Slamware RESTful API Development Manual

content

1. Overview

Slamware firmware versions 4.0 and later provide a `RESTful API` that is easier to use and richer than `C++ SDK`

It is compatible with any client system and programming language.

The port of the service is `1448` . This article introduces the basic usage of the API . Please refer to the online document for the specific definition of the interface: https://docs.slamtec.com/#/ .

If the firmware version is above 4.6.0, you can debug the API online by entering `IP: 1448` in the browser. Example: `192.168.11.1:1448`

For example, the interface for connecting to the robot hotspot and obtaining the power status of the robot is as follows:

```
GET http://192.168.11.1:1448/api/core/system/v1/power/status
```

Return content is

```
{
"batteryPercentage": 90,
"dockingStatus": "on_dock",
"isCharging": true,
"isDCConnected": false,
"powerStage": "running",
"sleepMode": "awake"
}
```

2. Interface specifications

2.1 Naming method

API interface `endpoint` specification

Most interfaces are organized in the following structure:

```
/api/{plugin}/{feature}/{version}/{resource…}
```

plugin

- Core: Agent core framework and general services
- Platform: Plug-in for universal chassis, providing basic functions such as reporting equipment events and uploading logs

- multi_floor: A plugin that provides multi-floor map management and cross-floor mobility capabilities, while being compatible with single-floor maps
- Delivery: A plugin that provides delivery services, which can be applied to restaurants, hotels and other scenarios

feature

- Robot Function Category

version

- Version number

2.2 Type of Method

Currently used `GET, PUT, POST, DELETE` four types of methods

- **GET** fetch resources (security, idempotence)
- **PUT** Create, update resources (non-secure, idempotence)
- **POST** creates a resource or performs an action (non-secure, non-idempotence)
- **DELETE Delete** a resource (non-secure, idempotence)

2.3 API parameters

Query type

`The query` parameter is followed by a question mark with one or more pairs of key = value, as shown below to obtain the POI of E building the 2nd floor:

```
GET http://127.0.0.1:1448/api/multi-floor/map/v1/pois?floor=2F&building=E
```

Path type

`The path` parameter is placed directly in the path, such as: `DELETE /api/core/artifact/v1/lines/{usage}/{id}` ,

The following indicates deleting the virtual wall with id 199:

```
DELETE http://127.0.0.1:1448/api/core/artifact/v1/lines/walls/199
```

Request Body

That is, the `Content-Type` of the API request is `application/json` , such as:

```
Java
curl -X 'POST' \
'http://127.0.0.1:1448/api/core/motion/v1/actions' \
```

```
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
    "action_name":"slamtec.agent.actions.MoveToAction",
    "options":{
        "target":{
            "x":0,
            "y":0,
            "z":0
},
        "move_options":{
            "mode":0,
            "flags":[],
            "yaw":0,
            "acceptable_precision":0,
            "fail_retry_count":0
       }
    }
}'
```

2.4 Return status code

2xx: Success

`200 (OK)` indicates that any operation requested by the Client was successfully performed.

`204 (no content)` indicates that the server has successfully completed the request and there is no content to send in the response payload body.

4xx: Client error

`400 (Bad Request)` Generic Client error state, used when there are no other `4xx` error codes.

`404 (Not Found) The URI` resource requested by the `REST API` could not be found .

5xx: Server error

`500` Server Internal Error

2.5 Return value

When the interface returns a status code of 200, `Content-Type` has the following types:

- `application/json`

Most interfaces return data in this JSON format

- `application/octet-stream`

Binary stream, obtaining the return value of explore map and stcm map is a binary stream

- `text/plain`

The return value of some interfaces is a simple string.

