensure the job is modified reliably once.

`itk/cmd/job/modify/req`

To modify by goal, for "modify_type" in the payload, enter integer value 5:

```
{
    "segment_id": "PICKUP57",
    "modify_type": 5,
    "modify_value": "Goal102"
}
```

To modify by priority, for "modify_type" in the payload, enter integer value 6:

```
{
    "segment_id": "PICKUP57",
    "modify_type": 6,
    "modify_value": 20
}
```

5 Check the messages received in the response topic:

`itk/cmd/job/modify/res/+`

6 If the job modification was successful, you will receive a message from the following topic:

`itk/cmd/job/modify/res/PICKUP57`

Example response when you subscribe to the above topic:

```
{
  "type": "JobModify",
  "id": "62e645c20d6011efa13d",
  "upd": 1715203710100,
  "status": "Success",
  "description": "Successfully modified segment PICKUP57 with value Goal102",
  "segment_id": "PICKUP57",
  "modify_type": 5,
  "modify_value": "Goal102"
}
```

7 If there was an error in the job modification, you will receive a message from the topic below.

`itk/cmd/job/modify/res/error`

Example response when you subscribe to the topic above:

```
{
  "type": "JobModify",
  "id": "8722f2d20d6011efa13d",
  "upd": 1715203770879,
```

```
  "status": "Error",
  "description": "job segment already completed: PICKUP59",
  "segment_id": "PICKUP57",
  "modify_type": 5,
  "modify_value": "Goal102"
}
```


8 After success is confirmed, you will receive job status updates in the following data topic:
`itk/dt/job/status/NewJob1`
Example of a modified job status when you subscribe to the above topic:


```
{
  "type": "JobStatus",
  "id": "632aaef60d6011efa8b9",
  "upd": 1715203710353,
  "segment_id": "PICKUP57",
  "job_id": "JOB1",
  "priority": 10,
  "segment_state": "InterruptedByModify",
  "segment_sub_state": "None",
  "goal": "Goal105",
  "robot": "Sim144",
  "queued_time": "2024-05-08T13:28:24-04:00",
  "completed_time": "",
  "fail_count": 0,
  "job_type": "Multi",
  "linked_segment_id": "",
  "job_state": "Modifying",
  "linked_job_id": "",
  "segment_state_str": "None"
}
```


Cancel Job

Cancel a job (in progress or pending) by its segment ID, job ID, robot name, or job state. The table below shows the value assigned for each cancel type.

Cancel Type	Value
Segment ID	1
Job ID	2
Robot name	3
State	4

● Job Cancel Procedure

Follow the steps below to cancel a job by its segment ID, job ID, robot name, or by the job state.

1 Subscribe to the following topic to confirm success or failure of the request before publishing the job cancellation.

itk/cmd/job/cancel/res/+

2 Subscribe to the following topic to monitor the job (Job ID: JOB1) status after it has been successfully canceled.

itk/dt/job/status/+

Complete the above steps before publishing the job cancel request to ensure messages are not missed.

3 Assume a job was requested and it is currently queued. A pending job with job ID: JOB1. Subscribe to topic:

itk/cmd/job/status/JOB1

```
{
  "type": "JobStatus",
  "id": "3f6de8f80d5f11efa8b9",
  "upd": 1715203220899,
  "segment_id": "PICKUP57",
  "job_id": "JOB1",
  "priority": 17,
  "segment_state": "Pending",
  "segment_sub_state": "None",
  "goal": "Goal117",
  "robot": "",
  "queued_time": "2024-05-08T13:20:20-04:00",
  "completed_time": "",
  "fail_count": 0,
  "job_type": "Multi",
  "linked_segment_id": "",
  "job_state": "Pending",
  "linked_job_id": "",
  "segment_state_str": "None"
}
```

4 Create the payload following the correct schema for the topic below and publish. Use QoS 2 to ensure the job is canceled reliably once.

