OMRON

Fleet Operation Workspace Core Integration Toolkit

User's Manual

I637-E-02

Copyright Notice

Table of Contents

Revision History

Revision code	Date	Revised Content
01	June, 2019	Original release
02	July, 2020	Updates for FLOW 2.0 release

Chapter 1: Introduction

This document contains information that is necessary to use the Integration Toolkit facilitating integration between the Fleet Manager and the end user's client application.

Please read this document and make sure that you understand the functionality and performance of the Integration Toolkit before you attempt to use it with a fleet of AMRs. Read and understand all related manuals and safety guides before using the Integration Toolkit.

1.1 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, AMR's, and the EM2100 appliance.
- Personnel familiar with the Advanced Robotics Command Language (ARCL), RESTful Web Services, SQL, or RabbitMQ.

1.2 Abbreviations and Terminology

The following abbreviations and terminology will be used throughout this document.

Term	Description
AMR	Autonomous Mobile Robot
ARCL	Advanced Robotics Command Language
API	Application Programming Interface
Client Application	A warehouse management system, manufacturing execution system, enterprise resource planning system, or similar application that interacts with the Integration Toolkit.
EM2100 Appliance	Hardware appliance which connects all Omron autonomous mobile robots and runs the fleet management software.
Entity	A basic object in the Integration Toolkit that contains data sent or received to or from the Fleet Manager, such as a job or DataStore value.
Fleet Manager	Elements of the FLOW Core software that controls traffic, charging, job assignment, etc. and collaborates with the client application(s) to assign and manage tasks across the mobile robot fleet.
FLOW	Fleet Operations Workspace Omron's software suite that manages all autonomous mobile robot

Term	Description
	navigation, safety, and fleet management functions.
IntegrationDB	The primary database that includes all table views.
JSON	JavaScript Object Notation
Job	A basic activity for a robot to execute consisting of one or more Pickup or Dropoff segments.
REST	RESTful Web Services
SQL	Structured Query Language
URI	Uniform Resource Identifier

1.3 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.

```
{
        "example": "example",
        "Example": 10,
        "example_example": "p34",
}
```

Application Specific Placeholders

IP Address

The Fleet Manager IP Address is indicated with [IP] throughout this document as shown below.

https://**[IP]**:8443

cURL String Options

Flags may be required in cURL strings for user credentials or other options. These options are indicated as [options] throughout this document as shown below.

```
curl [options] -X POST "https://[IP]:8443/PickupDropoff" -H  "accept:
application/json; charset=utf-8" -H  "Content-Type: application/json;
charset=utf-8" -d "
{\"pickupGoal\":\"p5\",\"pickupPriority\":10,\"dropoffGoal\":\"p34\",\"dropoffPr
iority\":20}"
```

1.4 System Requirements

The Integration Toolkit has the following minimum system requirements.

- EM2100 appliance
- Fleet Operations Workspace software version 1.0.0 or higher

1.5 How to Get Help

Find additional information on the corporate website: http://www.ia.omron.com.

Related Manuals

There are additional manuals that describe how to program your fleet, reconfigure installed components, and add other optional equipment. These manuals provide information about safety, related products, advanced configurations, and system specifications.

Manual Title	Description
Mobile Robot LD Safety Guide (Cat. No. I616)	Describes safety information for OMRON LD series AMRs.
Mobile Robot HD Safety Guide (Cat. No. I647)	Describes safety information for OMRON HD-1500 AMRs.
LD Platform OEM User's Guide (Cat. No. I611)	Describes the installation, start-up, operation, and maintenance of the LD-60 and LD-90 AMRs.
LD-250 Platform User's Guide (Cat. No. I642)	Describes the installation, start-up, operation, and maintenance of the LD-250 AMR.
HD-1500 Platform User's Manual (Cat. No. I645)	Describes the installation, start-up, operation, and maintenance of the HD-1500 AMR
LD Platform Peripherals Guide (Cat. No. I613)	Covers peripherals for LD AMRs, such as the Touchscreen, Call/Door box, and Acuity Localization options.
Mobile Robots - HD Platform Peripherals Manual (Cat. No. I646)	Covers peripherals for HD AMRs, such as HAPS.
EM2100 Installation Guide (Cat. No. I634)	Describes the installation and initial configuration of an EM2100 appliance.
Fleet Operations Workspace Migration Guide (Cat. No. I636)	Describes the procedures for migrating your AMR from legacy to FLOW Core software, and from an EM1100 to an EM2100.
Fleet Operations Workspace Core User's Manual (Cat. No. I635)	Describes use of the EM2100 and the software that runs on it for managing a fleet of AMRs.
Fleet Simulator User's Manual (Cat. No. I641)	Describes the operation of the Fleet Simulator.
Fleet Operations Workspace/ EM2100 Migration Guide (Cat. No. I636)	Describes how to upgrade or downgrade a system between Legacy and FLOW Core solutions which includes software upgrade processes and tools as well as guidance on any necessary hardware changes.
Advanced Robotics Command Language Enterprise Manager Integration Guide (Cat. No. I618)	Describes the Advanced Robotics Command Language (ARCL) version for use with the EM2100 software. ARCL is a simple text-based command and response server used for integrating the Fleet Operations Workspace Core platform with an external automation system.

Manual Title	Description
Advanced Robotics Command Language Reference Guide (Cat. No. I617)	Describes the Advanced Robotics Command Language (ARCL), which is a simple, text-based language from which you can control Omron's AMRs.
LD Platform Cart Transporter User's Guide (Cat. No. I612)	Describes the operation and maintenance of the Cart Transporter AMR.

Chapter 2: Functions and Features

The Integration Toolkit is Omron's interface application that enables integration between the Fleet Manager and the end user's client application. This integration layer facilitates autonomous control for a fleet of AMRs using standard communication methods.

It facilitates full management and monitoring of all AMR job types such as pickup, dropoff, and multi-segment. The Integration Toolkit also allows tracking of AMR data directly and in real-time.

The Integration Toolkit has a flexible architecture to provide multiple communication channel options. These communication channels provide flexibility and choice in how a system interacts with an AMR fleet and the Fleet Manager.

> **NOTE:** The Integration Toolkit can operate in parallel with existing ARCL communication. The Integration Toolkit does not replace ARCL for direct AMR control (once it has reached a goal). Refer to the *ARCL Reference Guide - Mobile Robots* and the *ARCL Enterprise Manager Integration Guide* for more information.
>
> **IMPORTANT:** Access to the Integration Toolkit is not possible until a password is generated. Generate a password before attempting to use the Integration Toolkit. Refer to Integration Toolkit Password on page 15 for more details.

Communication Channels

The Integration Toolkit offers 3 different communication channels.

- RESTful Web Services
- SQL with PostgreSQL Database
- RabbitMQ

> **NOTE:** While it may be possible to achieve desired functionality with any one of the communication channels, using a combination of them is suggested for efficient integration with client applications.

2.1 RESTful Web Services

The REST communication channel is ideal for achieving real-time interaction with the Fleet Manager. With this communication channel, a system can create, modify, and cancel jobs on demand. The REST method allows access to other specific AMR data and can also provide job history query functions.

Structuring of externally transferred data with REST is implemented with JSON for both creating requests and receiving data from the system.

The REST communication channel provides encrypted and secure interaction with the Integration Toolkit over HTTPS. Refer to Security on page 15 for more information.

REST Communication Channel Advantages

- Low-latency interaction with the AMR fleet.
- Ad hoc access to DataStore values and job history data.
- REST is independent from platform or language type. REST does not rely on specific drivers or libraries.

REST Communication Channel Considerations

- The RESTful API is hosted on the EM2100 appliance using port 8443.
- The client needs to provide a JSON data format as input for all POST calls and may need a third party library to help create and process these strings.
- The SSL is implemented with a self-signed certificate which the client needs to trust.
- Most data entities can be accessed with the following paths:
  1. **/Stream:** opens up a connection to receive updates as the entity changes (real time).
  2. **/ByKey/{namekey}**: returns entity information for entity with associated with the given namekey.
  3. **/UpdatedSince?sinceTime={time millis}**: returns all updates of entities since time provided. A value of 0 can be used to get all updates (since epoch time).
- Certain data items are enumerated and correct syntax and schema of these items are critical for proper functionality. Refer to the schema and examples in this document for details.

  NOTE: Audit (enumerated) items can be found in some schema examples. These are reserved items that are used internally for the operation of the Integration Toolkit.

2.2 SQL with PostgreSQL

The SQL communication channel is ideal when database-level interaction with the AMR fleet is necessary. PostgreSQL is a relational database management system where *update*, *insert*, *select*, and *delete* commands are used to monitor and control an AMR fleet.

SQL Communication Channel Advantages

- Database-level interaction with an AMR fleet.
- Simple batch/bulk job creation.
- Complex SQL querying against the entire job history.

SQL Communication Channel Considerations

- When any action is requested through the SQL channel, there may be a delay of up to 5 seconds because the table or view in question must be polled by the system.
- PostgreSQL is accessed using the standard port 5432.
- Automation using the SQL communication channel is achieved through programming languages and associated libraries such as PostgreSQL JDBC driver for Java. When using the JDBC driver to connect, include the "sslmode=require" parameter in the connection URL like this: "jdbc:postgresql://[IP]/IntegrationDB?sslmode=require”.

- Even though the Fleet Manager is storing data and is keeping a full history of job data, it should not be considered secure in the way that a database server with redundant disk capabilities would be. The Fleet Manager can be used to obtain data which can be stored on an external system to act as a backup. The Fleet Manager's hard drive should be considered a single point of failure.
- Certain data items are enumerated and correct syntax and schema of these items are critical for proper functionality. Refer to the schema and examples in this document for details.

PostgreSQL Tables and Views

SQL includes access to both tables and views. There is post insert / update logic in these views which is executed, and controls how the data interacts with the system (in some cases impacting multiple tables).

Views have a naming convention ending with "_view" (data_store_item_view for example).

> **IMPORTANT:** Database interaction is only supported with views as described in this document. Modifying or directly updating tables or the schema in general is not supported.

2.3 RabbitMQ

The RabbitMQ communication channel is a robust method for monitoring and controlling an AMR fleet.

RabbitMQ provides a management mechanism to ensure that all messages are delivered even if a network problem occurs or a broker / client crashes. RabbitMQ can update a process running on an external system in real-time while also storing messages for future delivery if the external application becomes temporarily unavailable.

Structuring of externally transferred data with RabbitMQ is implemented with JSON for receiving data from the system.

RabbitMQ Communication Channel Advantages

- Asynchronous messaging with read / write capabilities.
- Incoming and outgoing message buffering.
- Simple monitoring for the AMR fleet.

RabbitMQ Communication Channel Considerations

- When using inbound channels (such as those used to create jobs), the system will need to provide JSON of the same format used for REST.
- Libraries must be used and code must be implemented to send JSON-formatted data to the RabbitMQ queue.
- A RabbitMQ management console is not available and some level of programming is required to facilitate interaction with the system when using this channel.
- Buffered messages expire after 12 hours. If the client application does not consume these buffered messages before they expire, they will be lost.

- Certain data items are enumerated and correct syntax and schema of these items are critical for proper functionality. Refer to the schema and examples in this document for details.

2.4 Software Management

SetNetGo provides an area to manage the Integration Toolkit software installation. The following software management functions are available.

