OMRON

Fleet Operations Workspace Core Integration Toolkit

用户手册

I637-CN5-02

版权声明

编于美国

目录

修订历史

修订代码	日期	修订内容
01	2019 年 6 月	初版
02	2020 年 7 月	FLOW 2.0 版本更新

第 1 章： 简介

本文档包含使用 Integration Toolkit 集成 Fleet Manager 和最终用户客户端应用所必需的信息。

请阅读本文档，并确保您在尝试将 Integration Toolkit 与 AMR 车队调度一起使用之前，了解 Integration Toolkit 的功能和性能。请在使用 Integration Toolkit 之前，阅读并理解所有相关手册和安全指南。

1.1 目标受众

本文档适用于下述人员。

- 集成欧姆龙 AMR 解决方案与制造执行系统（MES）、企业资源规划（ERP）解决方案或其他类似系统的人员。
- 熟悉欧姆龙车队调度管理软件、AMR 以及 EM2100 设备的人员。
- 熟悉 Advanced Robotics Command Language（ARCL）、RESTful Web Services、SQL 或 RabbitMQ 的人员。

1.2 缩略语与术语

下述缩略语和术语将适用于本文档。

术语	说明
AMR	自主导航机器人
ARCL	Advanced Robotics Command Language
API	应用程序设计接口
客户端应用程序	仓库管理系统、制造执行系统、企业资源规划系统或与 Integration Toolkit 交互的类似应用程序。
EM2100 设备	连接所有欧姆龙自主导航机器人以及运行车队调度管理软件的硬件设备。
实体	Integration Toolkit 中的基本对象，包含发送至或接收自 Fleet Manager 的数据，如作业或 DataStore 值。
Fleet Manager	FLOW Core 软件的基本组成，控制交通、充电、作业分配等，并与客户端应用程序协作，以分配和管理整个机器人车队调度的任务。
FLOW	Fleet Operations Workspace 欧姆龙的软件套件，用于管理所有自主导航机器人的导航、安全和车队调度管理功能。
IntegrationDB	主要数据库，包含所有表格视图。
JSON	JavaScript 对象简谱
作业	机器人执行的基本活动，包含一个或多个拾取或卸货部分。
REST	RESTful Web Services