itk/cmd/job/cancel/req

To cancel by segment ID, for "cancel_type" in the payload, enter integer value 1.

```
{
  "cancel_type" : 1,
  "cancel_value": "PICKUP57",
  "echo_msg": "JOB1",
  "cancel_reason": "Obstacle"
}
```

To cancel by job ID, for "cancel_type" in the payload, enter integer value 2.

```
{
  "cancel_type" : 2,
  "cancel_value": "JOB1"
}
```

To cancel by robot name, for "cancel_type" in the payload, enter integer value 3.

```
{
  "cancel_type" : 3,
  "cancel_value": "Sim147"
}
```

To cancel by job state, for "cancel_type" in the payload, enter integer value 4.

```
{
  "cancel_type" : 4,
  "cancel_value": "Pending"
}
```

5 Check the messages received in the response topic:
`itk/cmd/job/cancel/res/+`

6 If the job cancellation was successful, you will receive a message from the following topic:
`itk/cmd/job/cancel/res/JOB1`

Payload example when you subscribe to the topic above:

```
{
  "type": "JobCancel",
  "id": "152cf0680d6211efa13d",
  "upd": 1715204439193,
  "status": "Success",
  "description": "Successfully cancelled job with value JOB1",
  "cancel_type": 2,
  "cancel_value": "JOB1",
  "echo_msg": "",
  "cancel_reason": ""
}
```

7 If the job cancellation was unsuccessful, you will receive a message from the following topic:
`itk/cmd/job/cancel/res/error`

Payload example when you subscribe to the above topic

```
{
  "type": "JobCancel",
  "id": "32e029680d6211efa13d",
  "upd": 1715204488727,
  "status": "Error",
  "description": "jobId error: JOB1",
  "cancel_type": 2,
  "cancel_value": "JOB1",
  "echo_msg": "",
  "cancel_reason": ""
}
```

8 After success is confirmed, you will receive job status updates in the following data topic:
`itk/dt/job/status/JOB1`

Example of a canceled job status when you subscribe to the above topic:

```
{
  "type": "JobStatus",
```

```
    "id": "15bddb780d6211efa8b9",
    "upd": 1715204439449,
    "segment_id": "PICKUP57",
    "job_id": "JOB1",
    "priority": 10,
    "segment_state": "Cancelled",
    "segment_sub_state": "ContainsCancelReason",
    "goal": "Goal117",
    "robot": "Sim144",
    "queued_time": "2024-05-08T13:40:25-04:00",
    "completed_time": "2024-05-08T13:40:39-04:00",
    "fail_count": 0,
    "job_type": "Multi",
    "linked_segment_id": "",
    "job_state": "Cancelled",
    "linked_job_id": "",
    "segment_state_str": "None"
}
```


Turn Digital Output On or Off

A usage example is provided for turning a digital output device on or off using the MQTT API. The table below lists all the topics used for this procedure.

Topics	Action	Description
`itk/cmd/digOutputSwitch/req`	Publish	Publish to switch a digital output device on or off.
`itk/cmd/digOutputSwitch/res/+`	Subscribe	Subscribe to receive a success or error confirmation about the digital output switch request. Receive success message on the topic: `itk/cmd/digOutputSwitch/res/<dig-output-device>` Receive errors on the topic: `itk/cmd/digOutputSwitch/res/error`
`itk/cmd/arcl/req`	Publish	Publish an ARCL command, in this example it's used to obtain the list of output devices and their current state.
`itk/cmd/arcl/res/+`	Subscribe	Subscribe to receive a success or error confirmation about the ARCL command sent. Receive success message on the topic: itk/cmd/arcl/res/<id> Receive errors on the topic: `itk/cmd/arcl/res/error`
`itk/dt/arcl/update`	Subscribe	Subscribe to receive ARCL updates after an ARCL command is sent.

● On/Off Procedure

Follow the procedure below to obtain a list and the state of available digital output devices and then turn them on or off.

1 Before using a command topic, subscribe to its corresponding response topic and data topic to ensure the requests are processed successfully.