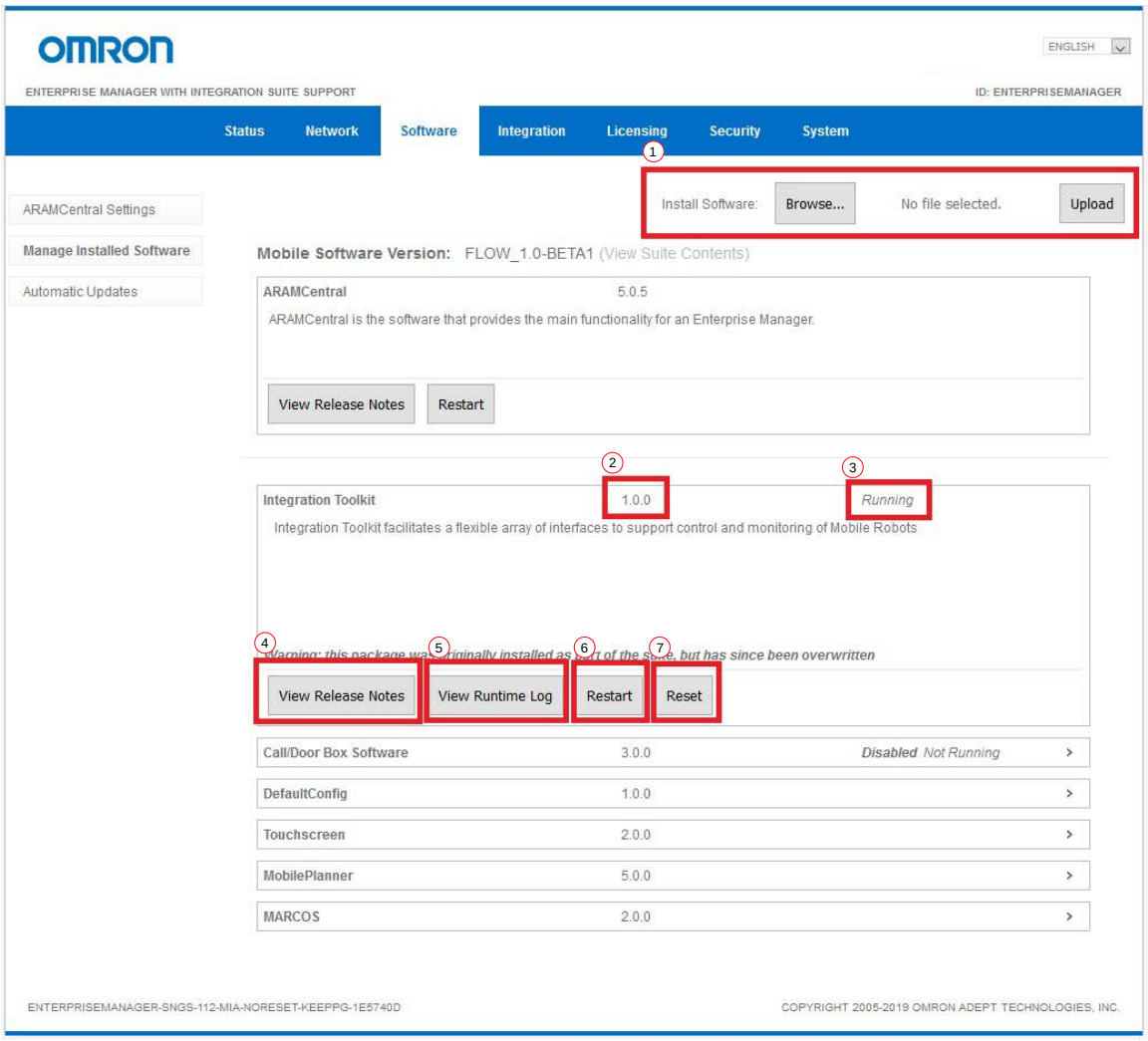

Figure 2-1. Integration Toolkit Software Management

Table 2-1. Integration Toolkit Software Management Descriptions

Item	Description
1	Select and upload a new FLOW Core software file (see note below).
2	The current version of the installed Integration Toolkit.

Item	Description
3	The operational status of the Integration Toolkit (Running or Not running).
4	Opens a dialog box to display the Integration Toolkit release notes.
5	Opens a dialog box to display the Integration Toolkit RunTime Log for diagnostic purposes.
6	Restarts the Integration Toolkit services (see note below).
7	Resets the Integration Toolkit to default values. **IMPORTANT:** This will delete and then rebuild the database. All data will be lost such as job history and current job data.

NOTE: Integration Toolkit functionality relies on ARAMCentral operation. Any action that stops the Integration Toolkit or ARAMCentral may have an impact on data integrity and functionality. Pausing fleet activity during a planned software stoppage is recommended.

2.5 Security

Integration Toolkit security is implemented in a common way for all communication channels. A self-signed certificate is used to establish an encrypted connection between the client application and the Integration Toolkit. A user id and password common to all communication channels are used for authentication.

This security mechanism is not optional and it is not possible to configure communications without this encrypted connection.

NOTE: If there are concerns for the secure transport of the self-signed certificate (since the client is not authenticating the certificate), then this certificate should be moved, loaded, and trusted manually.

Integration Toolkit Password

To generate a new password, access the SetNetGo interface of the Fleet Manager with MobilePlanner or a web browser. Refer to the Fleet Operations Workspace Core User's Guide for more information about accessing SetNetGo.

The username for the Integration Toolkit is always fixed as "toolkitadmin".

IMPORTANT: A password is not set for a new installation of the Integration Toolkit. Access to the Integration Toolkit is not possible until a password is generated. Generate a password before attempting to use the Integration Toolkit for the first time.

Use the following procedure to generate a new Integration Toolkit password.

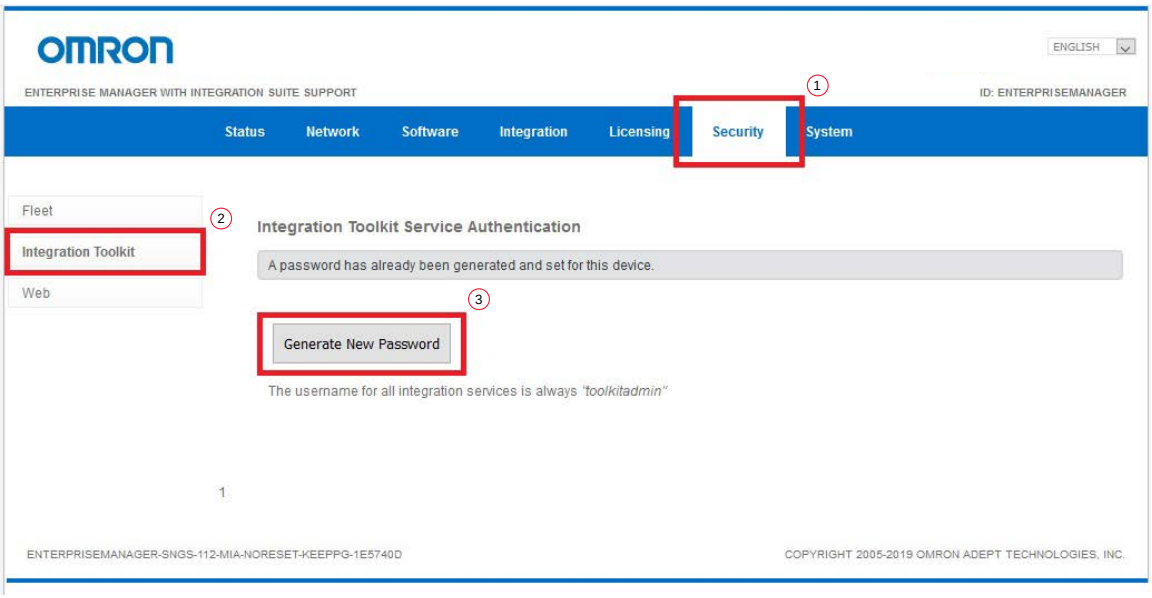

Figure 2-2. Set or Change Password in SetNetGo

1. After accessing the SetNetGo interface, click the Security tab.
2. Select the Integration Toolkit section within the security area.
3. Click the **Generate New Password** button.

   IMPORTANT: When the **Generate New Password** button is pressed, warning messages are displayed to indicate this action will cause a restart of the Integration and Fleet Management services. Proceed only if a restart is not problematic.

4. After the password is changed and the services restart, the new password is displayed for recording. Copy this password and keep it in a safe place for future use.

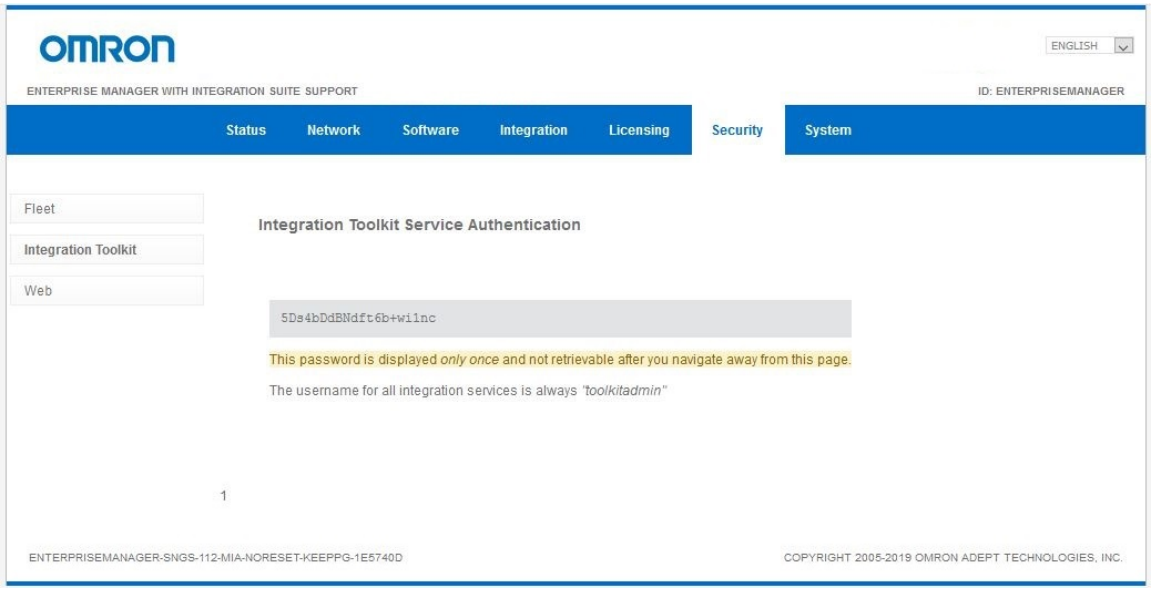

Figure 2-3. New Password Display

2.6 Namekey Concept

Namekey is a unique identifier used both internally and externally to track data entities. Making use of these will allow finer control and tracking from within the automated systems interacting with the Integration Toolkit.

The use of namekey provides the programmer with a non-ambiguous method to create and track entities. This provides a mechanism to retrieve status and states of requests. When using communication channels to create an entity (such as creating a job), providing a namekey is optional. If a namekey is not provided, the system will supply a self-generated namekey.

Where needed, the system will create unique namekeys (for example in job segment and history entries) to allow query functionality.

Chapter 3: Getting Started

Use the following examples to assist in creating a simple Pickup and Dropoff job request to an AMR associated with the Fleet Manager in your system.

> **NOTE:** These examples use goal names and namekeys that may not exist in your application. Use goal names and namekeys appropriate for your system.

3.1 PickupDropoff Job - REST Example

The following JSON object will create a new PickupDropoff job at the goals on the Flow Manager map of p5 and p34. This job will have a pickup priority level of 10 and a dropoff priority level of 20. TestPickup101 is the namekey of the entity.

The namekey and jobId will be automatically generated because they are not specified in the JSON command body parameters.

REST Command Details

- Method: POST
- Endpoint: https://[IP]:8443
- Resource: /PickupDropoff

JSON Command Body

To create a PickupDropoff job, issue a POST request https://[IP]:8443/PickupDropoff with the following JSON object body:

```
{
       "pickupGoal": "p5",
       "pickupPriority": 10,
       "dropoffGoal": "p34",
       "dropoffPriority": 20
}
```

cURL String

Issue the following cURL string to create a PickupDropoff job:

```
curl [options] -X POST "https://[IP]:8443/PickupDropoff" -H  "accept:
application/json; charset=utf-8" -H  "Content-Type: application/json;
charset=utf-8" -d "
{\"pickupGoal\":\"p5\",\"pickupPriority\":10,\"dropoffGoal\":\"p34\",\"dropoffPr
iority\":20}"
```

Response Example

```
{
       "code": 201,
        "entity": "PickupDropoff",
        "id": "7ddabddb-fcbe-4f49-b9a3-2bfd4f8d2e33",
        "message": "Entity created"
}
```

3.2 PickupDropoff SQL Example

The following SQL statement will create a new PickupDropoff job at the pre-designated locations of p5 and p34. This job will have a pickup priority level of 10 and a dropoff priority level of 20, and a new job identifier of Test1.