术语	说明
SQL	结构化查询语言
URI	统一资源标识符

1.3 符号

本文档中使用了编程代码和语法示例。这些文本将使用如下所示的字体表示，以与其他非代码文本区分开来。

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

应用特定占位符

IP 地址

在本文档中，Fleet Manager IP 地址采用 [IP] 表示，如下所示。

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

cURL 字符串选项

用户凭证或其他选项的 cURL 字符串中可能需要使用标记。这些选项在本文档中表示为 [options]，如下所示。

```
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 系统要求

Integration Toolkit 需满足以下系统要求。

- EM2100 设备
- 1.0.0 或更高版本的 Fleet Operations Workspace 软件

1.5 如何获得帮助

请访问公司网站：http://www.fa.omron.com.cn，获取更多信息。

相关手册

此外，还有其他手册，描述了如何进行车队调度编程，重新配置已安装组件，以及添加其他可选设备。这些手册提供了安全、相关产品、高级配置和系统规格的相关信息。

手册标题	说明
移动机器人 LD 安全指南（目录编号：I616）	描述欧姆龙 LD 系列 AMR 的安全信息。
移动机器人 HD 安全指南（目录编号：I647）	描述欧姆龙 HD-1500 AMR 的安全信息。
LD 机器人车体 OEM 用户指南（目录编号：I611）	描述 LD-60 和 LD-90 AMR 的安装、启动、操作和维护。
LD-250 机器人车体用户指南（目录编号：I642）	描述 LD-250 AMR 的安装、启动、操作和维护。
HD-1500 机器人车体用户指南（目录编号：I645）	描述 HD-1500 AMR 的安装、启动、操作和维护。
LD 机器人车体外设指南（目录编号：I613）	涵盖 LD AMR 外设，如触摸屏、呼叫器和 Acuity 定位器（选配件）。
移动机器人 -HD 机器人车体手册（目录编号：I646）	涵盖 HD AMR 外设，如 HAPS。
EM2100 安装指南（目录编号：I634）	描述 EM2100 设备的安装和初始配置。
Fleet Operations Workspace 迁移指南（目录编号：I636）	描述将 AMR 从传统软件迁移至 FLOW Core 软件，以及从 EM1100 迁移至 EM2100 的步骤。
Fleet Operations Workspace Core 用户手册（目录编号：I635）	描述如何使用 EM2100 以及在其上运行的软件，以管理 AMR 车队调度。
Fleet Simulator 用户手册（目录编号：I641）	描述 Fleet Simulator 的操作。
Fleet Operations Workspace/EM2100 迁移指南（目录编号：I636）	描述如何在传统解决方案和 FLOW Core 解决方案间升级或降级系统，其中包括软件升级流程和工具，以及所有必要硬件变更的相关指导。
Advanced Robotics Command Language Enterprise Manager 集成指南（目录编号：I618）	描述与 EM2100 软件一起使用的 Advanced Robotics Command Language（ARCL）版本。ARCL 是一个简单的基于文本的命令和响应服务器，用于集成 Fleet Operations Workspace Core 机器人车体和外部自动化系统。
Advanced Robotics Command Language 参考指南（目录编号：I617）	描述 Advanced Robotics Command Language（ARCL），即一种基于文本的简单语言，您可以用其来控制欧姆龙 AMR。
LD 机器人车体推车运输装置用户指南（目录编号：I612）	描述推车运输装置 AMR 的操作和维护。

第 2 章： 功能和特性

Integration Toolkit 是欧姆龙的接口应用程序，可实现 Fleet Manager 和最终用户客户端应用程序的集成。该集成层可实现使用标准通信方法的 AMR 车队调度的自主控制。

它可实现所有 AMR 作业类型（如拾取、卸货和多段作业）的全面管理和监控。Integration Toolkit 还允许直接实时地跟踪 AMR 数据。

Integration Toolkit 具有灵活的架构，可提供多个通信通道选项。这些通信通道为系统与 AMR 车队调度和 Fleet Manager 交互方式提供了灵活多样的选项。

> **注：**Integration Toolkit 可与现有的 ARCL 通信并行运行。Integration Toolkit 并不会取代 ARCL 直接控制 AMR（一旦达到目标）。更多信息请参见《ARCL 参考指南—移动机器人》和《ARCL Enterprise Manager 集成指南》。

> **重要提示：**在生成密码之前，无法访问 Integration Toolkit。在尝试使用 Integration Toolkit 之前，请先生成密码。更多详细信息请参见第 15 页的“Integration Toolkit 密码”。

通信通道

Integration Toolkit 提供 3 种不同的通信通道。

- RESTful Web Services
- 使用 PostgreSQL 数据库的 SQL
- RabbitMQ

> **注：**虽然使用任何一种通信通道都可以实现所需功能，但建议将其组合使用，以便与客户端应用程序进行有效集成。

2.1 RESTful Web Services

REST 通信通道非常适用于实现与 Fleet Manager 的实时交互。使用这个通信通道，系统可以根据需要创建、修改和取消作业。REST 方法允许访问其他特定的 AMR 数据，并且还可以提供作业历史记录查询功能。

使用 REST 组织外部传输的数据是通过 JSON 来实现的，JSON 可用于创建请求和接收系统发出的数据。

REST 通信通道可提供通过 HTTPS 与 Integration Toolkit 进行经过加密的安全交互。更多信息请参见第 15 页的“2.5 安全”。

REST 通信通道优势

- 与 AMR 车队调度的低延迟交互。
- 临时访问 DataStore 值和作业历史记录数据。
- REST 不受机器人车体或语言类型的影响。REST 不依赖于特定驱动器或软件库。

REST 通信通道的注意事项

- RESTful API 托管在使用 8443 端口的 EM2100 设备上。
- 客户端需要提供 JSON 数据格式作为所有 POST 调用的输入，并且可能需要使用第三方软件库，以帮助创建和处理这些字符串。
- SSL 是通过需要获得客户端信任的自签名证书来实现的。
- 通过下述路径可访问大多数数据实体：
  1. **/Stream:** 当实体发生变化时，打开连接以接收更新（实时）。
  2. **/ByKey/{namekey}:** 返回与给定 namekey 相关联实体的实体信息。
  3. **/UpdatedSince?sinceTime={time millis}:** 返回自指定时间以来实体的所有更新。0 值可用于获得所有更新（自新纪元时间以来）。
- 某些数据项会被枚举，这些数据项语法和模式的正确性对于实现适当的功能至关重要。详细信息请参见本文档中的模式和示例。

  注：您可以在一些模式示例中找到（被枚举的）检查项。这些为保留项，在内部用于 Integration Toolkit 的操作。

2.2 使用 PostgreSQL 的 SQL

当需要与 AMR 车队调度进行数据库级交互时，SQL 通信通道是理想之选。PostgreSQL 是一种关系数据库管理系统，您可在该系统中使用**更新**、**插入**、**选择**和**删除**命令来监控和控制 AMR 车队调度。

SQL 通信通道优势

- 与 AMR 车队调度的数据库级交互。
- 轻松创建批量作业。
- 针对全部作业历史记录进行复杂的 SQL 查询。

SQL 通信通道的注意事项

- 当通过 SQL 通道请求任何操作时，可能会有长达 5 秒钟的延迟，因为系统必须轮询相关的表格或视图。
- 可使用标准端口 5432 访问 PostgreSQL。
- 使用 SQL 通信通道的自动化可通过编程语言和相关联软件库（例如：针对 Java 的 PostgreSQL JDBC 驱动）来实现。使用 JDBC 驱动器进行连接时，将“sslmode=require”参数包含在连接 URL 中，如下所示：“jdbc:postgresql://[IP]/IntegrationDB?sslmode=require”。
- 即使 Fleet Manager 在存储数据并保持着作业数据的全部历史记录，也不能认为其具备与具有冗余磁盘功能的数据库服务器同等的安全性。Fleet Manager 可用于获取存储在外部系统以作为备份的数据。Fleet Manager 的硬盘驱动应视为单点故障。
- 某些数据项会被枚举，这些数据项语法和模式的正确性对于实现适当的功能至关重要。详细信息请参见本文档中的模式和示例。

PostgreSQL 表格和视图

SQL 包括对表格和视图的访问。您可以执行这些视图中存在的“后插入 / 更新”逻辑，且该逻辑可控制数据如何与系统交互（在某些情况下会影响多个表格）。

视图的命名约定以“_view”结尾（例如：data_store_item_view）。

> **重要提示：**只有本文档中描述的视图才支持数据库交互。通常不支持修改或直接更新表格或模式。

2.3 RabbitMQ

RabbitMQ 通信通道是监控和控制 AMR 车队调度的一种稳健方法。

RabbitMQ 可提供一种管理机制，以确保即使出现网络问题或代理 / 客户端崩溃，也能传递出所有消息。RabbitMQ 可实时更新在外部系统上运行的进程，同时还可以在外部应用程序暂时不可用时存储消息以备未来交付。

使用 RabbitMQ 组织外部传输的数据是通过 JSON 来实现的，JSON 可用于接收系统发出的数据。

RabbitMQ 通信通道优势

- 具有读 / 写功能的异步消息传递。
- 传入和传出消息缓冲。
- 简单监控 AMR 车队调度。

RabbitMQ 通信通道的注意事项

- 使用入站通道（例如：用于创建作业的通道）时，系统需要提供与 REST 使用的格式相同的 JSON。
- 将 JSON 格式的数据发送至 RabbitMQ 队列必须使用软件库并执行代码。
- 使用该通道时，RabbitMQ 管理控制台不可用，并且需要进行一定程度的编程来实现与系统的交互。
- 缓冲的消息会在 12 个小时后过期。如果客户端应用程序在这些缓冲的消息过期之前未使用，这些消息将会丢失。
- 某些数据项会被枚举，这些数据项语法和模式的正确性对于实现适当的功能至关重要。详细信息请参见本文档中的模式和示例。

2.4 软件管理

SetNetGo 提供了管理 Integration Toolkit 软件安装的区域。提供下述软件管理功能。

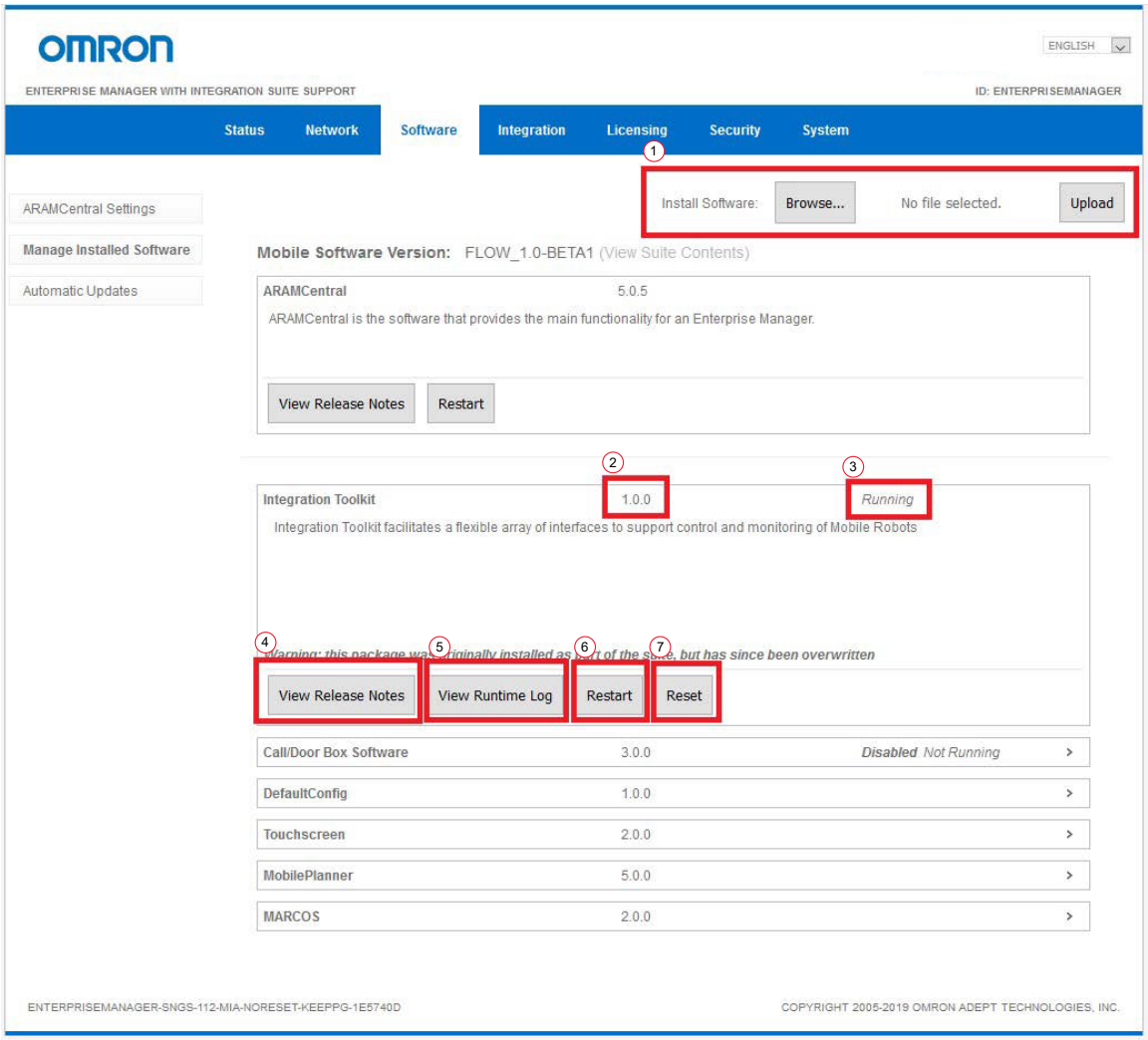

图 2-1. Integration Toolkit 软件管理

表 2-1.Integration Toolkit 软件管理描述

项目	说明
1	选择并上传新的 FLOW Core 软件文件（参见下面的注释）。
2	已安装 Integration Toolkit 的当前版本。
3	Integration Toolkit 的运行状态（正在运行或未运行）。
4	打开对话框，显示 Integration Toolkit 的发布说明。
5	打开对话框，显示 Integration Toolkit 的运行时日志，以用于诊断目的。
6	重新启动 Integration Toolkit 服务（参见下面的注释）。

项目	说明
7	将 Integration Toolkit 重置为默认值。 **重要提示：**这将删除数据库，然后又重建数据库。所有数据都将丢失，如作业历史记录和当前作业数据。

注：Integration Toolkit 功能依赖于 ARAMCentral 的运行。任何停止 Integration Toolkit 或 ARAMCentral 运行的操作都有可能对数据完整性和功能产生影响。建议在计划的软件运行中断期间暂停车队调度活动。

2.5 安全

Integration Toolkit 对所有通信通道都采用同一种方式来实现其安全性。自签名证书用于在客户端应用程序和 Integration Toolkit 之间建立加密连接。对于所有通信通道，用于进行身份验证的用户 ID 和密码均相同。

该安全机制并非可选项，如果没有这种加密连接，就无法配置通信。

注：如果自签名证书的安全传输存在问题（因为客户端并不会验证证书），则应该手动移动、加载和信任该证书。

Integration Toolkit 密码

要生成新密码，请使用 MobilePlanner 或网页浏览器访问 Fleet Manager 的 SetNetGo 界面。有关访问 SetNetGo 的更多信息，请参见《Fleet Operations Workspace Core 用户指南》。

Integration Toolkit 的用户名通常固定为“toolkitadmin”。

重要提示：Integration Toolkit 刚安装时未设置密码。在生成密码之前，无法访问 Integration Toolkit。在首次尝试使用 Integration Toolkit 之前，请先生成密码。

使用下述程序生成 Integration Toolkit 的新密码。

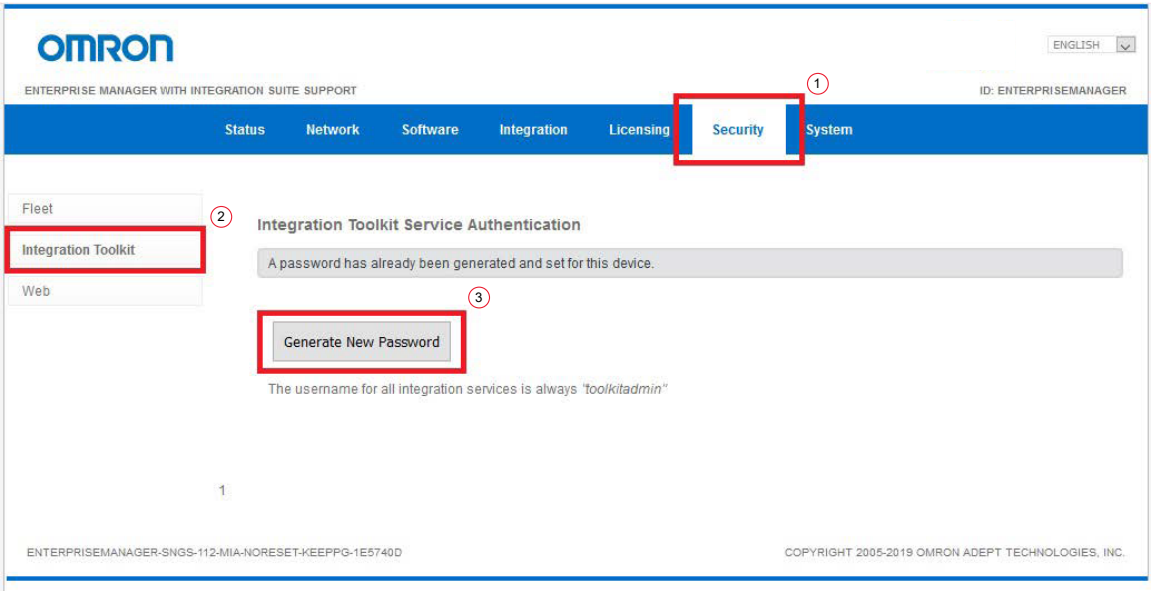

图 2-2. 在 SetNetGo 中设置或更改密码

1. 访问 SetNetGo 界面后，单击“安全”选项卡。
2. 在安全区域中选择 Integration Toolkit 部分。
3. 单击“**生成新密码**”按钮。

   重要提示：当按下“**生成新密码**”按钮时，会显示警告消息，表明此操作将导致 Integration 和 Fleet Management 服务重启。只有在重启不会导致问题的情况下才能继续操作。

4. 密码更改完成并重启服务后，会显示新密码以便于记录。复制该密码，并将其保存在安全位置，以供未来使用。

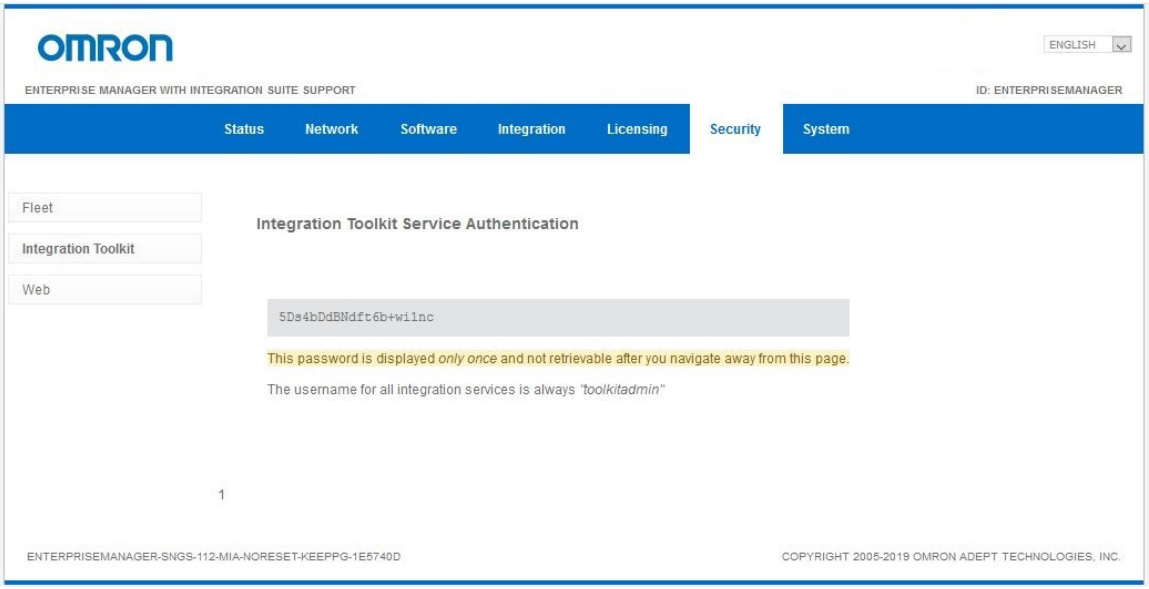

图 2-3. 新密码显示

2.6 Namekey 概念

Namekey 是在内部和外部用于跟踪数据实体的唯一标识符。利用这些特性，可以在与 Integration Toolkit 交互的自动化系统中进行更出色的控制和跟踪。

通过使用 namekey，可为程序员提供明确的实体创建和跟踪方法。这可提供一种检索请求状态或语句的机制。当使用通信通道创建实体时（如创建作业时），提供 namekey 为可选项。如果未提供 namekey，系统将提供自生成的 namekey。

如果需要，系统将创建唯一的 namekey（例如：在作业段和历史记录条目中），以支持查询功能。

第 3 章： 入门

您可以借助下述示例来创建一个简单的拾取和卸货作业请求，以发送至与您系统中 Fleet Manager 相关联的 AMR。

> **注：**这些示例使用的目标名称和 namekey 可能在您的应用程序中不存在。请使用适合您系统的目标名称和 namekey。

3.1 拾取卸货作业—REST 示例

下述 JSON 对象将根据 p5 和 p34 的 Flow Manager 地图中的目标，创建一个新的拾取卸货作业。该作业的拾取优先级为 10，卸货优先级为 20。TestPickup101 为实体的 namekey。

namekey 和 jobId 将自动生成，因为它们并未在 JSON 命令主体参数中被指定。

REST 命令详情

- 方法：POST
- 端点：https://[IP]:8443
- 资源：/PickupDropoff

JSON 命令主体

要创建拾取卸货作业，请将以下 JSON 对象主体与 POST 请求 https://[IP]:8443/PickupDropoff 共同发出：

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

cURL 字符串

发出以下 cURL 字符串，以创建拾取卸货作业：