1) Subscribe to topic:
   `itk/cmd/digOutputSwitch/res/+`
   for receiving response when you publish to the topic:
   `itk/cmd/digOutputSwitch/req`
2) Subscribe to topic:
   `itk/cmd/arcl/res/+`
   for receiving response when you publish to the topic:
   `itk/cmd/arcl/req`
3) Subscribe to the data topic below to receive ARCL command responses:
   `itk/dt/arcl/update`

2 Use the ARCL command and data topics to obtain available output devices for switching on or off.

1) Publish to the topic below with the following example payload:
   `itk/cmd/arcl/req`

```
{
  "id": "c0ba5010165e11ef9cbc00"
  "command": "outputList"
}
```

2) Verify a message was received on the topic below with the status "Forwarded".
   `itk/cmd/arcl/res/c0ba5010165e11ef9cbc00`

```
{
  "type": "ArclRequest",
  "id": "c0ba5010165e11ef9cbc00",
  "upd": 1716203368794,
  "status": "Forwarded",
  "description": "Command was forwarded to ARCL server. Subscribe to `itk/dt/arcl/update` for updates.",
  "command": "outputList"
}
```

3) Process all the messages received on the ARCL data topic below. The following is an example message:
   `itk/dt/arcl/update`

```
{
  "type": "ArclUpdate",
  "id": "c0c9e35e165e11ef9cbc00",
  "upd": 1716203368896,
  "message": "OutputList: o2"
}
```

3 Choose an output device to check its current state.

1) Publish to the topic below with the following example payload:
   `itk/cmd/arcl/req`

```
{
  "id": "403fa718165f11ef9cbc00",
  "command": "outputQuery o2"
}
```

2) Verify a message was received on the topic below with the status "Forwarded".

```
itk/cmd/arcl/res/403fa718165f11ef9cbc00
{
  "type": "ArclRequest",
  "id": "403fa718165f11ef9cbc00",
  "upd": 1716203582739,
  "status": "Forwarded",
  "description": "Command was forwarded to ARCL server. Subscribe to `itk/dt/arcl/update` for updates.",
  "command": "outputQuery o2"
}
```

3) Finally, process the messages received on ARCL data topic below. The following is an example message:

```
itk/dt/arcl/update
{
  "type": "ArclUpdate",
  "id": "404f230a165f11ef9cbc00",
  "upd": 1716203582840,
  "message": "Output: o2 on"
}
```

You have obtained the list and state of available digital output devices in steps 2 and 3 above.

4 To change the state of the digital output switch, follow the steps below.

1) Publish to the topic below to turn the device "o2" from "on" to "off" with the following example payload.

```
itk/cmd/digOutputSwitch/req
{
  "id": "8aba5266165f11efa84800",
  "dig_output_name": "o2",
  "switch": "off"
}
```

2) Verify a message was received on the topic below with the status "Success".

```
itk/cmd/digOutputSwitch/res/o2
{
  "type": "DigitalOutputSwitch",
  "id": "8aba5266165f11efa84800",
  "upd": 1716203707794,
  "status": "Success",
  "description": "Digital output switch o2 is off",
  "dig_output_name": "o2",
  "switch": "off"
}
```

5 Repeat step 3 to check the current state for device "o2" and confirm that it was switched "off" successfully.

3-2-3 Error Message Examples

The following sections describe MQTT API example error messages and their payload format.

Decode Payload Error

This error is triggered when the payload received from the user is not encoded as valid JSON.
Payload example

```
{
  "type": "PayloadDecodeError",
  "id": "207baeda117e11efa481",
  "upd": 1715667088030,
  "status": "Error",
  "description": "Payload is not UTF-8 encoded."
}
```

Governor Error

This error is triggered by message governor restrictions. Example, when the message rate limit is exceeded.
Payload example

```
{
  "type": "GovernorError",
  "id": "207baeda117e11efa481",
  "upd": 1715667088030,
  "status": "Error",
  "description": "Payload was not processed due to message rate limit"
}
```

Schema Error

This error is triggered when the decoded payload does not follow the entity schema.
Payload example below describes the schema error.

```
{
  "type": "SchemaError",
  "id": "026bb179117e11ef9f96",
  "upd": 1715667037593,
  "status": "Error",
  "description": "'switch' is a required property",
  "payload": {
    "dig_output_name": "o1"
  }
}
```

Logic Error

This error is triggered when logic is invalid. Example, using a job id that is already in use.