SQL Statement

To create a PickupDroppoff job, make the following statement:

```
INSERT INTO pickup_dropoff_View (pickup_goal, pickup_priority, dropoff_goal,
dropoff_priority, job_id) VALUES ('p5', '10', 'p34', '20', 'Test1');
```

Response

```
INSERT 0 1
```

3.3 RabbitMQ Python Examples

The Python examples below use RabbitMQ to interact with the Integration Toolkit. These examples can be modified for other functionality by changing the queue names and messages.

> **IMPORTANT:** These examples do not require a security certificate explicitly, but will use the one obtained from the Integration Toolkit server without an attempt at validation. The connection is still encrypted and the user id and password are used for authentication. cert_reqs=ssl.CERT_REQUIRED should be used when and SSL certificate is present and the programmer wishes to force its use.
>
> **NOTE:** Example values must be changed for application-specific conditions, as noted in the script comments below.
>
> **Additional Information**: All required libraries and packages must be installed before these examples can be used.
>
> The examples provided in this document were created using Python 3 and Pika 0.12.0.

Publish a Message to the inbound.PickupDroppoff Queue

The example below publishes a PickupDroppoff message to the inbound.PickupDropoff queue.

```
import os
import ssl
import pika
import logging

logging.basicConfig(level=logging.INFO)

def get_connection(host, username, password, cert_path):
    cp = pika.ConnectionParameters(
        host=host,
        port=5671,
        ssl=True,
        credentials=pika.PlainCredentials(username, password),
        ssl_options=dict(
            ssl_version=ssl.PROTOCOL_TLSv1,
            ca_certs=cert_path,
            cert_reqs=ssl.CERT_OPTIONAL
        )
    )
    return pika.BlockingConnection(cp)

def publish_message_to_queue(conn, inbound_queue, message):
    ch = conn.channel()
    assert ch.is_open
    ch.basic_publish(exchange='', routing_key=inbound_queue, body=message)
        #nameless ('') exchange should be used.
    print(" [x] published message %r to queue %r" % (message, inbound_queue))


if __name__ == '__main__':
    import sys
    user = 'toolkitadmin'
    password = 'uzJny0tb3FyhteE9BCuQ'
        # Change password to the generated password.
    FMIP = '10.151.26.181'
        # Change to the Fleet Manager IP

    conn = get_connection(
        FMIP,
        user,
        password,
        os.path.join(os.path.expanduser("~"), "Desktop\\itk-1-tests\\cert.crt")
        # Change the path to the path used in your application.
    )
    inbound_queue = 'inbound.PickupDropoff'
        # Change the queue to the desired inbound queue.
    message = '{ "pickupGoal": "Goal01", "dropoffGoal": "Goal02" }'
        # Change this message the the desired input value.
    publish_message_to_queue(conn, inbound_queue, message)
```

Consume Messages of the outbound.Job Queue

The example below consumes messages of the outbound.Job queue.

> **NOTE:** Set *no_ack =True* for acknowledging or *no_ack =False* to not acknowledge the consumed messages in subscribe_queue_and_print method.

```
import os
import ssl
import pika
import logging

logging.basicConfig(level=logging.INFO)

def get_connection(host, username, password, cert_path):
    cp = pika.ConnectionParameters(
        host=host,
        port=5671,
        ssl=True,
        credentials=pika.PlainCredentials(username, password),
        ssl_options=dict(
            ssl_version=ssl.PROTOCOL_TLSv1,
            ca_certs=cert_path,
            cert_reqs=ssl.CERT_OPTIONAL
        )
    )
    return pika.BlockingConnection(cp)

def subscribe_queue_and_print(conn, outbound_queue, no_ack=True):
        # Set no_ack =False to not acknowledge the consumed messages
    ch = conn.channel()
    assert ch.is_open
    print(ch.basic_get(outbound_queue))
    def print_body(ch, method, properties, body):
        print(" [x] %r" % body)
    ch.basic_consume(print_body, queue=outbound_queue, no_ack=no_ack)
    ch.start_consuming()


if __name__ == '__main__':
    import sys
    user = 'toolkitadmin'
    password = 'uzJny0tb3FyhteE9BCuQ'
        # Change password to the generated password.
    FMIP = '10.151.26.181'
        # Change to the Fleet Manager IP

    conn = get_connection(
        FMIP,
        user,
        password,
        os.path.join(os.path.expanduser("~"), "Desktop\\itk-1-tests\\cert.crt")
        # Change the path to the path used in your application.
    )
    outbound_queue = 'outbound.Job'
        # Change the queue to the desired outbound queue.
    subscribe_queue_and_print(conn, outbound_queue, True)
```

Chapter 4: DataStore

DataStore items are data points at the Fleet Manager and each AMR that can be evaluated for status and other monitoring purposes. They are specific to the Fleet Manager and each AMR. Each DataStore item describes the data point, i.e. data type, its name, source, etc.

Some examples of DataStore items are:

- Cumulative number of jobs an AMR has completed
- The distance driven by an AMR
- An AMR's current location
- An AMR's current battery state
- Information about the SetNetGo software version

At startup, the Integration Toolkit generates or updates its list of available DataStore items (a common list is kept for all AMRs). To see a complete list of all DataStoreItem items that are available on your system, issue a REST call or review the data_store_item_view view. Refer to DataStore on page 23 and SQL with PostgreSQL on page 12 for more information.

Values for these DataStore items are made available from the Integration Toolkit through a subscription model (available through all three channels) or with the DataStoreValueLatest REST command, which ignores subscription state and obtains the latest value from the entity in question. With the subscription model, a user specifies a subscription interval for a DataStore item and the Integration Toolkit will obtain DataStore values from the target entity at the subscription interval specified, and provide access to it via all communication channels.

Both SQL and REST will return the most current DataStore values from the last subscription period. Streaming interfaces will return entries every time a DataStore value changes. If there are no changes to the value of a DataStore item, the value will not be updated.

> **IMPORTANT:** Care should be taken to avoid high frequency subscription rates that are less than 1 second to prevent over-subscription on large numbers of data entities. Though testing has been done at higher levels, subscription rates producing 200 or more updated values per second should be avoided.

4.1 Common DataStore Use Cases

Common use cases when working with the DataStore are provided below.

Manually Obtain a List of Possible DataStore Items

This use case is typically executed during the initial programming phase to learn about what data can be obtained from the system. A programmer should query the data_store_item_view to get the most current list of DataStore items and details that can be accessed during the automated run time.

This information can also be obtained with a REST call (/DataStoreItem) but the SQL query method provides an easier way for a programmer to view this information.

Obtain a Single, Current Value for use in the Automated System

This use case is similar to the existing ARCL functionality. With this functionality, the Integration Toolkit communicates with an AMR or the Fleet Manager and provides the latest data, ignoring any subscription states.

This can be accomplished with the REST channel using the /DataStoreValueLatest path. Values obtained this way are published on the RabbitMQ queue and sent to the database in the same way they would be if the value was subscribed to. This is the most common use-case for obtaining DataStore values.

Obtain Values with Streaming (http or RabbitMQ)

This use case involves the need for DataStore values to be sent to an application under a set schedule with any of the stream options, and involves the need to create a subscription in advance. Refer to Subscribing to DataStore Values on page 27. After the subscription is created, the value is automatically updated at the defined subscription interval.

Values that have a subscription can also be obtained with SQL and REST channels, but these calls will only return the value which was last obtained.

4.2 DataStore Model

The DataStore model consists of the SubscriptionConfig, DataStoreValue, and DataStoreItem components.

DataStoreItem

DataStoreItem is used to obtain information about the individual items available on the Fleet Manager or AMR for information retrieval.

SubscriptionConfig

SubscriptionConfig is used to command the Integration Toolkit to obtain values from the AMRs and the Fleet Manager to make them available for use.

Refer to Subscribing to DataStore Values on page 27 for more information.

DataStoreValue

DataStoreValue provides the latest recorded value of DataStore items. This value is returned as a String and conversion may be necessary before use. The DataStoreItem entity can be used to indicate the value type. Refer to Obtaining DataStore Values on page 28 for more information.

4.3 Obtaining Information about DataStore Items

The following table describes DataStore item schema.

Table 4-1. DataStore - Data Item Details

Item	Details	Data Type
namekey	Predefined string for the DataStore item's namekey. The AMR name will be appended for AMR-specific values, as shown below. • AMR-specific value - "namekey:AMR name" • Fleet Manager value - "namekey"	String
itemId	DataStore item internal ID (reserved for internal use)	Integer
source	Fleet Manager - empty non-Fleet Manager - AMR name	String
category	Item category in Fleet Manager.	
groupName	DataStore group name in Fleet Manager.	
groupDescr	DataStore description of a group.	
itemName	DataStore item name.	
displayName	DataStore display name.	
type	DataStore item type (string, long, integer, double, or boolean).	
description	DataStore item description.	

Using REST

Use the following REST calls to obtain information about DataStore items.

Table 4-2. DataStore Item REST Calls

Method	Resource	Function
GET	/DataStoreItem/ByKey/{namekey}	Get DataStoreItem by namekey.
GET	/DataStoreItem/UpdatedSince?sinceTime={time millis}	Get a list of DataStoreItem entities that have been updated since the given time.
GET	/DataStoreItem/BySource/{Source}	Get a list of DataStoreItem entities filtered by source.
GET	/DataStoreItem/ByItemName/{ItemName}	Get a list of DataStoreItem entities filtered by itemName.
GET	/DataStoreItem/ByType/{Type}	Get a list of DataStoreItem entities filtered by type.
GET	/DataStoreItem/ByDisplayName/{DisplayName}	Get a list of DataStoreItem entities filtered by displayName.

Method	Resource	Function
GET	/DataStoreItem/ByGroupName/{GroupName}	Get a list of DataStoreItem entities filtered by groupName.
GET	/DataStoreItem/ByCategory/{Category}	Get a list of DataStoreItem entities filtered by category.

DataStore Item JSON Schema

```
{
        "namekey": "string",
        "upd": {"millis": long},
        "itemId": integer,
        "source": "string",
        "category": "string",
        "groupName": "string",
        "groupDescr": "string",
        "itemName": "string",
        "displayName": "string",
        "type": "string",
        "description": "string"
}
```

DataStore Item JSON Example

Response example after GET request https://[IP]:8443/DataStoreItem/ByItemName/DateAndTime

```
{
        "namekey": "DateAndTime",
        "upd": {"millis": "1545173147124"},
        "itemId": 3,
        "source": "",
        "category": "System",
        "groupName": "DateAndTime",
        "groupDescr": "the human readable time (note that this can leap forwards or
         backwards if the time is changed)",
        "itemName": "DateAndTime",
        "displayName": "DateAndTime",
        "type": "string",
        "description": "the human readable time (note that this can leap forwards or
         backwards if the time is changed)","
}
```

cURL Command String Example

To get information about the DateAndTime DataStore item:

```
curl [options] -X GET "https://[IP]:8443/DataStore/ByKey/DateAndTime" -H
"accept: application/json; charset=utf-8"
```

Using SQL

To obtain information about DataStore items:

```
SELECT * FROM data_store_item_view;
```

4.4 Subscribing to DataStore Values

Subscribing to DataStore values is the process of alerting the system that you care about certain DataStore items.

The following table describes DataStore subscription item schema.

> **NOTE:** The /DataStoreValueLatest REST call does not require a subscription. Refer to Obtaining DataStore Values on page 28 for more information.