```
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}"
```

响应示例

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

3.2 拾取卸货作业 SQL 示例

下述 SQL 语句将根据 p5 和 p34 的预定位置，创建一个新的拾取卸货作业。该作业的拾取优先级为 10，卸货优先级为 20，新作业标识符为 Test1。

SQL 语句

要创建拾取卸货作业，请创建下述语句：

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

响应

```
INSERT 0 1
```

3.3 RabbitMQ Python 示例

以下 Python 示例使用 RabbitMQ 与 Integration Toolkit 交互。可以通过改变队列名称和消息来修改这些示例，以实现其他功能。

重要提示：这些示例并未明确要求使用安全证书，但将使用从 Integration Toolkit 服务器获得的安全证书，而不尝试进行验证。连接仍然进行了加密，且使用用户 ID 和密码进行身份验证。当存在 SSL 证书，且程序员希望强制使用该证书时，应使用 cert_reqs=ssl.CERT_REQUIRED。

注：必须针对应用程序特定的条件更改示例值，如下述脚本注释中所述。

附加信息：在使用这些示例之前，必须先安装所有必需的软件库和软件包。

本文档中的示例是使用 Python 3 和 Pika 0.12.0 创建的。

将消息发布至 inbound.PickupDroppoff 队列

下述示例将拾取卸货消息发布至 inbound.PickupDropoff 队列。

```
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)
```

使用 outbound.Job 队列的消息

下述示例使用 outbound.Job 队列的消息。

> **注：**在 subscribe_queue_and_print 方法中，设置 **no_ack =True** 表示确认或 **no_ack =False** 表示不确认已使用的消息。

```
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)
```

第 4 章： DataStore

DataStore 项是 Fleet Manager 和每个 AMR 的数据点，可根据状态及其他监控目的对其进行评估。它们是 Fleet Manager 和每个 AMR 特定的项。每个 DataStore 项都会对数据点进行描述，即数据类型、数据名称、数据来源等。

以下是 DataStore 项的一些示例：

- 某个 AMR 累计已完成的作业数量
- 某个 AMR 的行驶距离
- 某个 AMR 的当前位置
- 某个 AMR 的当前电池状态
- SetNetGo 软件版本的相关信息

在启动时，Integration Toolkit 会生成或更新其可用 DataStore 项的列表（为所有 AMR 保存一个通用列表）。要查看您系统中可用的所有 DataStoreItem 项的完整列表，请发出 REST 调用或查看 data_store_item_view 视图。更多信息请参见第 23 页的“4.1 常见的 DataStore 用例”和第 12 页的“2.2 使用 PostgreSQL 的 SQL”。

这些 DataStore 项的值可通过订阅模式（三种通道均提供）或使用 DataStoreValueLatest REST 命令从 Integration Toolkit 中获得，该命令会忽略订阅状态，从相关实体中获取最新的值。通过订阅模式，用户可为 DataStore 项指定订阅间隔，而 Integration Toolkit 将按照指定的订阅间隔从目标实体中获取 DataStore 值，并且可通过所有通信通道访问该值。

SQL 和 REST 都将返回最后一个订阅期的最新 DataStore 值。流媒体接口将在每次 DataStore 值发生改变时返回条目。如果 DataStore 项的值未发生变化，将不会更新该值。

> **重要提示：**应注意避免低于 1 秒的高频率订阅率，以防止对大量数据实体的超额订阅。虽然已经在更高的级别进行了测试，但仍应该避免每秒产生 200 或更多更新值的订阅率。

4.1 常见的 DataStore 用例

以下为使用 DataStore 的常见用例。

手动获取“可能的 DataStore 项列表”

该用例通常在初始编程阶段执行，以了解可以从系统中获取哪些数据。程序员应查询 data_store_item_view，以获得在自动运行时期间可以访问的最新 DataStore 项列表及详细信息。

该信息也可以通过 REST 调用（/DataStoreItem）获得，但 SQL 查询方法为程序员查看该信息提供了一种更简单的方法。

获取单个当前值，以便在自动化系统中使用

该用例类似于现有的 ARCL 功能。利用该功能，Integration Toolkit 可与 AMR 或 Fleet Manager 通信，并提供最新数据，同时忽略任何订阅状态。

这可使用 /DataStoreValueLatest 路径通过 REST 通道来实现。通过这种方式获得的值会被发布到 RabbitMQ 队列中，并以与订阅该值相同的方式发送到数据库中。这是获取 DataStore 值常见的用例。

通过流媒体（http 或 RabbitMQ）获取值

该用例涵盖了需要使用任意流送选项将 DataStore 值按照设定时间表发送至某个应用程序的情况，以及需要提前创建订阅的情况。请参见第 27 页的“4.4 订阅 DataStore 值”。创建订阅后，将按定义的订阅间隔自动更新该值。

您也可以通过 SQL 和 REST 通道获得具有订阅的值，但这些调用只会返回最后获得的值。

4.2 DataStore 模式

DataStore 模式由 SubscriptionConfig、DataStoreValue 和 DataStoreItem 组成。

DataStoreItem

DataStoreItem 用于获取 Fleet Manager 或 AMR 上可用的各个项目的相关信息，以便进行信息检索。

SubscriptionConfig

SubscriptionConfig 用于命令 Integration Toolkit 从 AMR 和 Fleet Manager 获取值，以便使用这些值。

更多信息请参见第 27 页的“4.4 订阅 DataStore 值”。

DataStoreValue

DataStoreValue 提供 DataStore 项的最新记录值。该值以字符串形式返回，且在使用之前可能需要进行转换。DataStoreItem 实体可用于指示值类型。更多信息请参见第 28 页的“4.5 获取 DataStore 值”。

4.3 获取 DataStore 项的相关信息

下表描述了 DataStore 项模式。

表 4-1.DataStore—数据项详情

项目	详细信息	数据类型
namekey	DataStore 项 namekey 的预定义字符串。 附加 AMR 名称作为 AMR 特定的值，如下所示。 • AMR 特定的值— “namekey：AMR 名称” • Fleet Manager 值— “namekey”	字符串
itemId	DataStore 项内部 ID（保留供内部使用）	整型
source	Fleet Manager—空 非 Fleet Manager—AMR 名称	字符串
category	Fleet Manager 中的项类别。	
groupName	Fleet Manager 中的 DataStore 组名称。	
groupDescr	DataStore 组的描述。	
itemName	DataStore 项名称。	
displayName	DataStore 显示名称。	
type	DataStore 项类型（字符串、长整型、整型、双精度型、布尔值）	
description	DataStore 项描述。	

使用 REST

使用下述 REST 调用来获取 DataStore 项的相关信息。

表 4-2.DataStore 项 REST 调用

方法	资源	功能
GET	/DataStoreItem/ByKey/{namekey}	按 namekey 获取 DataStoreItem。
GET	/DataStoreItem/UpdatedSince?sinceTime={time millis}	获取自给定时间以来已更新的 DataStoreItem 实体列表。
GET	/DataStoreItem/BySource/{Source}	获取按来源筛选的 DataStoreItem 实体列表。
GET	/DataStoreItem/ByItemName/{ItemName}	获取按 itemName 筛选的 DataStoreItem 实体列表。
GET	/DataStoreItem/ByType/{Type}	获取按类型筛选的 DataStoreItem 实体列表。
GET	/DataStoreItem/ByDisplayName/{DisplayName}	获取按 displayName 筛选的 DataStoreItem 实体列表。

方法	资源	功能
GET	/DataStoreItem/ByGroupName/{GroupName}	获取按 groupName 筛选的 DataStoreItem 实体列表。
GET	/DataStoreItem/ByCategory/{Category}	获取按 Category 筛选的 DataStoreItem 实体列表。

DataStore 项 JSON 模式

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

DataStore 项 JSON 示例

在发出 GET 请求“https://[IP]:8443/DataStoreItem/ByItemName/DateAndTime”之后的响应示例