Payload example


```
{
  "type": "JobRequest",
  "id": "f3327722114311efa30a",
  "upd": 1715631301652,
  "status": "Error",
  "description": "no such goal: NotInMapGoal.",
  "job_id": "JOB730a422",
  "default_priority": false,
  "details": [
    {
      "goal": "NotInMapGoal",
      "segment_type": "Pickup",
      "priority": 10
    }
  ]
}
```

4

Troubleshooting

This section describes MQTT API error codes, error messages, and corrective actions for troubleshooting purposes.

4-1 Error Codes and Messages

The error codes and messages are sent from the MQTT broker to the client. The errors are sent as MQTT error codes when you implement your client programmatically. If you use a client GUI application such as MQTT Explorer, error messages are displayed in the connection window.

4-1-1 Error Codes

The following table lists a few sample error codes and corrective actions.

Error Code		Details	Corrective Action
Decimal	Hex		
0	0x0	No error	
2	0x2	Connection Refused: Identifier rejected. The client identifier is correct, but not allowed by the server.	1. Ensure correct user is created in SetNetGo Fleet Accounts page and correct credentials are set. Refer to *2-4-1 Set Username and Password* on page 2-7 for more inofrmation. 2. Ensure username is a valid MQTT API username. 3. Reset the ExternalComms application. Refer to *2-2 Software Management* on page 2-3 for more information.
3	0x3	Connection Refused: Server Unavailable. Network connection has been made but MQTT service is unavailable.	
4	0x4	Connection Refused: bad username or password.	
5	0x5	Connection Refused: authorization error.	
6	0x6	Connection Timeout.	Ensure the ExternalComms application is running. If not, restart the application. Refer to *2-2 Software Management* on page 2-3.
7	0x7		
8	0x8		
9	0x9		
10	0xa		
11	0xb		
12	0xc		
21	0x15		
22	0x16		
23	0x17		
17	0x11	Subscription failure.	Ensure the client is subscribed to the correct topic.

4-1-2 Connection Problems

The following table lists a few connection issue scenarios when using the MQTT API and remedial actions.

Problem	Corrective Action
Frequent disconnections.	1. Ensure each MQTT client ID is unique. Refer to *2-5-2 Connect Using MQTT Client GUI* on page 2-10. 2. If you implemented your MQTT client programmatically, set the *Keep Alive* value between 60 and 120 seconds. Do not exceed 120 seconds or the MQTT broker will disconnect the client. Refer to the code snippet in *2-5 Establish Connection* on page 2-10 for more information.
Delay in message response.	1. Adjust QoS to best fit your network capacity: • Use QoS 0 for high frequency messages • Avoid QoS 2 for non-critical messages • Use QoS 1 when duplication is not an issue 2. Reduce message frequency. Refer to *2-7 Messaging Limits* on page 2-18 for more information. 3. Follow best subscription practices: avoid utilizing wildcard "#" to reduce overhead in MQTT broker message propagation to all subscribed clients. 4. Close unused connections to reduce MQTT Broker *Keep Alive* overhead. 5. If you implemented your MQTT client programmatically, adjust *Keep Alive* as needed. Avoid high frequency intervals to reduce network overhead.
No data or command response from MQTT API. MQTT client fails to read or write to topics from broker.	1. Ensure the MQTT API and the ARAMCentral applications are running. Refer to *2-2 Software Management* on page 2-3 for more information. If arnet or arcl connections are not ready, messages will not be sent downstream. 2. Ensure the ExternalComms version is 2.0 or greater. 3. Ensure MQTT client is subscribed to the correct topic. 4. Ensure the username apiControl is used for client utilized for writing to topics. Refer to *2-4 User Management and Access Control* on page 2-7 for more information about topic permissions. 5. Ensure correct QoS is set to guarantee message delivery. 6. Restart MQTT API application. Refer to *2-2 Software Management* on page 2-3 for more information.
SSL related error messages: • Self signed certificate in certificate chain • wrong version number • protocol interpretation error	1. Ensure certificate validation is disabled in client. Refer to *2-5-2 Connect Using MQTT Client GUI* on page 2-10 for more information. 2. Ensure TLS version is 1.2 or greater. 3. If implementing the MQTT client programmatically, ensure there is cipher compatibility. Set TLS context to EOS default.