Table 4-3. DataStore Subscription - Data Item Details

Item	Details	Data Type
namekey	Predefined string for the DataStore item's namekey.	String
subscriptionInterval	Subscription Interval Units: ms, s, m, h, d Syntax: "1s", "2m", etc. Minimum value: 200 ms Default value: 0 ms **NOTE:** A setting of 0 will turn OFF the subscription.	

Using REST

Use the following REST calls to get or set a subscription.

Table 4-4. DataStore Subscription REST Calls

Method	Resource	Function
GET	/SubscriptionConfig/ByKey/{namekey}	Get SubscriptionConfig by namekey.
GET	/SubscriptionConfig/UpdatedSince?sinceTime={time millis}	Get a list of SubscriptionConfig entities that have been updated since the given time.
GET	/SubscriptionConfig/Stream	Listen for all SubscriptionConfig updates.
PUT	/SubscriptionConfig	Update SubscriptionConfig.

> **NOTE:** Integration Toolkit creates an entry for each DataStore item upon startup. The DataStore subscription only updates existing entries and this is why no POST method is available.

DataStore Subscription JSON Schema

```
{
        "namekey": "string",
        "audit": {
                "namekey": "string",
                "crt": {
                "millis": "long"
                },
                "upd": {
                "millis": "long"
                },
        "ver": integer
                },
        "subscriptionInterval": "string"
}
```

DataStore Subscription JSON Example

To subscribe to the ARAM DataStore item's value with 1 second updates with a PUT request https://[IP]:8443/SubscriptionConfig :

```
{
        "namekey": "ARAM",
        "subscriptionInterval": "1s"
}
```

cURL Command String Example

To subscribe to the ARAM DataStore item with 1 second updates with a POST request:

```
curl [options] -X PUT "https://[IP]:8443/SubscriptionConfig" -H  "accept:
application/json; charset=utf-8" -H  "Content-Type: application/json;
charset=utf-8" -d "{\"namekey\":\"ARAM\",\"subscriptionInterval\":\"1s\"}"
```

Using SQL

Updating a subscription interval using SQL is accomplished by updating the 'subscription_ interval' column of the 'subscription_config_view' view as shown below.

```
UPDATE subscription_config_view SET subscription_interval = '1s' WHERE namekey =
'ARAM';
```

> **NOTE:** This view can also be used to ascertain the current value of these settings.

Using RabbitMQ

outbound.SubscriptionConfig

inbound.SubscriptionConfig

4.5 Obtaining DataStore Values

The following table describes DataStore value item schema.

Table 4-5. DataStore - Value Item Details

Item	Details	Data Type
namekey	Predefined string for the DataStore item's namekey. The AMR name will be appended for AMR-specific values, as shown below. • AMR-specific value - "namekey:AMR name" • Fleet Manager value - "namekey"	String
value	Current value.	

Using REST

Use the following REST calls to get DataStore values.

Table 4-6. DataStore Value REST Calls

Method	Resource	Function
GET	/DataStoreValue/ByKey/{namekey}	Get DataStoreValue by namekey.
GET	/DataStoreValue/UpdatedSince?sinceTime={time millis}	Get a list of DataStoreValue entities that have been updated since the given time.
GET	/DataStoreValue/Stream	Listen for DataStoreValue updates.

Table 4-7. DataStoreValueLatest Item REST Calls

Method	Resource	Function
GET	/DataStoreValueLatest/{DataStore item name}	Return (without subscription) a one-time value for the DataStore item named.
GET	/DataStoreValueLatest/{DataStore item name}:{AMR name}	Return (without subscription) a one-time value for the DataStore item named on the AMR named.
GET	/DataStoreValueLatest/{DataStore item name}:*	Return (without subscription) a one-time value for the DataStore item named on all AMRs. This method will create a 2-second delay while all values are collected.

DataStore Value JSON Schema

```
{
       "namekey": "string",
        "upd": {
          "millis": "long"
        },
        "value": "string"
  }
```

DataStore Value JSON Example

Response example after GET request https://[IP]:8443/DataStoreValue/ByKey/ARAM

```
{
        "namekey": "ARAM",
        "upd": {
          "millis": "1553902317602"
        },
        "value": "5.0.2"
}
```

cURL Command String Example

To get information about the ARAM DataStore value (subscribed):

```
curl [options] -X GET "https://[IP]:8443/DataStoreValue/ByKey/ARAM" -H  "accept:
application/json; charset=utf-8"
```

To get information about the ARAM DataStore value (not subscribed, ad hoc):

```
curl [options] -X GET "https://[IP]:8443/DataStoreValueLatest/ARAM" -H "accept:
application/json; charset=utf-8"
```

Using SQL

To get information about the ARAM DataStore value:

```
SELECT * FROM data_store_value_view WHERE namekey='ARAM';
```

Using RabbitMQ

Queues exist for each DataStore item (outbound.datastore.ARAM, for example) while changes for all values can be tracked at outbound.DataStoreValue.

Chapter 5: Jobs

The Integration Toolkit supports the following job functions.

- Job creation methods: pickup, pickupdropoff, dropoff, and job request.
- The modification of jobs on the segment level (goal and priority changes).
- The cancellation of entire jobs.
- The cancellation (deletion) of single segments of multi-segment jobs.
- If a job request is created without a namekey or jobId, the Integration Toolkit will automatically generate unique values for these data items.

  NOTE: Creating a dropoff job overrides the Fleet Manager's AMR selection logic. For this reason, use this job type selectively.

  NOTE: A job request has the same functionality as the queueMulti ARCL command. This job type will queue the AMR for multiple pickups and dropoffs at multiple goals.

Job Creation Steps

Job creation is a two-step process:

1. A job request is made and a confirmation reply is sent after the Integration Toolkit accepts the request.

   NOTE: A confirmation reply is sent with REST only.

2. After the Integration Toolkit accepts the job request, a call is made to the Fleet Manager to initiate the job. When the job is initiated, job and job segment entries become available in the Integration Toolkit, allowing the job to be tracked in real time.

5.1 Common Job Creation Use Cases

Common use cases when working with job creation are provided below.

Create a Single Job

This use case involves the creation of a job by specifying only the goal(s). If the system does not need to track the job or verify status, using any one of the communication channels can create a job with a single call (or SQL row insertion).

Create a Job and Verify it was Accepted by the Fleet Manager

After a job request is created, the status of it can be obtained by checking the SQL entry or by running a GET call on the same path's /ByKey URI using the namekey originally returned with the call.

Create and Track a Job

This can be accomplished by generating a unique namekey and supplying it in the job creation step as the JobId. This namekey can then be used to obtain information about the job as the Fleet Manager processes it (refer to Monitoring of Jobs on page 42 for more information).

Alternatively, a system could create a job without a JobId and then query the job creation method with the Fleet Manager to obtain the assignedJobId which can then be used to track the job.

5.2 Job Creation

Information is provided below for the various job creation methods.

Creating Pickup Jobs

This section provides details for generating new Pickup jobs or deleting existing Pickup jobs. It also describes how to monitor existing Pickup jobs. These functions can be used to manage Pickup jobs for the AMR fleet.

The following table describes the Pickup item schema.

Table 5-1. Pickup Item Schema

Item	Details	Data Type
namekey	Unique identifier of a Pickup job entity. Optional for POST/insert/publish method. If omitted, the Integration Toolkit auto-generates.	String
jobId	JobId to assign to the job. Optional for POST/insert/publish method.	
goal	Name of the pickup goal. Required with POST/insert/publish method.	
priority	Priority of the pickup segment. Optional for POST/insert/publish method. If omitted, Fleet Manager assigns default priority.	Integer
assignedJobId	JobId assigned to the job. If JobId was not provided when the job was created, the Fleet Manager automatically creates the assignedJobId.	String
status	"Success" or failure message from the Queuing Manager.	

Pickup Job Using REST

Use the following REST calls to generate and delete existing Pickup jobs. These calls can also be used to get information for Pickup jobs queued to the Integration Toolkit.

Table 5-2. Pickup Job REST Call Resources

Method	Resource	Function
POST	/Pickup	Create Pickup job.
GET	/Pickup/UpdatedSince?sinceTime={time millis}	Get a list of Pickup job entities that have been updated since the given time.
GET	/Pickup/Stream	Listen for all Pickup job updates.
GET	/Pickup/ByKey/{namekey}	Get Pickup by namekey.
GET	/Pickup/ByJobId/{JobId}	Get a list of Pickup entities filtered by JobId.
GET	/Pickup/ByStatus/{Status}	Get a list of Pickup entities filtered by Status.
GET	/Pickup/ByAssignedJobId/{AssignedJobId}	Get a list of Pickup entities filtered by AssignedJobId.
DELETE	/Pickup/{namekey}	Delete Pickup by namekey (see note below).

NOTE: The DELETE method is only required if a namekey needs to be reused.

Pickup Job JSON Schema

```
{
        "namekey": "string",
        "audit": {
                "namekey": "string",
        "crt":  {
                "millis": "long"
                },
        "upd": {
                "millis": "long"
                },
        "ver": integer
                },
        "goal": "string",
        "priority": integer,
        "jobId": "string",
        "status": "string",
        "assignedJobId": "string"
}
```

Pickup Job JSON Example

To create a pickup job with a POST request https://[IP]:8443/Pickup :

```
{
        "namekey": "PickupJob1",
        "goal": "Goal1",
        "priority": 10,
        "jobId": "TestJob1"
}
```

cURL Command String Example

To create a pickup job with a POST request:

```
curl [options] -X POST "https://[IP]:8443/Pickup" -H  "accept:
application/json; charset=utf-8" -H  "Content-Type: application/json;
charset=utf-8" -d "
{\"namekey\":\"PickupJob1\",\"goal\":\"Goal1\",\"priority\":10,\"jobId\":\"Tes
tJob1\"}"
```

Pickup Job Using SQL

To create a pickup job:

```
INSERT INTO pickup_view (namekey, goal, priority, job_id) VALUES (
'PickupJob1','Goal1', 10, 'TestJob1');
```

Pickup Job Using RabbitMQ

inbound.Pickup

outbound.Pickup

Creating PickupDropoff Jobs

This section provides details for generating new PickupDropoff jobs or deleting existing PickupDropoff jobs. It also describes how to monitor existing PickupDropoff jobs. These functions can be used to manage PickupDropoff jobs for the AMR fleet.

The following table describes the PickupDropoff item schema.

Table 5-3. PickupDropoff Item Schema

Item	Details	Data Type
namekey	Unique identifier of a PickupDropoff job entity. Optional for POST/insert/publish method. If omitted, the Integration Toolkit auto-generates.	String
jobId	JobId to assign to the job. Optional for POST/insert/publish method.	
pickupGoal	Name of the pickup goal. Required with POST/insert/publish method.	
pickupPriority	Priority of the pickup segment. Optional for POST/insert/publish method. If omitted, Fleet Manager assigns default priority.	Integer
dropoffGoal	Name of the dropoff goal.	String

Item	Details	Data Type
	Required with the POST/insert/publish method.	
dropoffPriority	Priority of the dropoff segment. Optional for POST/insert/publish method. If omitted, Fleet Manager assigns default priority.	Integer
assignedJobId	JobId assigned to the job. If JobId was not provided when the job was created, the Fleet Manager automatically creates the assignedJobId.	String
status	"Success" or failure message from the Queuing Manager.	String

PickupDropoff Job Using REST

Use the following REST calls to generate and delete existing PickupDropoff jobs. These calls can also be used to get information for PickupDropoff jobs queued to the Integration Toolkit.