```
{
        "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 命令字符串示例

获取 DateAndTime DataStore 项的相关信息：

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

使用 SQL

获取 DataStore 项的相关信息：

```
SELECT * FROM data_store_item_view;
```

4.4 订阅 DataStore 值

订阅 DataStore 值就是提醒系统您关注某些 DataStore 项的进程。

下表描述了 DataStore 订阅项模式。

注：/DataStoreValueLatest REST 调用不要求订阅。更多信息请参见第 28 页的“4.5 获取 DataStore 值”。

表 4-3.DataStore 订阅—数据项详情

项目	详细信息	数据类型
namekey	DataStore 项 namekey 的预定义字符串。	字符串
subscriptionInterval	订阅间隔 单位：ms、s、m、h、d 语法：“1s”、“2m”等 最低值：200 ms 默认值：0 ms **注：**当设置为 0 时，关闭订阅。	

使用 REST

使用下述 REST 调用来获取或设置订阅。

表 4-4.DataStore 订阅 REST 调用

方法	资源	功能
GET	/SubscriptionConfig/ByKey/{namekey}	按 namekey 获取 SubscriptionConfig。
GET	/SubscriptionConfig/UpdatedSince?sinceTime={time millis}	获取自给定时间以来已更新的 SubscriptionConfig 实体列表。
GET	/SubscriptionConfig/Stream	监控所有 SubscriptionConfig 更新。
PUT	/SubscriptionConfig	更新 SubscriptionConfig。

注：Integration Toolkit 会在启动之时为每个 DataStore 项创建一个条目。DataStore 订阅只会更新现有条目，这就是无可用 POST 方法的原因所在。

DataStore 订阅 JSON 模式

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

DataStore 订阅 JSON 示例

使用 1 秒更新和一个 PUT 请求 https://[IP]:8443/SubscriptionConfig 来订阅 ARAM DataStore 项的值：

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

cURL 命令字符串示例

使用 1 秒更新和一个 POST 请求来订阅 ARAM DataStore 项：

```
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\"}"
```

使用 SQL

使用 SQL 更新订阅间隔可通过更新“subscription_config_view”视图（如下所示）中的“subscription_ interval”列来实现。

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

注：该视图还可用于确定这些设置的当前值。

使用 RabbitMQ

outbound.SubscriptionConfig

inbound.SubscriptionConfig

4.5 获取 DataStore 值

下表描述了 DataStore 值项模式。

表 4-5.DataStore—值项详情

项目	详细信息	数据类型
namekey	DataStore 项 namekey 的预定义字符串。 附加 AMR 名称作为 AMR 特定的值，如下所示。 • AMR 特定的值— “namekey：AMR 名称” • Fleet Manager 值— “namekey”	字符串
值	当前值。	

使用 REST

使用下述 REST 调用来获取 DataStore 值。

表 4-6.DataStore 值 REST 调用

方法	资源	功能
GET	/DataStoreValue/ByKey/{namekey}	按 namekey 获取 DataStoreValue。
GET	/DataStoreValue/UpdatedSince?sinceTime= {time millis}	获取自给定时间以来已更新的 DataStoreValue 实体列表。
GET	/DataStoreValue/Stream	监控 DataStoreValue 更新。

表 4-7.DataStoreValueLatest 项 REST 调用

方法	资源	功能
GET	/DataStoreValueLatest/{DataStore item name}	针对指定名称的 DataStore 项，返回（不订阅）一个一次性值。
GET	/DataStoreValueLatest/{DataStore item name}:{AMR name}	针对在指定名称的 AMR 上指定名称的 DataStore 项，返回（不订阅）一个一次性值。
GET	/DataStoreValueLatest/{DataStore item name}:*	针对所有 AMR 上指定名称的 DataStore 项，返回（不订阅）一个一次性值。 该方法将创建 2 秒的延迟，同时收集所有值。

DataStore 值 JSON 模式

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

DataStore 值 JSON 示例

在发出 GET 请求“https://[IP]:8443/DataStoreValue/ByKey/ARAM”之后的响应示例

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

cURL 命令字符串示例

获取 ARAM DataStore 值（已订阅）的相关信息：

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

获取 ARAM DataStore 值（未订阅，临时）的相关信息：

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

使用 SQL

获取 ARAM DataStore 值的相关信息：

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

使用 RabbitMQ

每个 DataStore 项都存在队列（例如：outbound.datastore.ARAM），而所有值的更改都可以根据 outbound.DataStoreValue 进行跟踪。

第 5 章： 作业

Integration Toolkit 支持下述作业功能。

- 作业创建方法：拾取、拾取卸货、卸货和作业请求。
- 在部分级别上修改作业（目标和优先级变更）。
- 取消整个作业。
- 取消（删除）多段作业的单段作业。
- 如果创建作业请求时未使用 namekey 或 jobId，Integration Toolkit 将自动为这些数据项生成唯一值。

  注：创建卸货作业会覆盖 Fleet Manager 的 AMR 选择逻辑。为此，请选择性地使用该作业类型。

  注：作业请求的功能与 queueMulti 相同。ARCL 命令这种作业类型将使 AMR 排队完成多个目标的多个拾取和卸货作业。

作业创建步骤

作业创建分两步：

1. 发出作业请求，Integration Toolkit 接受请求之后，会发送确认回复。

   注：只有使用 REST 时，才会发送确认回复。

2. 在 Integration Toolkit 接受作业请求之后，向 Fleet Manager 发送调用命令，以开始实施作业。开始实施作业时，作业和作业阶段条目在 Integration Toolkit 中可用，从而可以实时跟踪作业。

5.1 常见的作业创建用例

以下为使用作业创建的常见用例。

创建单项作业

该用例涵盖通过仅指定目标的方式创建作业。如果系统无需跟踪作业或验证状态，则使用任何一种通信通道都可以通过单个调用（或插入 SQL 行）创建作业。

创建作业并验证 Fleet Manager 是否接受该作业

创建作业请求之后，可通过检查 SQL 条目或通过使用调用最初返回的 namekey 在相同路径的 /ByKey URI 上运行 GET 调用来获得该作业请求的状态。

创建并跟踪作业

通过生成唯一 namekey，并在作业创建步骤中将其用作为 JobId 来实现该功能。然后，在 Fleet Manager 处理作业时，可使用该 namekey 来获取作业相关信息（更多信息请参见第 42 页的“5.3 作业监控”）。

或者，系统可以在没有 JobId 的情况下创建作业，然后使用 Fleet Manager 查询作业创建方法，以获取可用于跟踪该作业的 assignedJobId。

5.2 作业创建

下面提供了各种作业创建方法的相关信息。

创建拾取作业

本节介绍了如何生成新的拾取作业或删除现有的拾取作业。此外，还介绍了如何监控现有的拾取作业。这些功能可用于管理 AMR 车队调度的拾取作业。

下表描述了拾取项模式。

表 5-1. 拾取项模式

项目	详细信息	数据类型
namekey	拾取作业实体的唯一标识符。 对 POST/ 插入 / 发布方法来说为可选。如果省略，Integration Toolkit 会自动生成。	字符串
jobId	分配给作业的 JobId。 对 POST/ 插入 / 发布方法来说为可选。	
goal	拾取目标的名称。 对 POST/ 插入 / 发布方法来说为必选。	
priority	拾取部分的优先级。 对 POST/ 插入 / 发布方法来说为可选。如果忽略，Fleet Manager 会分配默认优先级。	整型
assignedJobId	已分配给作业的 JobId。 如果在创建作业时未提供 JobId，Fleet Manager 会自动创建 assignedJobId。	字符串
status	来自队列管理器的“成功”或失败消息。	

使用 REST 的拾取作业

使用下述 REST 调用生成或删除现有拾取作业。这些调用还可用于获取排队等候 Integration Toolkit 执行的拾取作业的信息。

表 5-2. 拾取作业 REST 调用资源

方法	资源	功能
POST	/Pickup	创建拾取作业。
GET	/Pickup/UpdatedSince?sinceTime={time millis}	获取自给定时间以来已更新的拾取作业实体列表。
GET	/Pickup/Stream	监控所有拾取作业更新。
GET	/Pickup/ByKey/{namekey}	按 namekey 获取拾取作业信息。
GET	/Pickup/ByJobId/{JobId}	获取按 JobId 筛选的拾取作业实体列表。
GET	/Pickup/ByStatus/{Status}	获取按 Status 筛选的拾取作业实体列表。
GET	/Pickup/ByAssignedJobId/{AssignedJobId}	获取按 AssignedJobId 筛选的拾取作业实体列表。
DELETE	/Pickup/{namekey}	按 namekey 删除拾取作业（参见下面的注释）。

注：仅在需要重复使用 namekey 时才要求使用 DELETE 方法。

拾取作业 JSON 模式

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

拾取作业 JSON 示例

使用 POST 请求“https://[IP]:8443/Pickup”创建拾取作业：

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

cURL 命令字符串示例

使用 POST 请求创建拾取作业：

```
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\"}"
```

使用 SQL 的拾取作业

创建拾取作业：

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

使用 RabbitMQ 的拾取作业

inbound.Pickup

outbound.Pickup

创建拾取卸货作业

本节介绍了如何生成新的拾取卸货作业或删除现有的拾取卸货作业。此外，还介绍了如何监控现有的拾取卸货作业。这些功能可用于管理 AMR 车队调度的拾取卸货作业。

下表描述了拾取卸货项模式。

表 5-3. 拾取卸货项模式

项目	详细信息	数据类型
namekey	拾取卸货作业实体的唯一标识符。 对 POST/ 插入 / 发布方法来说为可选。如果省略，Integration Toolkit 会自动生成。	字符串
jobId	分配给作业的 JobId。 对 POST/ 插入 / 发布方法来说为可选。	
pickupGoal	拾取目标的名称。 对 POST/ 插入 / 发布方法来说为必选。	
pickupPriority	拾取部分的优先级。 对 POST/ 插入 / 发布方法来说为可选。如果忽略，Fleet Manager 会分配默认优先级。	整型
dropoffGoal	卸货目标的名称。 对 POST/ 插入 / 发布方法来说为必选。	字符串

项目	详细信息	数据类型
dropoffPriority	卸货部分的优先级。 对 POST/ 插入 / 发布方法来说为可选。如果忽略，Fleet Manager 会分配默认优先级。	整型
assignedJobId	已分配给作业的 JobId。 如果在创建作业时未提供 JobId，Fleet Manager 会自动创建 assignedJobId。	字符串
status	来自队列管理器的“成功”或失败消息。	字符串

使用 REST 的拾取卸货作业

使用下述 REST 调用生成或删除现有拾取卸货作业。这些调用还可用于获取排队等候 Integration Toolkit 执行的拾取卸货作业信息。

表 5-4. 拾取卸货作业 REST 调用资源

方法	资源	功能
POST	/PickupDropoff	创建拾取卸货作业。
GET	/PickupDropoff/UpdatedSince?sinceTime= {time millis}	获取自给定时间以来已更新的拾取卸货作业实体列表。
GET	/PickupDropoff/Stream	监控所有拾取卸货作业更新。
GET	/PickupDropoff/ByKey/{namekey}	按 namekey 获取拾取卸货作业信息。
GET	/PickupDropoff/ByJobId/{JobId}	获取按 JobId 筛选的拾取卸货作业实体列表。
GET	/PickupDropoff/ByStatus/{Status}	获取按 Status 筛选的拾取卸货作业实体列表。
GET	/PickupDropoff/ByAssignedJobId/{AssignedJobId}	获取按 AssignedJobId 筛选的拾取卸货作业实体列表。
DELETE	/PickupDropoff/{namekey}	按 namekey 删除拾取卸货作业信息。