A

Appendix

A-1 ARCL Commands

The following is a list of MQTT API permitted ARCL commands.
Refer to *Advanced Robotics Command Language AMR Reference Guide (Cat. No. I617)* for more information about these commands.

```
applicationFaultClear
applicationFaultQuery
applicationFaultSet
arclSendText
configAdd
configParse
configStart
dock
doTask
doTaskInstant
etaRequest
executeMacro
extIOAdd
extIOInputUpdate
extIOInputUpdateBit
extIOInputUpdateByte
extIOOutputUpdate
extIOOutputUpdateBit
extIOOutputUpdateByte
extIORemove
faultsGet
getConfigSectionInfo
getConfigSectionList
getConfigSectionValues
getDataStoreFieldList
getDataStoreFieldValues
getDataStoreGroupInfo
getDataStoreGroupList
getDataStoreGroupValues
getDateTime
getGoals
getMacros
getRoutes
goalDistanceRemaining
goto
gotoPoint
gotoRouteGoal
help
inputList
```

```
inputQuery
localizeToPoint
log
odometer
odometerReset
oneLineStatus
outputList
outputOff
outputOn
outputQuery
patrol
patrolOnce
patrolResume
pauseTaskCancel
pauseTaskState
payloadQuery
payloadQueryLocal
payloadRemove
payloadSet
payloadSlotCount
payloadSlotCountLocal
play
popupSimple
pq
pql
pr
ps
psc
pscl
qc
qd
qf
qm
qmod
qp
qpd
qq
qs
qsc
qsr
queryDockStatus
queryFaults
queryMotors
queueCancel
```

```
queueCancelMultiSegment
queueDropoff
queueModify
queueMulti
queuePickup
queuePickupDropoff
queueQuery
queueShow
queueShowCompleted
queueShowRobot
say
setPayload
shutdown
status
stop
tripReset
undock
waitTaskCancel
waitTaskState
waitTaskFail
```

OMRON Corporation **Industrial Automation Company**

Kyoto, JAPAN **Contact : www.ia.omron.com**

Regional Headquarters

OMRON EUROPE B.V.
Wegalaan 67-69, 2132 JD Hoofddorp
The Netherlands
Tel: (31) 2356-81-300 Fax: (31) 2356-81-388

OMRON ASIA PACIFIC PTE. LTD.
438B Alexandra Road, #08-01/02 Alexandra
Technopark, Singapore 119968
Tel: (65) 6835-3011 Fax: (65) 6835-3011

OMRON ELECTRONICS LLC
2895 Greenspoint Parkway, Suite 200
Hoffman Estates, IL 60169 U.S.A.
Tel: (1) 847-843-7900 Fax: (1) 847-843-7787

OMRON ROBOTICS AND SAFETY TECHNOLOGIES, INC.
4225 Hacienda Drive, Pleasanton, CA 94588 U.S.A.
Tel: (1) 925-245-3400 Fax: (1) 925-960-0590

OMRON (CHINA) CO., LTD.
Room 2211, Bank of China Tower, 200 Yin Cheng Zhong Road,
PuDong New Area, Shanghai, 200120, China
Tel: (86) 21-6023-0333 Fax: (86) 21-5037-2388

Authorized Distributor:

Cat. No. M107-E-01 0624

20569-080 A