Table 5-4. PickupDropoff Job REST Call Resources

Method	Resource	Function
POST	/PickupDropoff	Create PickupDropoff.
GET	/PickupDropoff/UpdatedSince?sinceTime={time millis}	Get a list of PickupDropoff entities that have been updated since the given time.
GET	/PickupDropoff/Stream	Listen for all PickupDropoff job updates.
GET	/PickupDropoff/ByKey/{namekey}	Get PickupDropoff by namekey.
GET	/PickupDropoff/ByJobId/{JobId}	Get a list of PickupDropoff entities filtered by JobId.
GET	/PickupDropoff/ByStatus/{Status}	Get a list of PickupDropoff entities filtered by Status.
GET	/PickupDropoff/ByAssignedJobId/{AssignedJobId}	Get a list of PickupDropoff entities filtered by AssignedJobId.
DELETE	/PickupDropoff/{namekey}	Delete PickupDropoff by namekey.

PickupDropoff Job JSON Schema

```
{
        "namekey": "string",
        "pickupGoal": "string",
        "pickupPriority": integer,
        "dropoffGoal": "string",
        "dropoffPriority": integer,
        "jobId": "string",
        "status": "string",
        "assignedJobId": "string"
}
```

PickupDropoff Job JSON Example

To create aPickupDropoff job with a POST request https://[IP]:8443/PickupDropoff :

```
{
        "namekey":"PickupDropoff1",
        "pickupGoal": "Goal1",
        "pickupPriority": 10,
        "dropoffGoal": "Goal2",
        "dropoffPriority": 20,
        "jobId": "TestJob1"
}
```

cURL Command String Example

To create a PickupDropoff job with a POST request:

```
curl [options] -X POST "https://[IP]:8443/PickupDropoff" -H  "accept:
application/json; charset=utf-8" -H  "Content-Type: application/json;
charset=utf-8" -d "
{\"namekey\":\"PickupDropoff1\",\"pickupGoal\":\"Goal1\",\"pickupPriority\":10,
\"dropoffGoal\":\"Goal2\",\"dropoffPriority\":20,\"jobId\":\"TestJob1\"}"
```

PickupDropoff Job Using SQL

To create a PickupDropoff job:

```
INSERT INTO pickup_dropoff_view (namekey, pickup_goal, pickup_priority,
dropoff_goal,dropoff_priority, job_id) VALUES ( 'PickupDropoff1','Goal1',
10, 'Goal2', 20, 'TestJob1');
```

PickupDropoff Job Using RabbitMQ

inbound.PickupDropoff

outbound.PickupDropoff

Creating Dropoff Jobs

This section provides details for generating new Dropoff jobs or deleting existing Dropoff jobs. It also describes how to monitor existing Dropoff jobs. These functions can be used to manage Dropoff jobs for the AMR fleet.

The following table describes the Dropoff item schema.

> **NOTE:** Dropoff jobs represent a way to circumvent the Queue Management functionality of the Fleet Manager and should be avoided where possible. This feature may be deprecated in future versions.

Table 5-5. Dropoff Item Schema

Item	Details	Data Type
namekey	Unique identifier of a Dropoff job entity. Optional for POST/insert/publish method. If omitted, the Integration Toolkit auto-generates.	String
jobId	JobId to assign to the job. Optional for POST/insert/publish method.	
priority	Priority of the dropoff segment. Optional for POST/insert/publish method. If omitted, Fleet Manager assigns default priority.	Integer
robot	AMR for this Dropoff job request.	String
goal	Name of the dropoff goal. Required with the POST/insert/publish method.	
assignedJobId	JobId assigned to the job. If JobId was not provided when the job was created, the Fleet Manager automatically creates the assignedJobId.	
status	"Success" or failure message from the Queuing Manager.	

Dropoff Job Using REST

Use the following REST calls to generate and delete existing Dropoff jobs. These calls can also be used to get information for Dropoff jobs queued to the Integration Toolkit.

Table 5-6. Dropoff Job REST Call Resources

Method	Resource	Function
POST	/Dropoff	Create Dropoff job.
GET	/Dropoff/UpdatedSince?sinceTime={time millis}	Get a list of Dropoff entities that have been updated since the given time.
GET	/Dropoff/Stream	Listen for all Dropoff job updates.
GET	/Dropoff/ByKey/{namekey}	Get Dropoff by namekey.
GET	/Dropoff/ByJobId/{JobId}	Get a list of Dropoff entities filtered by JobId.
GET	/Dropoff/ByStatus/{Status}	Get a list of Dropoff entities filtered by status.

Method	Resource	Function
GET	/Dropoff/ByAssignedJobId/{AssignedJobId}	Get a list of Dropoff entities filtered by Assigned JobId.
GET	/Dropoff /ByRobot/{AMR}	Get a list of Dropoff entities filtered by AMR.

Dropoff Job JSON Schema

```
{
        "namekey": "string",
        "robot": "string",
        "goal": "string",
        "priority": integer,
        "jobId": "string",
        "status": "string",
        "assignedJobId": "string"
}
```

Dropoff Job JSON Example

To create a dropoff job using a POST request https://[IP]:8443/Dropoff :

```
{
        "namekey": "Dropoff1",
        "robot": "Robot6",
        "goal": "Goal1",
        "priority": 10,
        "jobId": "TestJob1"
}
```

cURL Command String Example

To create a dropoff job using the POST method:

```
curl [options] -X POST "https://[IP]:8443/Dropoff" -H  "accept:
application/json; charset=utf-8" -H  "Content-Type: application/json;
charset=utf-8" -d "
{\"namekey\":\"Dropoff1\",\"robot\":\"Robot6\",\"goal\":\"Goal1\",\"priority\":
10,\"jobId\":\"TestJob1\"}"
```

Dropoff Job Using SQL

```
INSERT INTO dropoff_view (namekey, robot, goal, priority, job_id) VALUES
('Dropoff1', 'Robot1', 'Goal1', 10, 'TestJob1');
```

Dropoff Job Using RabbitMQ

inbound.Dropoff

outbound.Dropoff

Creating Job Request Job Types (Multi-segment)

Job Requests are jobs with a variable number of segments (like queueMulti from ARCL).

This section provides details about the following Job Request functions. These functions can be used to manage Job Requests for the AMR fleet.

- Queue an AMR for multiple Pickups and Dropoffs at multiple goals with a new Job Request.
- Monitor existing Job Requests.
- Delete existing Job Requests.

The following table describes the Job Request item schema.

Table 5-7. Job Request Item Schema

Item	Details	Data Type
namekey	Unique identifier of a Job request entity. Optional for POST/insert/publish method. If omitted, the Integration Toolkit auto-generates. Required for SQL INSERT statements.	String
jobId	JobId to assign to the job. Optional for POST/insert/publish method.	
defaultPriority	Default priority selection. Required for POST/insert/publish method. When set to true, this will override any priorities set in this job request.	Bool
goal (enumerated item) See note below	The goal name for the job segment. Refer to Job Request JSON Schema below. Required for POST/insert/publish method.	String
priority	Priority of the pickup segment. Optional for POST method. If omitted, Fleet Manager assigns default priority.	Integer
segmentType (enumerated item) See note below	Must be replaced with job segment type with either "pickupGoal" or "dropoffGoal". Refer to Job Request JSON example below. Required for POST/insert/publish method.	String
assignedJobId	JobId assigned to the job. If JobId was not provided when the job was created, the Fleet Manager automatically creates the assignedJobId.	
status	"Success" or failure message from the Queuing Manager.	

NOTE: The goal and segmentType items in every job request's details are presented in one type-value pair instead of two separate entries. If the segment type is pickup, segmentType and goal are presented as "pickupGoal":"*goal name*". If the segment type is dropoff, segmentType and goal are presented as "dropoffGoal":"*goal name*". Refer to the JSON example below for correct syntax.

Job Request Using REST

Use the following REST calls to generate and delete existing Job Requests. These calls can also be used to get information for Job Requests queued to the Integration Toolkit.

> **NOTE:** To obtain job details such as goal, priority, and segment type, use the JobRequestDetails resources listed in Table 5-9.

Table 5-8. Job Request REST Call Resources

Method	Resource	Function
POST	/JobRequest	Create Job Request.
GET	/JobRequest/UpdatedSince?sinceTime={time millis}	Get a list of Job Request entities that have been updated since the given time.
GET	/JobRequest/Stream	Listen for all Job Request updates.
GET	/JobRequest/ByKey/{namekey}	Get Job Request by namekey.
GET	/JobRequest/ByJobId/{JobId}	Get a list of Job Request entities filtered by JobId.
GET	/JobRequest/ByStatus/{Status}	Get a list of Job Request entities filtered by Status.
GET	/JobRequest/ByAssignedJobId/{AssignedJobId}	Get a list of Job Request entities filtered by AssignedJobId.
DELETE	/JobRequest/{namekey}	Delete Job Request by namekey.

Table 5-9. Job Request Detail REST Call Resources

Method	Resource	Function
GET	/JobRequestDetail/ByKey/{namekey}	Get JobRequestDetail by namekey
GET	/JobRequestDetail/UpdatedSince?sinceTime={time millis}	Get a list of JobRequestDetail entities that have been updated since the given time
GET	/JobRequestDetail/ByJobRequest/{JobRequest namekey}	Get a list of JobRequestDetail filtered by JobRequest

Job Request JSON Schema

```
{
	"namekey": "string",
	"audit": {
		"namekey": "string",
		"crt": {
			"millis": "long"
		},
		"upd": {
			"millis": "long"
		},
		"ver": integer
	},
	"jobId": "string",
	"defaultPriority": true,
	"details": [
		{
		"segmentType": "string",
		"priority": integer,
		}
	],
	"status": "string",
	"assignedJobId": "string"
}
```

Job Request JSON Example

To create a job request using a POST request https://[IP]:8443/JobRequest :

```
{
	"namekey": "JobRequest1",
	"jobId": "TestJob1",
	"defaultPriority": false,
	"details": [
		{
		"pickupGoal": "Goal1",
		"priority": 10
		},
		{
		"dropoffGoal": "Goal2",
		"priority": 20
		}
	]
}
```

cURL Command String Example

To create a job request using a POST request:

```
curl [options] -X POST "https://[IP]:8443/JobRequest" -H  "accept:
application/json; charset=utf-8" -H  "Content-Type: application/json;
charset=utf-8" -d "
{\"namekey\":\"JobRequest1\",\"jobId\":\"TestJob1\",\"defaultPriority\":false,\
"details\":[{\"pickupGoal\":\"Goal1\",\"priority\":10},
{\"dropoffGoal\":\"Goal2\",\"priority\":20}]}"
```

Job Request Using SQL

Job requests involve inserting into both the job_request_view and job_request_detail_view. To accomplish this you will need add the segments associated with the job in the job_request_detail_view after creating the job itself in the job_request_view. These database insertions need to be created in the same transaction.

To create a job request using the POST method:

```
BEGIN; WITH new_jobRequest AS (INSERT INTO job_request_view(job_id, default_
priority, namekey) VALUES ('JobID1', true,uuid_generate_v1()) RETURNING namekey)


INSERT INTO job_request_detail_view (job_request,idx, goal, priority, segment_
type, namekey) VALUES ((SELECT namekey FROM new_
jobRequest),'1','Goal1',10,'Pickup',(SELECT namekey FROM new_jobRequest) || '
1'), ((SELECT namekey FROM new_jobRequest),'1','Goal2',20,'Dropoff',(SELECT
namekey FROM new_jobRequest) || '2');

COMMIT;
```

Job Request Using RabbitMQ

inbound.JobRequest

outbound.JobRequest

5.3 Monitoring of Jobs

The monitoring of jobs can be accomplished with any communication channel. Live updates can be obtained from the REST stream endpoints or from the RabbitMQ channel. On demand data can be obtained with non-stream REST calls and from the database.

When monitoring jobs, keep in mind the two-step nature of job creation (refer to Job Creation Steps on page 31). If job creation fails (for instance when a specified goal doesn't exist on the map), no job will be created and the failure will be reported on the status field of the job creation record without making any job or job_segment record. Given this functionality, a programmer may find it beneficial to check the status of a created job prior to working with the job table, REST path, or RabbitMQ queue to monitor its progress.