拾取卸货作业 JSON 模式

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

拾取卸货作业 JSON 示例

使用 POST 请求“https://[IP]:8443/PickupDropoff”创建拾取卸货作业：

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

cURL 命令字符串示例

使用 POST 请求创建拾取卸货作业：

```
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\"}"
```

使用 SQL 的拾取卸货作业

创建拾取卸货作业：

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

使用 RabbitMQ 的拾取卸货作业

inbound.PickupDropoff

outbound.PickupDropoff

创建卸货作业

本节介绍了如何生成新的卸货作业或删除现有的卸货作业。此外，还介绍了如何监控现有的卸货作业。这些功能可用于管理 AMR 车队调度的卸货作业。

下表描述了卸货项模式。

> **注：**卸货作业是一种绕过 Fleet Manager 队列管理功能的方法，应尽可能避免。未来版本可能会放弃使用该功能。

表 5-5. 卸货项模式

项目	详细信息	数据类型
namekey	卸货作业实体的唯一标识符。 对 POST/ 插入 / 发布方法来说为可选。如果省略，Integration Toolkit 会自动生成。	字符串
jobId	分配给作业的 JobId。 对 POST/ 插入 / 发布方法来说为可选。	
priority	卸货部分的优先级。 对 POST/ 插入 / 发布方法来说为可选。如果忽略，Fleet Manager 会分配默认优先级。	整型
robot	用于此卸货作业请求的 AMR。	字符串
goal	卸货目标的名称。 对 POST/ 插入 / 发布方法来说为必选。	
assignedJobId	已分配给作业的 JobId。 如果在创建作业时未提供 JobId，Fleet Manager 会自动创建 assignedJobId。	
status	来自队列管理器的“成功”或失败消息。	

使用 REST 的卸货作业

使用下述 REST 调用生成或删除现有卸货作业。这些调用还可用于获取排队等候 Integration Toolkit 执行的卸货作业信息。

表 5-6. 卸货作业 REST 调用资源

方法	资源	功能
POST	/Dropoff	创建卸货作业。
GET	/Dropoff/UpdatedSince?sinceTime={time millis}	获取自给定时间以来已更新的卸货作业实体列表。
GET	/Dropoff/Stream	监控所有卸货作业更新。
GET	/Dropoff/ByKey/{namekey}	获取按 namekey 筛选的卸货作业。
GET	/Dropoff/ByJobId/{JobId}	获取按 JobId 筛选的卸货作业实体列表。
GET	/Dropoff/ByStatus/{Status}	获取按 Status 筛选的卸货作业实体列表。
GET	/Dropoff/ByAssignedJobId/{AssignedJobId}	获取按 AssignedJobId 筛选的卸货作业实体列表。
GET	/Dropoff /ByRobot/{AMR}	获取按 AMR 筛选的卸货作业实体列表。

卸货作业 JSON 模式

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

卸货作业 JSON 示例

使用 POST 请求“https://[IP]:8443/Dropoff”创建卸货作业：

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

cURL 命令字符串示例

使用 POST 方法创建卸货作业：

```
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\"}"
```

使用 SQL 的卸货作业

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

使用 RabbitMQ 的卸货作业

inbound.Dropoff

outbound.Dropoff

创建作业请求作业类型（多段作业）

作业请求是部分数量可变的作业（如 ARCL 的 queueMulti）。

本节介绍了以下作业请求功能。这些功能可用于管理 AMR 车队调度的作业请求。

- 通过使用新的作业请求，使 AMR 排队完成多个目标的多个拾取和卸货作业。
- 监控现有作业请求。
- 删除现有作业请求。

下表描述了作业请求项模式。

表 5-7. 作业请求项模式

项目	详细信息	数据类型
namekey	作业请求实体的唯一标识符。 对 POST/ 插入 / 发布方法来说为可选。如果省略，Integration Toolkit 会自动生成。 对 SQL INSERT 语句来说为必需。	字符串
jobId	分配给作业的 JobId。 对 POST/ 插入 / 发布方法来说为可选。	
defaultPriority	默认优先级选择。 对 POST/ 插入 / 发布方法来说为必选。 当设置为真时，将覆盖该作业请求中设置的任何优先级。	布尔值
目标 （枚举项）参见下面的注释	作业部分的目标名称。 请参见下面的作业请求 JSON 模式。 对 POST/ 插入 / 发布方法来说为必需。	字符串
priority	拾取部分的优先级。 对于 POST 方法来说为可选。如果忽略，Fleet Manager 会分配默认优先级。	整型
segmentType （枚举项）参见下面的注释	必须更换为作业部分类型：“pickupGoal”或“dropoffGoal”。 请参见下面的作业请求 JSON 示例。 对 POST/ 插入 / 发布方法来说为必需。	字符串
assignedJobId	已分配给作业的 JobId。 如果在创建作业时未提供 JobId，Fleet Manager 会自动创建 assignedJobId。	
status	来自队列管理器的“成功”或失败消息。	

注：每个作业请求详情中的目标和 segmentType 项都显示为一个类型值对，而非两个单独条目。如果部分类型为拾取，则 segmentType 和目标显示为“pickupGoal”：“**目标名称**”。如果部分类型为卸货，则 segmentType 和目标显示为“dropoffGoal”：“**目标名称**”。正确的语法请参见下述 JSON 示例。

使用 REST 的作业请求

使用下述 REST 调用生成或删除现有作业请求。这些调用还可用于获取排队等候 Integration Toolkit 执行的作业请求信息。

> **注：**要想获取目标、优先级和部分类型等作业详情，请使用表 5-9 中列出的 JobRequestDetails 资源。

表 5-8. 作业请求 REST 调用资源

方法	资源	功能
POST	/JobRequest	创建作业请求。
GET	/JobRequest/UpdatedSince?sinceTime={time millis}	获取自给定时间以来已更新的作业请求实体列表。
GET	/JobRequest/Stream	监控所有作业请求更新。
GET	/JobRequest/ByKey/{namekey}	按 namekey 获取作业请求。
GET	/JobRequest/ByJobId/{JobId}	获取按 JobId 筛选的作业请求实体列表。
GET	/JobRequest/ByStatus/{Status}	获取按状态筛选的作业请求实体列表。
GET	/JobRequest/ByAssignedJobId/{AssignedJobId}	获取按 AssignedJobId 筛选的作业请求实体列表。
DELETE	/JobRequest/{namekey}	按 namekey 删除作业请求。

表 5-9. 作业请求详情 REST 调用资源

方法	资源	功能
GET	/JobRequestDetail/ByKey/{namekey}	按 namekey 获取 JobRequestDetail。
GET	/JobRequestDetail/UpdatedSince?sinceTime={time millis}	获取自给定时间以来已更新的 JobRequestDetail 实体列表
GET	/JobRequestDetail/ByJobRequest/{JobRequest namekey}	获取按 JobRequest 筛选的 JobRequestDetail 列表。

作业请求 JSON 模式

```
{
        "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"
}
```

作业请求 JSON 示例

使用 POST 请求“https://[IP]:8443/JobRequest”创建作业请求：

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

cURL 命令字符串示例

使用 POST 请求创建作业请求：

```
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}]}"
```

使用 SQL 的作业请求

作业请求包括插入至 job_request_view 和 job_request_detail_view。为此，在 job_request_view 中创建了作业之后，您需要在 job_request_detail_view 中添加与该作业相关联的部分。您需要在同一事务中创建这些数据库插入。

使用 POST 方法创建作业请求：

```
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;
```

使用 RabbitMQ 的作业请求

inbound.JobRequest

outbound.JobRequest

5.3 作业监控

您可以使用任意通信通道完成作业监控。可从 REST 数据流端点或 RabbitMQ 通道获取实时更新。可使用非数据流 REST 调用以及从数据库按需获取数据。

监控作业时，请记住作业创建的两步性质（请参见第 31 页的“作业创建步骤”）。如果作业创建失败（例如，当地图中不存在指定目标时），不会创建任何作业，并且会在作业创建记录的状态字段中报告失败，同时不会产生任何作业或 job_segment 记录。鉴于这一功能，在使用作业表、REST 路径或 RabbitMQ 队列监控作业进程之前，检查已创建作业的状态可能会比较有利。

作业监控模式实体

作业监控模式包含以下实体：

- 作业—获取已排队作业的状态信息。
- 作业部分—获取已排队作业部分的状态信息。
- 作业历史记录—获取作业的历史记录信息。
- 作业部分历史记录—获取作业部分的历史记录信息。

作业监控详情

本节介绍了有关监控所有现有作业的详细信息。

下表描述了作业监控项模式。

表 5-10. 作业监控项模式

项目	详细信息	数据类型
namekey	作业实体的唯一标识符。	字符串
jobId	Fleet Manager 分配的 jobId。	
queuedTimestamp	作业排队的时间。	整型
jobType	作业类型：拾取（P）、卸货（D）、拾取卸货（PD）或多个目标（M）。	字符串
lastAssignedRobot	最后分配给此作业的 AMR 名称。	
cancelReason	作业被取消的原因。 仅当由客户端在作业取消期间提供时，由 Fleet Manager 提供。	
status	作业状态：Pending、InProgress、Completed、Cancelled、Cancelling 或 Modifying。	
completedTimestamp	作业完成的时间。	整型
linkedJob	保留以备未来使用（空白）。	字符串
failCount	作业失败的次数。 只有在作业失败时才会显示。	整型

使用 REST 监控作业

使用下述 REST 调用监控现有作业。

表 5-11. 作业监控 REST 调用资源

方法	资源	功能
GET	/Job/ByKey/{namekey}	按 namekey 获取作业。
GET	/Job/UpdatedSince?sinceTime={time millis}	获取自给定时间以来已更新的作业实体列表。
GET	/Job/Stream	监控所有作业更新。
GET	/Job/History?sinceTime={time millis}	获取自给定时间以来的作业历史记录实体列表。
GET	/Job/History?sinceTime={time millis}&namekey={namekey}	获取自给定时间以来特定 namekey 的作业历史记录实体列表。
GET	/Job/ByJobId/{JobId}	获取按 JobId 筛选的作业实体列表。
GET	/Job/ByLastAssignedRobot/{LastAssignedRobot}	获取按 lastAssignedRobot 筛选的作业实体列表。
GET	/Job/ByStatus/{Status}	获取按状态筛选的作业实体列表。

作业监控 JSON 模式

```
{
       "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"
}
```