Job Monitoring Schema Entities

Job monitoring schema consists of the following entities:

- Job - get the status of queued jobs.
- Job Segment - get the status of queued job segments.
- Job History - get the history of jobs.
- Job Segment History - get the history of job segments.

Job Monitoring Details

This section provides details for monitoring all existing jobs.

The following table describes the Job Monitoring item schema.

Table 5-10. Job Monitoring Item Schema

Item	Details	Data Type
namekey	Unique identifier of a job entity.	String
jobId	Assigned jobId by the Fleet Manager.	
queuedTimestamp	Time when the job was queued.	Integer
jobType	Job type of Pickup (P), Dropoff (D), PickupDropoff (PD), or Multi-goal (M).	String
lastAssignedRobot	Name of the AMR last assigned to this job.	
cancelReason	Reason why the job was cancelled. Provided by Fleet Manager only when supplied by the client during job cancellation.	
status	Job status of Pending, InProgress, Completed, Cancelled, Cancelling, or Modifying.	
completedTimestamp	Time when the job was completed.	Integer
linkedJob	*Reserved for future use (empty).*	String
failCount	Number of times the job has failed. Only present if the job has failed.	Integer

Job Monitoring Using REST

Use the following REST calls to monitor an existing job.

Table 5-11. Job Monitoring REST Call Resources

Method	Resource	Function
GET	/Job/ByKey/{namekey}	Get job by namekey.
GET	/Job/UpdatedSince?sinceTime={time millis}	Get a list of job entities that have been updated since the given time.
GET	/Job/Stream	Listen for all Job updates.
GET	/Job/History?sinceTime={time millis}	Get a list of Job history entities since the given time.
GET	/Job/History?sinceTime={time millis}&namekey={namekey}	Get a list of Job history entities for a specific namekey since the given time.
GET	/Job/ByJobId/{JobId}	Get a list of Job entities filtered by JobId.
GET	/Job/ByLastAssignedRobot/{LastAssignedRobot}	Get a list of Job entities filtered by lastAssignedRobot.
GET	/Job/ByStatus/{Status}	Get a list of Job entities filtered by

Method	Resource	Function
		status.

Job Monitoring JSON Schema

```
{
       "namekey": "string",
       "upd": {
       "millis": "long"
  },
       "jobId": "string",
       "jobType": "P, D, PD, M",
       "queuedTimestamp": {
       "millis": "long"
  },
       "completedTimestamp": {
       "millis": "long"
  },
       "status": "Pending, InProgress, Completed, Cancelled, Cancelling,
        Modifying",
       "linkedJob": "string",
       "failCount": integer,
       "lastAssignedRobot": "string",
       "cancelReason": "string"
}
```

Job Monitoring JSON Example

Response example after GET request https://[IP]:8443/Job/ByStatus/Completed :

```
{
       "namekey": "JOB700-5c9eadd6",
       "upd": {
         "millis": "1553903125825"
       },
       "jobId": "JOB700",
       "jobType": "M",
       "queuedTimestamp": {
         "millis": "1553903062000"
       },
       "completedTimestamp": {
         "millis": "1553903105000"
       },
       "status": "Completed",
       "lastAssignedRobot": "Robot154"
  }
```

cURL Command String Example

To get information about completed jobs using the GET method:

```
curl [options] -X GET "https://[IP]:8443/Job/ByStatus/Completed" -H  "accept:
application/json; charset=utf-8"
```

Job Monitoring Using SQL

To get information about completed jobs:

```
SELECT * FROM job_view WHERE status = 'Completed';
```

Job Monitoring Using RabbitMQ

outbound.Pickup

outbound.Dropoff

outbound.PickupDropoff

outbound.JobRequest

outbound.Job

> **Additional Information**: outbound.Pickup/Dropoff/PickupDropoff/JobRequest provide updates on these request entities. outbound.Job provides updates on jobs sent from the Integration Toolkit to the Fleet Manager.

Job Segment Monitoring Details

The section below provides details for monitoring Job Segments.

The following table describes the Job Segment Monitoring item schema.

Table 5-12. Job Segment Item Schema

Item	Details	Data Type
namekey	Unique identifier of a Job Segment entity.	String
segmentId	Job segmentId.	
segmentType	Job Segment type of Pickup or Dropoff.	
seq	Job Segment sequence number.	Integer
status	Job Segment status of Pending, InProgress, Interrupted, Completed, Cancelled, Cancelling, Failed, Modifying, or Modified.	String
subStatus	Job Segment sub-status of Unallocated, Allocated, BeforePickup, BeforeDropOff, BeforeEvery, Before, Driving, After, AfterEvery, AfterPickup, AfterDropOff, Buffering, Buffered, None, ContainsCancelReason, ContainsLinkedReason, AssignedRobotOffline, NoMatchingRobotForLinkedJob, NoMatchingRobotForOtherSegment, or NoMatchingRobot.	
job	NameKey of the associated job.	
robot	Job Segment assigned AMR name.	

Item	Details	Data Type
linkedJobSegment	Job segmentId of the previous job segmentId (if applicable). The value of this item is null for the first segment of the job.	
priority	Job Segment priority.	Integer
goalName	Job segment goal name.	String
cancelReason	Reason why the Job Segment was cancelled. Provided by Fleet Manager only when supplied by the client during job cancellation.	

Job Segment Monitoring Using REST

Use the following REST calls to monitor an existing job segment.

Table 5-13. Job Segement Monitoring REST Call Resources

Method	Resource	Function
GET	/JobSegment/ByKey/{namekey}	Get Job Segment by namekey.
GET	/JobSegment/UpdatedSince?sinceTime={time millis}	Get a list of Job Segment entities that have been updated since the given time.
GET	/JobSegment/Stream	Listen for all Job Segment updates.
GET	/JobSegment/History?sinceTime={time millis}	Get a list of Job Segment history entities that have been updated since the given time.
GET	/JobSegment/History?sinceTime={time millis}&namekey={namekey}	Get a list of Job Segment history entities for a specific namekey that have been updated since the given time.
GET	/JobSegment/ByStatus/{Status}	Get a list of Job Segment entities filtered by status.
GET	/JobSegment/ByJob/{Job}	Get a list of Job Segment entities filtered by job.
GET	/JobSegment/ByRobot/{AMR}	Get a list of Job Segment entities filtered by AMR.
GET	/JobSegment/ByGoalName/{GoalName}	Get a list of Job Segment entities filtered by goalName.

Job Segment Monitoring JSON Schema

```
{
       "namekey": "string",
       "upd": {
              "millis": "long"
              },
       "seq": integer,
       "segmentId": "string",
       "segmentType": "Pickup, DropOff",
       "status": "Pending, InProgress, Interrupted, Completed, Cancelled,
        Cancelling, Failed, Modifying, Modified, InterruptedByModify",
       "subStatus": "Unallocated, Allocated, BeforePickup, BeforeDropOff,
        BeforeEvery, Before, Driving, After, AfterEvery, AfterPickup, AfterDropOff,
        Buffering, Buffered, None, ContainsCancelReason, ContainsLinkedReason,
        AssignedRobotOffline, NoMatchingRobotForLinkedJob,
        NoMatchingRobotForOtherSegment, NoMatchingRobot",
       "job": "string",
       "robot": "string",
       "linkedJobSegment": "string",
       "goalName": "string",
       "priority": integer,
       "completedTimestamp": {
              "millis": "long"
              },
       "cancelReason": "string"
}
```

Job Segment Monitoring JSON Example

Response example after GET request https://[IP]:8443/JobSegment/ByStatus/Completed :

```
{
       "namekey": "JobMultiBasic-REST-5c9eacdc-PICKUP688",
        "upd": {
          "millis": "1553902841229"
        },
        "segmentId": "PICKUP688",
        "segmentType": "Pickup",
        "status": "Completed",
        "subStatus": "None",
        "robot": "Robot154",
        "job": "JobMultiBasic-REST-5c9eacdc",
        "goalName": "GoalN1",
        "priority": 10,
        "completedTimestamp": {
          "millis": "1553902821000"
        },
        "seq": 1
  }
```

cURL Command String Example

To get information about completed Job Segments using the GET method:

```
curl [options] -X GET "https://[IP]:8443/JobSegment/ByStatus/Completed" -H
"accept: application/json; charset=utf-8"
```

Job Segment Monitoring Using SQL

To get information about completed Job Segments:

```
SELECT * FROM job_segment_view WHERE status = 'Completed';
```

Job Segment Monitoring Using RabbitMQ

outbound.JobSegment

5.4 Job Segment Modification

Jobs that have not yet completed can be modified by having a segment's goal and/or its priority changed. These modifications can occur at any point prior to a segment completing.

A priority change will not be recorded if the segment has already begun as the priority has already been used to schedule the segment.

> **NOTE:** If a Job segment modification request is issued after the job has completed, it will be recorded but it will not affect the Job.

Job Segment Modification Details

This section provides details about how to modify existing Job Segments. It also describes how to monitor and delete a modified Job Segment.

The following table describes the Job Segment Modification item schema.

Table 5-14. Jog Segment Modification Item Schema

Item	Description	Data Type
namekey	Unique identifier of a JobSegmentModify request entity. Optional for POST/insert/publish method. If omitted, the Integration Toolkit auto-generates.	String
segment namekey	Unique identifier of a Job Segment being modified. This or segmentId required for POST/insert/publish method.	
segmentId	Modify pending or in progress jobs based on the segment namekey. This or segment namekey required for POST/insert/publish method.	
goal	Name of the goal for the modified Job Segment. Optional for POST/insert/publish method.	
priority	Priority of the modified Job Segment. Optional for POST/insert/publish method.	Integer
status	"Success" or failure message from the Queuing Man-	

Item	Description	Data Type
	ager.	

Job Segment Modification Using REST

Use the following REST calls to modify and delete existing Job Segments. These calls can also be used to get information for modified Job Segments queued to the Integration Toolkit.

Table 5-15. Job Segment REST Call Resources

Method	Resource	Function
GET	/JobSegmentModify/ByKey/{namekey}	Get modified Job Segments by namekey.
DELETE	/JobSegmentModify/{namekey}	Delete modified Job Segments by namekey.
GET	/JobSegmentModify/UpdatedSince?sinceTime={time millis}	Get a list of modified Job Segment entities updated since the given time.
POST	/JobSegmentModify	Create a modified Job Segment.
GET	/JobSegmentModify/BySegmentId/{SegmentID}	Get a list of modified Job Segment entities filtered by SebmentId.

Job Segment Modification JSON Schema

```
{
        "namekey": "string",
        "audit": {
                "namekey": "string",
                "crt": {
                        "millis": "long"
                },
                "upd": {
                        "millis": "long"
                },
                "ver": integer
        },
        "segmentId": "string",
        "segmentNamekey": "string",
        "priority": integer,
        "goal": "string",
        "status": "string"
}
```

Job Segment Modification JSON Example

To modify a Job Segment with a POST request https://[IP]:8443/JobSegmentModify :

```
{
        "segmentId": "DROPOFF6",
        "priority": 10,
        "goal": "L6_2"
}
```

cURL Command String Example

To modify a Job Segment with a POST request:

```
curl [options] -X POST "https://[IP]:8443/JobSegmentModify" -H  "accept:
application/json; charset=utf-8" -H  "Content-Type: application/json;
charset=utf-8" -d "{\"segmentId\":\"DROPOFF6\",\"priority\":10,\"goal\":\"L6_
2\"}"
```

Job Segment Modification Using SQL

```
INSERT INTO job_segment_modify_view (goal, priority, segment_id) VALUES ('L6_2',
10, 'DROPOFF6');
```

Job Segment Modification Using RabbitMQ

inbound.JobSegmentModify

outbound.JobSegmentModify

5.5 Job and Job Segment Cancellation

Job and Job segment cancellations are achieved by providing a method and associated value to identify the Job to be cancelled.

Job and Job Segment Cancellation Details

This section provides details for cancelling Jobs and Job segments. Existing Job cancellations can be monitored and deleted.

The following table describes the Job and Job segment cancellation item schema.