作业监控 JSON 示例

在发出 GET 请求“https://[IP]:8443/Job/ByStatus/Completed”之后的响应示例：

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

cURL 命令字符串示例

使用 GET 方法获取已完成作业的相关信息：

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

使用 SQL 监控作业

获取已完成作业的相关信息：

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

使用 RabbitMQ 监控作业

outbound.Pickup

outbound.Dropoff

outbound.PickupDropoff

outbound.JobRequest

outbound.Job

> **其他信息：**outbound.Pickup/Dropoff/PickupDropoff/JobRequest 可提供这些请求实体的更新信息。outbound.Job 可提供与从 Integration Toolkit 发送至 Fleet Manager 的作业相关的更新信息。

作业部分监控详情

以下部分介绍了监控作业部分的详细信息。

下表描述了作业部分监控项模式。

表 5-12. 作业部分项模式

项目	详细信息	数据类型
namekey	作业部分实体的唯一标识符。	字符串
segmentId	作业 segmentId。	
segmentType	拾取或卸货的作业部分类型。	
seq	作业部分序号。	整型
status	作业部分的状态：Pending、InProgress、Interrupted、Completed、Cancelled、Cancelling、Failed、Modifying 或 Modified。	字符串
subStatus	作业部分的子状态：Unallocated、Allocated、BeforePickup、BeforeDropOff、BeforeEvery、Before、Driving、After、AfterEvery、AfterPickup、AfterDropOff、Buffering、Buffered、None、ContainsCancelReason、ContainsLinkedReason、AssignedRobotOffline、NoMatchingRobotForLinkedJob、NoMatchingRobotForOtherSegment 或 NoMatchingRobot。	
作业	相关联作业的 namekey。	
robot	已分配了作业部分的 AMR 的名称。	
linkedJobSegment	上一个作业 segmentId 的作业 segmentId（如果适用）。 对于第一个作业部分，该项值为空。	
priority	作业部分优先级。	整型

项目	详细信息	数据类型
goalName	作业部分目标名称。	字符串
cancelReason	作业部分被取消的原因。 仅当由客户端在作业取消期间提供时，由 Fleet Manager 提供。	

使用 REST 监控作业部分

使用下述 REST 调用监控现有作业部分。

表 5-13. 作业部分监控 REST 调用资源

方法	资源	功能
GET	/JobSegment/ByKey/{namekey}	按 namekey 获取作业部分。
GET	/JobSegment/UpdatedSince?sinceTime={time millis}	获取自给定时间以来已更新的作业部分实体列表。
GET	/JobSegment/Stream	监控所有作业部分更新。
GET	/JobSegment/History?sinceTime={time millis}	获取自给定时间以来已更新的作业部分历史记录实体列表。
GET	/JobSegment/History?sinceTime={time millis} &namekey={namekey}	获取自给定时间以来已更新的指定 namekey 的作业部分历史记录实体列表。
GET	/JobSegment/ByStatus/{Status}	获取按状态筛选的作业部分实体列表。
GET	/JobSegment/ByJob/{Job}	获取按作业筛选的作业部分实体列表。
GET	/JobSegment/ByRobot/{AMR}	获取按 AMR 筛选的作业部分实体列表。
GET	/JobSegment/ByGoalName/{GoalName}	获取按 goalName 筛选的作业部分实体列表。

作业部分监控 JSON 模式

```
{
        "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"
}
```

作业部分监控 JSON 示例

在发出 GET 请求“https://[IP]:8443/JobSegment/ByStatus/Completed”之后的响应示例：

```
{
        "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 命令字符串示例

使用 GET 方法获取已完成作业部分的相关信息：

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

使用 SQL 监控作业部分

获取已完成作业部分的相关信息：

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

使用 RabbitMQ 监控作业部分

outbound.JobSegment

5.4 作业部分修改

尚未完成的作业可通过更改部分目标和 / 或其优先级来进行修改。这些修改可在作业部分完成之前的任何时间点进行。

如果作业部分已经开始，则不会记录优先级变更，因为优先级已用于作业部分调度。

> **注：**如果在作业完成之后发出了作业部分修改请求，则会进行记录，但不会对作业产生影响。

作业部分修改详情

本节介绍了如何修改现有作业部分。此外，还描述了如何监控和删除已修改的作业部分。

下表描述了作业部分修改项模式。

表 5-14. 作业部分修改项模式

项目	说明	数据类型
namekey	JobSegmentModify 请求实体的唯一标识符。 对 POST/ 插入 / 发布方法来说为可选。如果省略，Integration Toolkit 会自动生成。	字符串
segment namekey	正在修改的作业部分的唯一标识符。 对 POST/ 插入 / 发布方法来说此项或 segmentId 为必需。	
segmentId	根据 segment namekey 修改待处理或正在处理的作业。 对 POST/ 插入 / 发布方法来说此项或 segment namekey 为必需。	
goal	已修改作业部分的目标名称。 对 POST/ 插入 / 发布方法来说为可选。	
priority	已修改作业部分的优先级。 对 POST/ 插入 / 发布方法来说为可选。	整型
status	来自队列管理器的“成功”或失败消息。	

使用 REST 的作业部分修改

使用下述 REST 调用修改并删除现有作业部分。这些调用还可用于获取排队等候 Integration Toolkit 执行的已修改作业部分信息。

表 5-15. 作业部分 REST 调用资源

方法	资源	功能
GET	/JobSegmentModify/ByKey/{namekey}	按 namekey 获取已修改作业部分。
DELETE	/JobSegmentModify/{namekey}	按 namekey 删除已修改作业部分。
GET	/JobSegmentModify/UpdatedSince?sinceTime={time millis}	获取自给定时间以来已更新的已修改作业部分实体列表。
POST	/JobSegmentModify	创建已修改作业部分。
GET	/JobSegmentModify/BySegmentId/{SegmentID}	获取按 SebmentId 筛选的已修改作业部分实体列表。

作业部分修改 JSON 模式

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

作业部分修改 JSON 示例

使用 POST 请求“https://[IP]:8443/JobSegmentModify”修改作业部分：

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

cURL 命令字符串示例

使用 POST 请求修改作业部分：

```
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\"}"
```

使用 SQL 的作业部分修改

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

使用 RabbitMQ 的作业部分修改

inbound.JobSegmentModify

outbound.JobSegmentModify

5.5 作业和作业部分取消

作业和作业部分取消通过提供方法和相关联值来标识待取消作业来实现。

作业和作业部分取消详情

本节介绍了取消作业和作业部分的详细信息。您可以监控并删除现有作业取消。

下表描述了作业和作业部分取消项模式。

表 5-16. 作业和作业部分取消模式

项目	详细信息	数据类型
namekey	作业取消请求实体的唯一标识符。 对 POST/ 插入 / 发布方法来说为可选。如果省略，Integration Toolkit 会自动生成。	字符串
cancelType（枚举项）	必须用下面的取消方法代替。 请参见下面的作业请求 JSON 示例。 对 POST/ 插入 / 发布方法来说为必需。	
cancelValue（枚举项）	必须替换为所选 cancelType 的值。 提供所选“cancelType”方法的值。	
cancelReason	作业或作业部分被取消的原因。 对 POST/ 插入 / 发布方法来说为可选。	
status	来自队列管理器的“成功”或失败消息。	

必须为 cancelType 项指定取消方法。支持下述 cancelType 方法。

表 5-17. 作业和作业部分取消方法描述

cancelType 方法	说明
jobId	根据作业 JobId 取消待处理或正在处理的作业。
jobNamekey	根据作业的 namekey 取消待处理或正在处理的作业。
segmentId	根据 segmentId 取消待处理或正在处理的作业。
segmentNamekey	根据 Segment namekey 取消待处理或正在处理的作业。
jobStatus	取消指定状态的所有作业。
robot	取消分配给指定 AMR 的所有作业。
removeSegmentId	取消使用指定 SegmentId 的作业部分。
removeSegmentNamekey	取消使用指定 segment namekey 的作业部分。

使用 REST 的作业和作业部分取消功能

使用下述 REST 调用取消作业和作业部分，并监控现有作业取消。

表 5-18. 作业和作业部分取消 REST 调用资源

方法	资源	功能
GET	/JobCancel/ByKey/{namekey}	按作业取消请求 namekey 获取作业取消请求。
DELETE	/JobCancel/{namekey}	按 namekey 删除作业或作业部分取消请求。
GET	/JobCancel/UpdatedSince?sinceTime={time millis}	获取自给定时间以来已更新的已取消作业实体列表。
POST	/JobCancel	创建取消作业

作业和作业部分取消 JSON 模式

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

作业和作业部分取消 JSON 示例

使用 POST 请求“https://[IP]:8443/JobCancel”取消作业：

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

cURL 命令字符串示例

使用 POST 请求取消作业：

```
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\"}"
```

使用 SQL 的作业取消

取消作业：

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

使用 RabbitMQ 取消作业

inbound.JobCancel

outbound.JobCancel

5.6 WaitTaskCancel

带 FLOW 2.0 的 Integration Toolkit 版本包含一个附加功能。它能够仅通过 REST 取消等待状态，并且能够检查等待状态的状态。

版本信息：该功能在 FLOW 2.0 Integration Toolkit 版本 1.1.0 中可用。

表 5-19.WaitTask 取消 REST 调用资源

方法	资源	功能
POST	/WaitTaskCancel	取消等待状态。
GET	/WaitTaskState/{robot}	获取机器人的等待状态。

cURL 示例

以下为机器人 Robot1 的 POST 和 GET cURL 示例：

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"
```

响应

这两个 REST 调用都会返回一个反映机器人等待状态的代码，如下表所示：

代码	说明	注
正值	“等待 <s> 秒”	
0	“已完成等待”	不太可能返回该值，因为是暂时状态
-1	“一直等待下去”	
-2	“等待任务中断”	不太可能返回该值，因为是暂时状态
-3	“等待由用户停止”	不太可能返回该值，因为是暂时状态
-4	“等待由该客户端中断”	
-5	“现在未处于等待状态”	

第 6 章： AMR 信息和故障

使用 Integration Toolkit 可监控 AMR 信息和故障。以下各节进行了详细介绍。

6.1 AMR 信息

本节介绍了如何使用 Integration Toolkit 监控 AMR 信息。

AMR 信息监控详情

本节介绍了有关监控各个 AMR 状态的详细信息。

附加信息：通过访问相关联的 DataStore 值，可获取详细的信息，如位置和电池状态。更多信息请参见第 23 页的“4.1 常见的 DataStore 用例”。

版本信息：从 FLOW 2.0（Integration Toolkit 版本 1.1.0）开始，Integration Toolkit 提供的 AMR 列表将只反映目前附加至 Fleet Manager 的 AMR，并在机器人被从车队调度移除或被添加至车队调度的 10 秒内移除或添加机器人。

下表描述了 AMR 信息监控项模式。

表 6-1.AMR 信息监控项模式

项目	说明	数据类型
namekey	AMR 的唯一标识符（与车队调度中提供的 AMR 名称相同）。	字符串
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	