Table 5-16. Job and Job Segment Cancellation Schema

Item	Details	Data Type
namekey	Unique identifier for a Job cancel request entity. Optional for POST/insert/publish method. If omitted, the Integration Toolkit auto-generates.	String
cancelType (enumerated item)	Must be replaced with cancellation method from below. Refer to Job Request JSON Example below. Required for POST/insert/publish method.	
cancelValue	Must be replaced with a value for the	

Item	Details	Data Type
(enumerated item)	chosen cancelType. Provides the value for the "cancelType" method chosen.	
cancelReason	Reason Job or Job Segment was cancelled. Optional for POST/insert/publish method.	
status	"Success" or failure message from the Queuing Manager.	

A cancellation method must be specified for the cancelType item. The following cancelType methods are supported.

Table 5-17. Job and Job Segment Cancellation Method Descriptions

cancelType Method	Description
jobId	Cancel pending or in progress job based on the JobId.
jobNamekey	Cancel pending or in progress job based on the Job's namekey.
segmentId	Cancel pending or in progress job based on the segmentId.
segmentNamekey	Cancel pending or in progress job based on the Segment namekey.
jobStatus	Cancel all jobs with the given status.
robot	Cancel all jobs assigned to a given AMR.
removeSegmentId	Cancel a job segment using the SegmentId.
removeSegmentNamekey	Cancel a job segment using the segment namekey.

Job and Job Segment Cancellation Functions Using REST

Use the following REST calls to cancel Jobs and Job segments, and monitor existing Job cancellations.

Table 5-18. Job and Job Segment Cancellation REST Call Resources

Method	Resource	Function
GET	/JobCancel/ByKey/{namekey}	Get Job cancel request by the job cancel request namekey.
DELETE	/JobCancel/{namekey}	Delete cancel Job or Job Segmentcancel request, by namekey.
GET	/JobCancel/UpdatedSince?sinceTime={time millis}	Get a list of cancelled Job entities that have been updated since the given

Method	Resource	Function
		time.
POST	/JobCancel	Create a cancel Job.

Job and Job Segment Cancellation JSON Schema

```
{
        "namekey": "string",
        "cancelType": "string",
        "cancelValue": "string",
        "cancelReason": "string",
        "status": "string"
}
```

Job and Job Segment Cancellation JSON Example

To cancel a job with a POST request https://[IP]:8443/JobCancel :

```
{
        "namekey": "cancelJob1",
        "jobId": "TestJob1",
        "cancelReason": "Job needed to be cancelled"
}
```

cURL Command String Example

To cancel a job with a POST request:

```
curl [options] -X POST "https://[IP]:8443/JobCancel" -H  "accept:
application/json; charset=utf-8" -H  "Content-Type: application/json;
charset=utf-8" -d "
{\"namekey\":\"cancelJob1\",\"jobId\":\"TestJob1\",\"cancelReason\":\"Job
needed to be cancelled\"}"
```

Job Cancellation Using SQL

To cancel a job:

```
INSERT INTO job_cancel_view (namekey, cancel_type, cancel_value, cancel_
reason) VALUES ('cancelJob1', 'JobId', 'TestJob1', 'Job needed to be
cancelled');
```

Job Cancellation Using RabbitMQ

inbound.JobCancel

outbound.JobCancel

5.1 WaitTaskCancel

The Integration Toolkit version with FLOW 2.0 will include one additional feature. It adds the ability, through REST only, to cancel a wait state and also to check on the status of the wait state.

Version Information: This feature is available as of FLOW 2.0 Integration Toolkit version 1.1.0.

Table 5-19. WaitTask Cancellation REST Call Resources

Method	Resource	Function
POST	/WaitTaskCancel	Cancels a wait state.
GET	/WaitTaskState/{robot}	Obtains waiting status on a robot.

cURL examples

Following are POST and GET cURL examples for robot Robot1:

POST

```
curl [options] -X POST "https://[IP]:8443/WaitTaskCancel" -H
"accept: application/json; charset=utf-8"
-H "Content-Type: application/json; charset=utf-8" -d "{\"robot\":\"Robot1\"}"
```

GET

```
curl [options] -X GET "https://[IP]:8443/WaitTaskState/Robot1" -H "accept:
application/json; charset=utf-8"
```

Responses

Both REST calls return a code reflecting the waiting state of the robot per the following table:

Code	Description	Note
Positive value	"waiting for <s> seconds"	
0	"wait done"	Unlikely return value as state is transient
-1	"wait forever"	
-2	"wait task interrupted"	Unlikely return value as state is transient
-3	"wait stopped by user"	Unlikely return value as state is transient
-4	"wait interrupted by this client"	
-5	"not waiting now"	

Chapter 6: AMR Information and Faults

AMR information and faults can be monitored with the Integration Toolkit. Details are provided in the sections below.

6.1 AMR Information

This section provides details about how AMR information can be monitored using the Integration Toolkit.

AMR Information Monitoring Details

This section provides details for monitoring individual AMR status.

> **Additional Information**: Detailed information such as location and battery state can be obtained by accessing associated DataStore values. Refer to DataStore on page 23 for more information.
>
> **Version Information:** As of FLOW 2.0 (Integration Toolkit version 1.1.0) the list of AMRs provided by the Integration Toolkit will reflect only those currently attached to the Fleet Manager, with robots removed and/or added within 10 seconds of their addition or removal from the fleet.

The following table describes the AMR Information Monitoring item schema.

Table 6-1. AMR Information Monitoring Item Schema

Item	Description	Data Type
namekey	Unique identifier of an AMR (same as the name of the AMR provided in the fleet).	String
status	Available, InProgress, Unavailable, Unavailable_ Busy, Unavailable_NeedsAssistance, AvailableForJobs	
subStatus	Unallocated, Allocated, Available, Fault, BeforePickup, BeforeDropoff, BeforeEvery, Before, Driving, After, AfterEvery, AfterPickup, AfterDropoff, ModeIsLocked, Parking, Parked, Docking, Docked, DockParking, DockParked, ForcedDocking, ForcedDocked, Interrupted, InterruptedButNotYetIdle, OutgoingArclConnectionLost, EstopPressed, EstopRelieved, MotorsDisabled, Lost, AvailableForJobs, Buffering, Buffered	

AMR Information Monitoring Using REST

Use the following REST calls to monitor AMR Information.

Table 6-2. AMR Information Monitoring REST Call Resources

Method	Resource	Function
GET	/Robot/ByKey/{namekey}	Get AMR status by AMR name (namekey).
GET	/Robot/History?sinceTime={time millis}	Get a list of AMR history entities that have been updated since the given time.
GET	/Robot/History?sinceTime={time millis}&namekey={namekey}	Get a list of AMR history entities for a specific namekey that have been updated since the given time.
GET	/Robot/ByStatus/{Status}	Get a list of AMR entities filtered by status.
GET	/Robot/BySubStatus/{SubStatus}	Get a list of AMR entities filtered by SubStatus.
GET	/Robot/UpdatedSince?sinceTime={time millis}	Get a list of AMR entities that have been updated since the given time.

AMR Data Monitoring JSON Schema

```
{
      "namekey": "string",
      "upd": {
      "millis": "long"
      },
      "status": "Available, InProgress, Unavailable, Unavailable_Busy,
       Unavailable_NeedsAssistance, AvailableForJobs",
      "subStatus": "Unallocated, Allocated, Available, Fault, BeforePickup,
       BeforeDropoff, BeforeEvery, Before, Driving, After, AfterEvery,
       AfterPickup, AfterDropoff, ModeIsLocked, Parking, Parked, Docking,
       Docked, DockParking, DockParked, ForcedDocking, ForcedDocked, Interrupted,
       InterruptedButNotYetIdle, OutgoingArclConnectionLost, EstopPressed,
       EstopRelieved, MotorsDisabled, Lost, AvailableForJobs, Buffering,
       Buffered"
}
```

AMR Data Monitoring JSON Example

Response example after GET request https://[IP]:8443/Robot/ByStatus/Available :

```
{
        "namekey": "Robot154",
        "upd": {
        "millis": "1553904366723"
        },
       "status": "Unavailable_Busy",
```

```
        "subStatus": "InterruptedButNotYetIdle"
    }
```

cURL Command String Example

To get information about available AMRs:

```
curl [options] -X GET "https://[IP]:8443/Robot/ByStatus/Available" -H
"accept: application/json; charset=utf-8"
```

AMR Data Monitoring Using SQL

```
SELECT namekey, status, sub_status FROM robot_view WHERE status LIKE
'Available%';
```

AMR Information Monitoring Using RabbitMQ

outbound.Robot

6.2 AMR Faults

AMR faults can be monitored with the Integration Toolkit.

All AMR faults are created independent of the Integration Toolkit.

AMR Fault Monitoring

This section provides details for monitoring all AMR faults. These functions can be used to indicate when an AMR has a problem and can provide information about AMR fault recovery.

AMR Fault Monitoring Details

The following table describes the AMR Fault item schema.

Table 6-3. AMR Fault Item Schema

Item	Description	Data Type
namekey	Unique identifier of a fault entity.	String
upd	Timestamp when this record was last updated.	Long
robot	Name of AMR with fault state set.	String

Item	Description	Data Type
active	Current fault state.	Boolean
blockDriving	Block Driving fault.	
driving	Driving fault.	
critical	Critical fault.	
clearedOnGo	Fault clear on go.	
clearedOnAck	Fault clear on acknowledge.	
application	Application fault.	
name	Fault name.	String
type	Fault type.	
shortDescription	Short description of fault.	
longDescription	Long description of fault.	

AMR Fault Monitoring Using REST

Use the following REST calls to monitor AMR faults.

Table 6-4. AMR Fault Monitoring REST Call Resources

Method	Resource	Function
GET	/RobotFault/ByKey/{namekey}	Get AMR fault by namekey.
GET	/RobotFault/UpdatedSince?sinceTime={time millis}	Get a list of AMR fault entities that have been updated since the given time.
GET	/RobotFault/History?sinceTime={time millis}	Get a list of AMR fault history entities that have been updated since the given time.
GET	/RobotFault/History?sinceTime={time millis}&namekey={namekey}	Get a list of AMR fault history entities for a specific namekey that have been updated since the given time.
GET	/RobotFault/ByRobot/{AMR}	Get a list of AMR fault entities filtered by AMR.
GET	/RobotFault/ByType/{Type}	Get a list of AMR fault entities filtered by type.
GET	/RobotFault/ByName/{Name}	Get a list of AMR fault entities filtered by name.
GET	/RobotFault/ByActive/{Value}	Get a list of AMR fault entities filtered by active state.

AMR Fault Monitoring JSON Schema

```
{
        "namekey": "string",
        "upd": {
                "millis": "long"
                },
        "robot": "string",
        "active": boolean,
        "blockDriving": boolean,
        "driving": boolean,
        "critical": boolean,
        "clearedOnGo": boolean,
        "clearedOnAck": boolean,
        "application": boolean,
        "name": "string",
        "type": "string",
        "shortDescription": "string",
        "longDescription": "string"
}
```

AMR Fault Monitoring JSON Example

Response example after GET request https://[IP]:8443/RobotActive/false :

```
{
        "namekey": "Robot154:Fault1",
        "upd": {
          "millis": "1553904236251"
        },
        "robot": "Robot154",
        "active": false,
        "blockDriving": false,
        "driving": false,
        "critical": false,
        "clearedOnGo": false,
        "clearedOnAck": false,
        "application": true,
        "name": "Fault1",
        "type": "Fault_Application",
        "shortDescription": "Fault1 Desc",
        "longDescription": "Fault1 Long Desc"
  }
```

cURL Command String Example

To get information about AMR with active faults using the GET method:

```
curl [options] -X GET "https://[IP]:8443/RobotFault/ByActive/true" -H "accept:
application/json; charset=utf-8"
```

AMR Fault Monitoring Using SQL

```
SELECT * FROM robot_fault_view WHERE active = true;
```

AMR Fault Monitoring Using RabbitMQ

outbound.RobotFault

Appendices

A.1 SQL Database Schema

The following table view names are available in the IntegrationDB.

data_store_value_view

dropoff_view

goal_view

job_cancel_view

job_history_view

job_request_detail_view

job_request_view

job_segment_history_view

job_segment_modify_view

job_segment_view

job_view

pickup_dropoff_view

pickup_view

robot_fault_history_view

robot_fault_view

robot_history_view

robot_view

subscription_config_view

A.2 REST Calls

This section provides details for all supported REST calls.