使用 REST 监控 AMR 信息

使用下述 REST 调用监控 AMR 信息。

表 6-2.AMR 信息监控 REST 调用资源

方法	资源	功能
GET	/Robot/ByKey/{namekey}	获取按 AMR 名称（namekey）筛选的 AMR 状态。
GET	/Robot/History?sinceTime={time millis}	获取自给定时间以来已更新的 AMR 历史记录实体列表。
GET	/Robot/History?sinceTime={time millis}&namekey={namekey}	获取自给定时间以来已更新的指定 namekey 的 AMR 历史记录实体列表。
GET	/Robot/ByStatus/{Status}	获取按 Status 筛选的 AMR 实体列表。
GET	/Robot/BySubStatus/{SubStatus}	获取按 SubStatus 筛选的 AMR 实体列表。
GET	/Robot/UpdatedSince?sinceTime={time millis}	获取自给定时间以来已更新的 AMR 实体列表。

AMR 数据监控 JSON 模式

```
{
      "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 数据监控 JSON 示例

在发出 GET 请求“https://[IP]:8443/Robot/ByStatus/Available”之后的响应示例

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

cURL 命令字符串示例

获取可用 AMR 的相关信息：

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

使用 SQL 监控 AMR 数据

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

使用 RabbitMQ 监控 AMR 信息

outbound.Robot

6.2 AMR 故障

使用 Integration Toolkit 可监控 AMR 故障。

所有 AMR 故障都独立于 Integration Toolkit 产生。

AMR 故障监控

本节介绍了有关监控所有 AMR 故障的详细信息。这些功能可用于指示 AMR 何时出现了问题，并提供 AMR 故障恢复的相关信息。

AMR 故障监控详情

下表描述了 AMR 故障项模式。

表 6-3.AMR 故障项模式

项目	说明	数据类型
namekey	故障实体的唯一标识符。	字符串
upd	最后一次更新该记录的时间戳。	长整型
robot	处于故障状态的 AMR 名称。	字符串
active	当前故障状态。	布尔值
blockDriving	驱动模块故障。	
driving	驱动故障。	
critical	严重故障。	
clearedOnGo	故障清除，继续运行。	
clearedOnAck	已确认故障清除。	
application	应用程序故障。	

项目	说明	数据类型
name	故障名称	字符串
type	故障类型。	
shortDescription	故障的简要描述。	
longDescription	故障的详细描述。	

使用 REST 监控 AMR 故障

使用下述 REST 调用监控 AMR 故障。

表 6-4.AMR 故障监控 REST 调用资源

方法	资源	功能
GET	/RobotFault/ByKey/{namekey}	获取按 namekey 筛选的 AMR 故障。
GET	/RobotFault/UpdatedSince?sinceTime={time millis}	获取自给定时间以来已更新的 AMR 故障实体列表。
GET	/RobotFault/History?sinceTime={time millis}	获取自给定时间以来已更新的 AMR 故障历史记录实体列表。
GET	/RobotFault/History?sinceTime={time millis} &namekey={namekey}	获取自给定时间以来已更新的指定 namekey 的 AMR 故障历史记录实体列表。
GET	/RobotFault/ByRobot/{AMR}	获取按 AMR 筛选的 AMR 故障实体列表。
GET	/RobotFault/ByType/{Type}	获取按类型筛选的 AMR 故障实体列表。
GET	/RobotFault/ByName/{Name}	获取按名称筛选的 AMR 故障实体列表。
GET	/RobotFault/ByActive/{Value}	获取按活动状态筛选的 AMR 故障实体列表。

AMR 故障监控 JSON 模式

```
{
        "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 故障监控 JSON 示例

在发出 GET 请求“https://[IP]:8443/RobotActive/false”之后的响应示例：

```
{
        "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 命令字符串示例

使用 GET 方法获取带有活动故障的 AMR 相关信息：

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

使用 SQL 监控 AMR 故障

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

使用 RabbitMQ 监控 AMR 故障

outbound.RobotFault

附录

A.1 SQL 数据库模式

下表视图名称在 IntegrationDB 中可用。

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 调用

本节介绍了所有获支持 REST 调用的详细信息。

/DataStoreItem/ByCategory/{Category}

获取按 Category 筛选的 DataStoreItem 实体列表。

请求类型：GET

/DataStoreItem/ByDisplayName/{DisplayName}

获取按 DisplayName 筛选的 DataStoreItem 实体列表。

请求类型：GET

/DataStoreItem/ByGroupName/{GroupName}

获取按 GroupName 筛选的 DataStoreItem 实体列表。

请求类型：GET

/DataStoreItem/ByItemName/{ItemName}

获取按 ItemName 筛选的 DataStoreItem 实体列表。

请求类型：GET

/DataStoreItem/ByKey/{namekey}

按 namekey 获取 DataStoreItem。

请求类型：GET

/DataStoreItem/BySource/{Source}

获取按 Source（Source 就是 AMR 名称）筛选的 DataStoreItem 实体列表。

请求类型：GET

/DataStoreItem/ByType/{Type}

获取按类型筛选的 DataStoreItem 实体列表。

请求类型：GET

/DataStoreItem/UpdatedSince?sinceTime={time millis}

获取自给定时间以来已更新的 DataStoreItem 实体列表。

请求类型：GET

/DataStoreValue/ByKey/{namekey}

按 namekey 获取 DataStoreValue。

请求类型：GET

/DataStoreValue/UpdatedSince?sinceTime={time millis}

获取自给定时间以来已更新的 DataStoreValue 实体列表。

请求类型：GET

/DataStoreValueLatest/{DataStore item name}

针对指定名称的 DataStore 项，返回（不订阅）一个一次性值。

请求类型：GET

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

针对在指定名称的 AMR 上指定名称的 DataStore 项，返回（不订阅）一个一次性值。

请求类型： GET

/DataStoreValueLatest/{DataStore item name}:*

针对所有机器人上指定名称的 DataStore 项，返回（不订阅）一个一次性值。

请求类型：GET

/Dropoff

创建卸货作业。

请求类型：POST

/Dropoff/{namekey}

按 namekey 删除卸货作业。

请求类型：DELETE

/Dropoff/ByAssignedJobId/{AssignedJobId}

获取按 AssignedJobId 筛选的卸货作业实体列表。

请求类型：GET

/Dropoff/ByJobId/{JobId}

获取按 JobId 筛选的卸货作业实体列表。

请求类型：GET

/Dropoff/ByKey/{namekey}

获取按 namekey 筛选的卸货作业。

请求类型：GET

/Dropoff/ByRobot/{AMR}

获取按 AMR 筛选的卸货作业实体列表。

请求类型：GET

/Dropoff/ByStatus/{Status}

获取按 Status 筛选的卸货作业实体列表。

请求类型：GET

/Dropoff/UpdatedSince?sinceTime={time millis}

获取自给定时间以来已更新的卸货作业实体列表。

请求类型：GET

/Goal/ByKey/{namekey}

获取按 namekey 筛选的目标。

请求类型：GET

/Goal/UpdatedSince?sinceTime={time millis}

获取自给定时间以来已更新的目标实体列表。

请求类型：GET

/Job/ByJobId/{JobId}

获取按 JobId 筛选的作业实体列表。

请求类型：GET

/Job/ByKey/{namekey}

按 namekey 获取作业。

请求类型：GET

/Job/ByLastAssignedRobot/{LastAssignedRobot}

获取按 LastAssignedRobot 筛选的作业实体列表。

请求类型：GET

/Job/ByStatus/{Status}

获取按 Status 筛选的作业实体列表。

请求类型：GET

/Job/History?sinceTime={time millis}

获取自给定时间以来的作业历史记录实体列表。

请求类型：GET

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

获取自给定时间以来特定 namekey 的作业历史记录实体列表。

请求类型：GET

/Job/UpdatedSince?sinceTime={time millis}

获取自给定时间以来已更新的作业实体列表。

请求类型：GET

/JobCancel

创建 JobCancel。

请求类型：POST

/JobCancel/{namekey}

按 namekey 删除 JobCancel。

请求类型：DELETE

/JobCancel/ByKey/{namekey}

按 namekey 获取 JobCancel。

请求类型：GET

/JobCancel/UpdatedSince?sinceTime={time millis}

获取自给定时间以来已更新的 JobCancel 实体列表。

请求类型：GET

/JobRequest

创建 JobRequest。

请求类型：POST

/JobRequest/{namekey}

按 namekey 删除 JobRequest。

请求类型：DELETE

/JobRequest/ByAssignedJobId/{AssignedJobId}

获取按 AssignedJobId 筛选的 JobRequest 实体列表。

请求类型：GET

/JobRequest/ByJobId/{JobId}

获取按 JobId 筛选的 JobRequest 实体列表。

请求类型：GET

/JobRequest/ByKey/{namekey}

按 namekey 获取 JobRequest。

请求类型：GET

/JobRequest/ByStatus/{Status}

获取按 Status 筛选的 JobRequest 实体列表。

请求类型：GET

/JobRequestDetail/ByKey/{namekey}

按 namekey 获取 JobRequestDetail。

请求类型：GET

/JobRequestDetail/UpdatedSince?sinceTime={time millis}

获取自给定时间以来已更新的 JobRequestDetail 实体列表。

请求类型：GET

/JobRequest/UpdatedSince?sinceTime={time millis}

获取自给定时间以来已更新的 JobRequest 实体列表。

请求类型：GET

/JobSegment/ByGoalName/{GoalName}

获取按 GoalName 筛选的 JobSegment 实体列表。

请求类型：GET

/JobSegment/ByJob/{Job}

获取按 Jobnamekey 筛选的 JobSegment 实体列表。

请求类型：GET

/JobSegment/ByKey/{namekey}

获取按 namekey 筛选的 JobSegment 信息。

请求类型：GET

/JobSegment/ByRobot/{AMR}

获取按 AMR 筛选的 JobSegment 实体列表。

请求类型：GET

/JobSegment/ByStatus/{Status}

获取按 Status 筛选的 JobSegment 实体列表。

请求类型：GET

/JobSegment/History?sinceTime={time millis}

获取自给定时间以来已更新的作业部分历史记录实体列表。

请求类型：GET

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

获取自给定时间以来已更新的指定 namekey 的作业部分历史记录实体列表。

请求类型：GET

/JobSegment/UpdatedSince?sinceTime={time millis}

获取自给定时间以来已更新的作业部分实体列表。

请求类型：GET

/JobSegmentModify

创建 JobSegmentModify。

请求类型：POST

/JobSegmentModify/{namekey}

按 namekey 删除 JobSegmentModify。

请求类型：DELETE

/JobSegmentModify/ByKey/{namekey}

获取按 namekey 筛选的 JobSegmentModify。

请求类型：GET

/JobSegmentModify/BySegmentId/{SegmentId}

获取按 SegmentId 筛选的 JobSegmentModify 实体列表。

请求类型：GET

/JobSegmentModify/UpdatedSince?sinceTime={time millis}

获取自给定时间以来已更新的已修改作业部分实体列表。

请求类型：GET

/Pickup

创建拾取作业。

请求类型：POST

/Pickup/{namekey}

删除按 namekey 筛选的拾取作业。

请求类型：DELETE

/Pickup/ByAssignedJobId/{AssignedJobId}

获取按 AssignedJobId 筛选的拾取作业实体列表。

请求类型：GET

/Pickup/ByJobId/{JobId}

获取按 JobId 筛选的拾取作业实体列表。

请求类型：GET

/Pickup/ByKey/{namekey}

按 namekey 获取拾取作业信息。

请求类型：GET

/Pickup/ByStatus/{Status}

获取按 Status 筛选的拾取作业实体列表。

请求类型：GET

/Pickup/UpdatedSince?sinceTime={time millis}

获取自给定时间以来已更新的拾取作业实体列表。

请求类型：GET

/PickupDropoff

创建拾取卸货作业。

请求类型：POST

/PickupDropoff/{namekey}

按 namekey 删除拾取卸货作业信息。

请求类型：DELETE

/PickupDropoff/ByAssignedJobId/{AssignedJobId}

获取按 AssignedJobId 筛选的拾取卸货作业实体列表。

请求类型：GET

/PickupDropoff/ByJobId/{JobId}

获取按 JobId 筛选的拾取卸货作业实体列表。

请求类型：GET

/PickupDropoff/ByKey/{namekey}

按 namekey 获取拾取卸货作业信息。

请求类型：GET

/PickupDropoff/ByStatus/{Status}

获取按 Status 筛选的拾取卸货作业实体列表。

请求类型：GET

/PickupDropoff/UpdatedSince?sinceTime={time millis}

获取自给定时间以来已更新的拾取卸货作业实体列表。

请求类型：GET

/Robot/ByKey/{namekey}

获取按 namekey 筛选的 AMR 信息。

请求类型：GET

/Robot/ByStatus/{Status}

获取按 Status 筛选的 AMR 实体列表。

请求类型：GET

/Robot/BySubStatus/{SubStatus}

获取按 SubStatus 筛选的 AMR 实体列表。

请求类型：GET

/Robot/History?sinceTime={time millis}

获取自给定时间以来已更新的 AMR 历史记录实体列表。

请求类型：GET

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

获取自给定时间以来已更新的指定 namekey 的 AMR 历史记录实体列表。

请求类型：GET

/Robot/UpdatedSince?sinceTime={time millis}

获取自给定时间以来已更新的 AMR 实体列表。

请求类型：GET

/RobotFault/ByActive/{Active}

获取按 Active 筛选的 RobotFault 实体列表。

请求类型：GET

/RobotFault/ByKey/{namekey}

获取按 namekey 筛选的 RobotFault 信息。

请求类型：GET

/RobotFault/ByName/{Name}

获取按 Name 筛选的 RobotFault 实体列表。

请求类型：GET

/RobotFault/ByRobot/{AMR}

获取按 AMR 筛选的 RobotFault 实体列表。

请求类型：GET

/RobotFault/ByType/{Type}

获取按类型筛选的 RobotFault 实体列表。

请求类型：GET

/RobotFault/History?sinceTime={time millis}

获取自给定时间以来已更新的 AMR 故障历史记录实体列表。

请求类型：GET

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

获取自给定时间以来已更新的指定 namekey 的 AMR 故障历史记录实体列表。

请求类型：GET

/RobotFault/UpdatedSince?sinceTime={time millis}

获取自给定时间以来已更新的 AMR 故障实体列表。

请求类型：GET

/SubscriptionConfig

更新 SubscriptionConfig。

请求类型：PUT

/SubscriptionConfig/ByKey/{namekey}

按 namekey 获取 SubscriptionConfig。

请求类型：GET

/SubscriptionConfig/UpdatedSince?sinceTime={time millis}

获取自给定时间以来已更新的 SubscriptionConfig 实体列表。

请求类型：GET

/WaitTaskCancel

取消等待状态。

请求类型：POST

/WaitTaskState/{robot}

检查等待状态的状态。

请求类型：GET

A.3 RabbitMQ 队列

本节介绍了 RabbitMQ 通信通道的所有可用入站和出站队列的列表信息。

入站队列

inbound.Dropoff

inbound.JobCancel

inbound.JobRequest

inbound.JobSegmentModify

inbound.Pickup

inbound.PickupDropoff

inbound.SubscriptionConfig

出站队列

outbound.DataStoreValue

outbound.datastore.X（X 是订阅的 DataStoreValue）

outbound.Dropoff

outbound.Job

outbound.JobCancel

outbound.JobRequest

outbound.JobSegment

outbound.JobSegmentModify

outbound.Pickup

outbound.PickupDropoff

outbound.Robot

outbound.RobotFault

outbound.SubscriptionConfig

购买欧姆龙产品的客户须知

承诺事项

承蒙对欧姆龙株式会社（以下简称“本公司”）产品的一贯厚爱和支持，藉此机会再次深表谢意。
如果未特别约定，无论贵司从何处购买的产品，都将适用本承诺事项中记载的事项。
请在充分了解这些注意事项基础上订购。

1. 定义

本承诺事项中的术语定义如下。

(1)“本公司产品”：是指“本公司”的 FA 系统机器、通用控制器、传感器、电子 / 结构部件。

(2)“产品目录等”：是指与“本公司产品”有关的欧姆龙综合产品目录、FA 系统设备综合产品目录、安全组件综合产品目录、电子 / 机构部件综合产品目录以及其他产品目录、规格书、使用说明书、操作指南等，包括以电子数据方式提供的资料。

(3)“使用条件等”：是指在“产品目录等”资料中记载的“本公司产品”的使用条件、额定值、性能、运行环境、操作使用方法、使用时的注意事项、禁止事项以及其他事项。

(4)“客户用途”：是指客户使用“本公司产品”的方法，包括将“本公司产品”组装或运用到客户生产的部件、电子电路板、机器、设备或系统等产品中。

(5)“适用性等”：是指在“客户用途”中“本公司产品”的 (a) 适用性、(b) 动作、(c) 不侵害第三方知识产权、(d) 法规法令的遵守以及 (e) 满足各种规格标准。

2. 关于记载事项的注意事项

对“产品目录等”中的记载内容，请理解如下要点。

(1) 额定值及性能值是在单项试验中分别在各条件下获得的值，并不构成对各额定值及性能值的综合条件下获得值的承诺。

(2) 提供的参考数据仅作为参考，并非可在该范围内一直正常运行的保证。

(3) 应用示例仅作参考，不构成对“适用性等”的保证。

(4) 如果因技术改进等原因，“本公司”可能会停止“本公司产品”的生产或变更“本公司产品”的规格。

3. 使用时的注意事项

选用及使用本公司产品时请理解如下要点。

(1) 除了额定值、性能指标外，使用时还必须遵守“使用条件等”。

(2) 客户应事先确认“适用性等”，进而再判断是否选用“本公司产品”。“本公司”对“适用性等”不做任何保证。

(3) 对于“本公司产品”在客户的整个系统中的设计用途，客户应负责事先确认是否已进行了适当配电、安装等事项。

(4) 使用“本公司产品”时，客户必须采取如下措施：(i) 相对额定值及性能指标，必须在留有余量的前提下使用“本公司产品”，并采用冗余设计等安全设计 (ii) 所采用的安全设计必须确保即使“本公司产品”发生故障时也可将“客户用途”中的危险降到最小程度、(iii) 构建随时提示使用者危险的完整安全体系、(iv) 针对“本公司产品”及“客户用途”定期实施各项维护保养。

(5) 因 DDoS 攻击（分布式 DoS 攻击）、计算机病毒以及其他技术性有害程序、非法侵入，即使导致“本公司产品”、所安装软件、或者所有的计算机器材、计算机程序、网络、数据库受到感染，对于由此而引起的直接或间接损失、损害以及其他费用，“本公司”将不承担任何责任。
对于 (i) 杀毒保护、(ii) 数据输入输出、(iii) 丢失数据的恢复、(iv) 防止“本公司产品”或者所安装软件感染计算机病毒、(v) 防止对“本公司产品”的非法侵入，请客户自行负责采取充分措施。

(6)“本公司产品”是作为应用于一般工业产品的通用产品而设计生产的。除“本公司”已表明可用于特殊用途的，或已经与客户有特殊约定的情形外，若客户将“本公司产品”直接用于以下用途的，“本公司”无法作出保证。
(a) 必须具备很高安全性的用途（例：核能控制设备、燃烧设备、航空 / 宇宙设备、铁路设备、升降设备、娱乐设备、医疗设备、安全装置、其他可能危及生命及人身安全的用途）
(b) 必须具备很高可靠性的用途（例：燃气、自来水、电力等供应系统、24 小时连续运行系统、结算系统、以及其他处理权利、财产的用途等）
(c) 具有苛刻条件或严酷环境的用途（例：安装在室外的设备、会受到化学污染的设备、会受到电磁波影响的设备、会受到振动或冲击的设备等）
(d)“产品目录等”资料中未记载的条件或环境下的用途

(7) 除了不适用于上述 3.(6)(a) 至 (d) 中记载的用途外，“本产品目录等资料中记载的产品”也不适用于汽车（含二轮车，以下同）。请勿配置到汽车上使用。关于汽车配置用产品，请咨询本公司销售人员。

4. 保修条件

“本公司产品”的保修条件如下。

(1) 保修期限　自购买之日起 1 年。（但是，“产品目录等”资料中有明确说明时除外。）

(2) 保修内容　对于发生故障的“本公司产品”，由“本公司”判断并可选择以下其中之一方式进行保修。
(a) 在本公司的维修保养服务点对发生故障的“本公司产品”进行免费修理（但是对于电子、结构部件不提供修理服务。）
(b) 对发生故障的“本公司产品”免费提供同等数量的替代品

(3) 当故障因以下任何一种情形引起时，不属于保修的范围。
(a) 将“本公司产品”用于原本设计用途以外的用途
(b) 超过“使用条件等”范围的使用
(c) 违反本注意事项“3. 使用时的注意事项”的使用
(d) 非因“本公司”进行的改装、修理导致故障时
(e) 非因“本公司”出品的软件导致故障时
(f)“本公司”生产时的科学、技术水平无法预见的原因
(g) 除上述情形外的其它原因，如“本公司”或“本公司产品”以外的原因（包括天灾等不可抗力）

5. 责任限制

本承诺事项中记载的保修是关于“本公司产品”的全部保证。对于因“本公司产品”而发生的其他损害，“本公司”及“本公司产品”的经销商不负任何责任。

6. 出口管理

客户若将“本公司产品”或技术资料出口或向境外提供时，请遵守中国及各国关于安全保障进出口管理方面的法律、法规。否则，“本公司”有权不予提供“本公司产品”或技术资料。

IC321GC-zh
202202

注：规格如有变更，恕不另行通知。请以最新产品说明书为准。

欧姆龙自动化（中国）有限公司
http://www.fa.omron.com.cn　　咨询热线：**400-820-4535**

Cat. No. I637-CN5-02