/DataStoreItem/ByCategory/{Category}

Get a list of DataStoreItem entities filtered by Category.

Request Type: GET

/DataStoreItem/ByDisplayName/{DisplayName}

Get a list of DataStoreItem entities filtered by DisplayName.

Request Type: GET

/DataStoreItem/ByGroupName/{GroupName}

Get a list of DataStoreItem entities filtered by GroupName.

Request Type: GET

/DataStoreItem/ByItemName/{ItemName}

Get a list of DataStoreItem entities filtered by ItemName.

Request Type: GET

/DataStoreItem/ByKey/{namekey}

Get DataStoreItem by namekey.

Request Type: GET

/DataStoreItem/BySource/{Source}

Get a list of DataStoreItem entities filtered by Source (Source is AMR name).

Request Type: GET

/DataStoreItem/ByType/{Type}

Get a list of DataStoreItem entities filtered by Type.

Request Type: GET

/DataStoreItem/UpdatedSince?sinceTime={time millis}

Get a list of DataStoreItem entities that have been updated since the given time.

Request Type: GET

/DataStoreValue/ByKey/{namekey}

Get DataStoreValue by namekey.

Request Type: GET

/DataStoreValue/UpdatedSince?sinceTime={time millis}

Get a list of DataStoreValue entities that have been updated since the given time.

Request Type: GET

/DataStoreValueLatest/{DataStore item name}

Return (without subscription) a one-time value for the DataStore item named.

Request Type: GET

/DataStoreValueLatest/{DataStore item name}:{AMR name}

Return (without subscription) a one-time value for the DataStore item named on the AMR named.

Request Type: GET

*/DataStoreValueLatest/{DataStore item name}:**

Return (without subscription) a one-time value for the DataStore item named on all robots.

Request Type: GET

/Dropoff

Create Dropoff.

Request Type: POST

/Dropoff/{namekey}

Delete Dropoff by namekey.

Request Type: DELETE

/Dropoff/ByAssignedJobId/{AssignedJobId}

Get a list of Dropoff entities filtered by AssignedJobId.

Request Type: GET

/Dropoff/ByJobId/{JobId}

Get a list of Dropoff entities filtered by JobId.

Request Type: GET

/Dropoff/ByKey/{namekey}

Get Dropoff by namekey.

Request Type: GET

/Dropoff/ByRobot/{AMR}

Get a list of Dropoff entities filtered by AMR.

Request Type: GET

/Dropoff/ByStatus/{Status}

Get a list of Dropoff entities filtered by Status.

Request Type: GET

/Dropoff/UpdatedSince?sinceTime={time millis}

Get a list of Dropoff entities that have been updated since the given time.

Request Type: GET

/Goal/ByKey/{namekey}

Get Goal by namekey.

Request Type: GET

/Goal/UpdatedSince?sinceTime={time millis}

Get a list of Goal entities that have been updated since the given time.

Request Type: GET

/Job/ByJobId/{JobId}

Get a list of Job entities filtered by JobId.

Request Type: GET

/Job/ByKey/{namekey}

Get Job by namekey.

Request Type: GET

/Job/ByLastAssignedRobot/{LastAssignedRobot}

Get a list of Job entities filtered by LastAssignedRobot.

Request Type: GET

/Job/ByStatus/{Status}

Get a list of Job entities filtered by Status.

Request Type: GET

/Job/History?sinceTime={time millis}

Get a list of Job history entities since the given time.

Request Type: GET

/Job/History?sinceTime={time millis}&namekey={namekey}

Get a list of Job history entities for a specific namekey since the given time.

Request Type: GET

/Job/UpdatedSince?sinceTime={time millis}

Get a list of job entities that have been updated since the given time.

Request Type: GET

/JobCancel

Create JobCancel.

Request Type: POST

/JobCancel/{namekey}

Delete JobCancel by namekey.

Request Type: DELETE

/JobCancel/ByKey/{namekey}

Get JobCancel by namekey.

Request Type: GET

/JobCancel/UpdatedSince?sinceTime={time millis}

Get a list of JobCancel entities that have been updated since the given time.

Request Type: GET

/JobRequest

Create JobRequest.

Request Type: POST

/JobRequest/{namekey}

Delete JobRequest by namekey.

Request Type: DELETE

/JobRequest/ByAssignedJobId/{AssignedJobId}

Get a list of JobRequest entities filtered by AssignedJobId.

Request Type: GET

/JobRequest/ByJobId/{JobId}

Get a list of JobRequest entities filtered by JobId.

Request Type: GET

/JobRequest/ByKey/{namekey}

Get JobRequest by namekey.

Request Type: GET

/JobRequest/ByStatus/{Status}

Get a list of JobRequest entities filtered by Status.

Request Type: GET

/JobRequestDetail/ByKey/{namekey}

Get JobRequestDetail by namekey.

Request Type: GET

/JobRequestDetail/UpdatedSince?sinceTime={time millis}

Get a list of JobRequestDetail entities that have been updated since the given time.

Request Type: GET

/JobRequest/UpdatedSince?sinceTime={time millis}

Get a list of JobRequest entities that have been updated since the given time.

Request Type: GET

/JobSegment/ByGoalName/{GoalName}

Get a list of JobSegment entities filtered by GoalName.

Request Type: GET

/JobSegment/ByJob/{Job}

Get a list of JobSegment entities filtered by Job namekey.

Request Type: GET

/JobSegment/ByKey/{namekey}

Get JobSegment by namekey.

Request Type: GET

/JobSegment/ByRobot/{AMR}

Get a list of JobSegment entities filtered by AMR.

Request Type: GET

/JobSegment/ByStatus/{Status}

Get a list of JobSegment entities filtered by Status.

Request Type: GET

/JobSegment/History?sinceTime={time millis}

Get a list of job segment history entities that have been updated since the given time.

Request Type: GET

/JobSegment/History?sinceTime={time millis}&namekey={namekey}

Get a list of job segment history entities for a specific namekey that have been updated since the given time.

Request Type: GET

/JobSegment/UpdatedSince?sinceTime={time millis}

Get a list of job segment entities that have been updated since the given time.

Request Type: GET

/JobSegmentModify

Create JobSegmentModify.

Request Type: POST

/JobSegmentModify/{namekey}

Delete JobSegmentModify by namekey.

Request Type: DELETE

/JobSegmentModify/ByKey/{namekey}

Get JobSegmentModify by namekey

Request Type: GET

/JobSegmentModify/BySegmentId/{SegmentId}

Get a list of JobSegmentModify entities filtered by SegmentId.

Request Type: GET

/JobSegmentModify/UpdatedSince?sinceTime={time millis}

Get a list of modified job segment entities updated since the given time.

Request Type: GET

/Pickup

Create Pickup.

Request Type: POST

/Pickup/{namekey}

Delete Pickup by namekey.

Request Type: DELETE

/Pickup/ByAssignedJobId/{AssignedJobId}

Get a list of Pickup entities filtered by AssignedJobId.

Request Type: GET

/Pickup/ByJobId/{JobId}

Get a list of Pickup entities filtered by JobId.

Request Type: GET

/Pickup/ByKey/{namekey}

Get Pickup by namekey.

Request Type: GET

/Pickup/ByStatus/{Status}

Get a list of Pickup entities filtered by Status.

Request Type: GET

/Pickup/UpdatedSince?sinceTime={time millis}

Get a list of Pickup job entities that have been updated since the given time.

Request Type: GET

/PickupDropoff

Create PickupDropoff.

Request Type: POST

/PickupDropoff/{namekey}

Delete PickupDropoff by namekey.

Request Type: DELETE

/PickupDropoff/ByAssignedJobId/{AssignedJobId}

Get a list of PickupDropoff entities filtered by AssignedJobId.

Request Type: GET

/PickupDropoff/ByJobId/{JobId}

Get a list of PickupDropoff entities filtered by JobId.

Request Type: GET

/PickupDropoff/ByKey/{namekey}

Get PickupDropoff by namekey.

Request Type: GET

/PickupDropoff/ByStatus/{Status}

Get a list of PickupDropoff entities filtered by Status.

Request Type: GET

/PickupDropoff/UpdatedSince?sinceTime={time millis}

Get a list of PickupDropoff entities that have been updated since the given time.

Request Type: GET

/Robot/ByKey/{namekey}

Get AMR by namekey.

Request Type: GET

/Robot/ByStatus/{Status}

Get a list of AMR entities filtered by Status.

Request Type: GET

/Robot/BySubStatus/{SubStatus}

Get a list of AMR entities filtered by SubStatus.

Request Type: GET

/Robot/History?sinceTime={time millis}

Get a list of AMR history entities that have been updated since the given time.

Request Type: GET

/Robot/History?sinceTime={time millis}&namekey={namekey}

Get a list of AMR history entities for a specific namekey that have been updated since the given time.

Request Type: GET

/Robot/UpdatedSince?sinceTime={time millis}

Get a list of AMR entities that have been updated since the given time.

Request Type: GET

/RobotFault/ByActive/{Active}

Get a list of RobotFault entities filtered by Active.

Request Type: GET

/RobotFault/ByKey/{namekey}

Get RobotFault by namekey.

Request Type: GET

/RobotFault/ByName/{Name}

Get a list of RobotFault entities filtered by Name.

Request Type: GET

/RobotFault/ByRobot/{AMR}

Get a list of RobotFault entities filtered by AMR.

Request Type: GET

/RobotFault/ByType/{Type}

Get a list of RobotFault entities filtered by Type.

Request Type: GET

/RobotFault/History?sinceTime={time millis}

Get a list of AMR fault history entities that have been updated since the given time.

Request Type: GET

/RobotFault/History?sinceTime={time millis}&namekey={namekey}

Get a list of AMR fault history entities for a specific namekey that have been updated since the given time.

Request Type: GET

/RobotFault/UpdatedSince?sinceTime={time millis}

Get a list of AMR fault entities that have been updated since the given time.

Request Type: GET

/SubscriptionConfig

Update SubscriptionConfig.

Request Type: PUT

/SubscriptionConfig/ByKey/{namekey}

Get SubscriptionConfig by namekey.

Request Type: GET

/SubscriptionConfig/UpdatedSince?sinceTime={time millis}

Get a list of SubscriptionConfig entities that have been updated since the given time.

Request Type: GET

/WaitTaskCancel

Cancel a wait state.

Request Type: POST

/WaitTaskState/{robot}

Check on the status of the wait state.

Request Type: GET

A.3 RabbitMQ Queues

This section provides a list of all available inbound and outbound queues for the RabbitMQ communication channel.

Inbound Queues

inbound.Dropoff

inbound.JobCancel

inbound.JobRequest

inbound.JobSegmentModify

inbound.Pickup

inbound.PickupDropoff

inbound.SubscriptionConfig

Outbound Queues

outbound.DataStoreValue

outbound.datastore.X (X is any subscribed DataStoreValue)

outbound.Dropoff

outbound.Job

outbound.JobCancel

outbound.JobRequest

outbound.JobSegment

outbound.JobSegmentModify

outbound.Pickup

outbound.PickupDropoff

outbound.Robot

outbound.RobotFault

outbound.SubscriptionConfig

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.
No. 438A Alexandra Road # 05-05/08 (Lobby 2),
Alexandra Technopark,
Singapore 119967
Tel: (65) 6835-3011/Fax: (65) 6835-2711

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) 21-5037-2222/Fax: (86) 21-5037-2200

Authorized Distributor:

Cat. No. I637-E-02 0720 (0519)

20964-000 B