2.6 How to view interfaces in API documentation

Taking *`creating a new motion behavior (/api/core/motion/v1/actions)`* ***interface as an example***

Request/Response Overview :

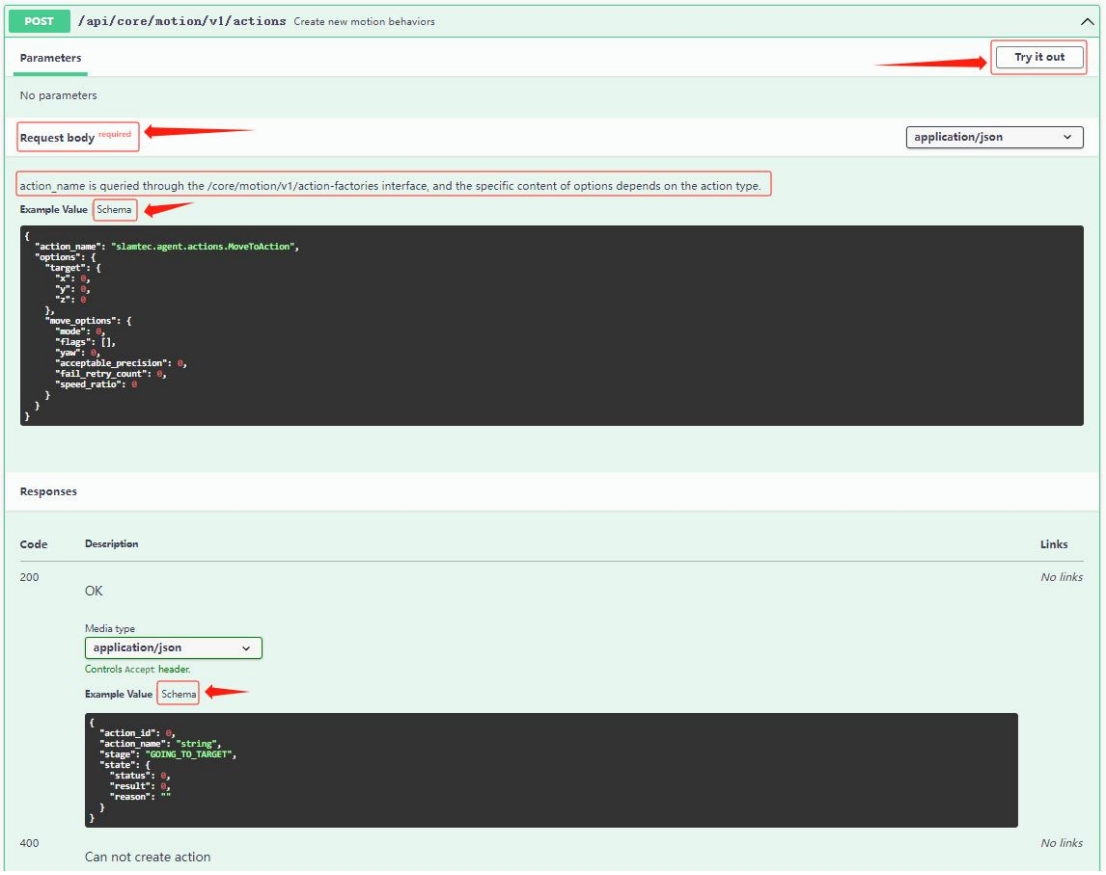
```
Java
POST http://127.0.0.1:1448/api/core/motion/v1/actions
Curl:
curl -X 'POST' \
'http://127.0.0.1:1448/api/core/motion/v1/actions' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
    "action_name":"slamtec.agent.actions.MoveToAction",
    "options":{
        "target":{
            "x":0.1,
            "y":0.2
        },
        "move_options":{
            "mode":0,
            "flags":[
                "with_yaw"
             ],
            "yaw":1
        }
    }
}'

Response Body
```

```
{
    "action_id":0,
    "action_name":"string",
    "stage":"GOING_TO_TARGET",
    "state":{
        "status":0,
        "result":0,
        "reason":""
    }
}
```

Interface `Request Body` and `Responses Body` Detail

In the figure below, there are three arrows from top to bottom on the left, and the first arrow has the word `required` . Any interface document with this word is a required field, the second and third arrows have the words `Schema` and `Example Value` , the default displays `Example Value` details under the word, that is, the Json structure of the `Request Body` or `Responses Body` , the word `Schema` can be

Click on it and click on `Schema` to see the generation rules of the Json structure of the `Example Value` .



How to view `schemas` in interface documentation

`Schema` may appear in both `Request Body` and `Responses` . Here is an example of

`Request Body` .

The viewing methods of the two are similar.

The following image shows the details of the `schema` of the `Request body` after expansion.

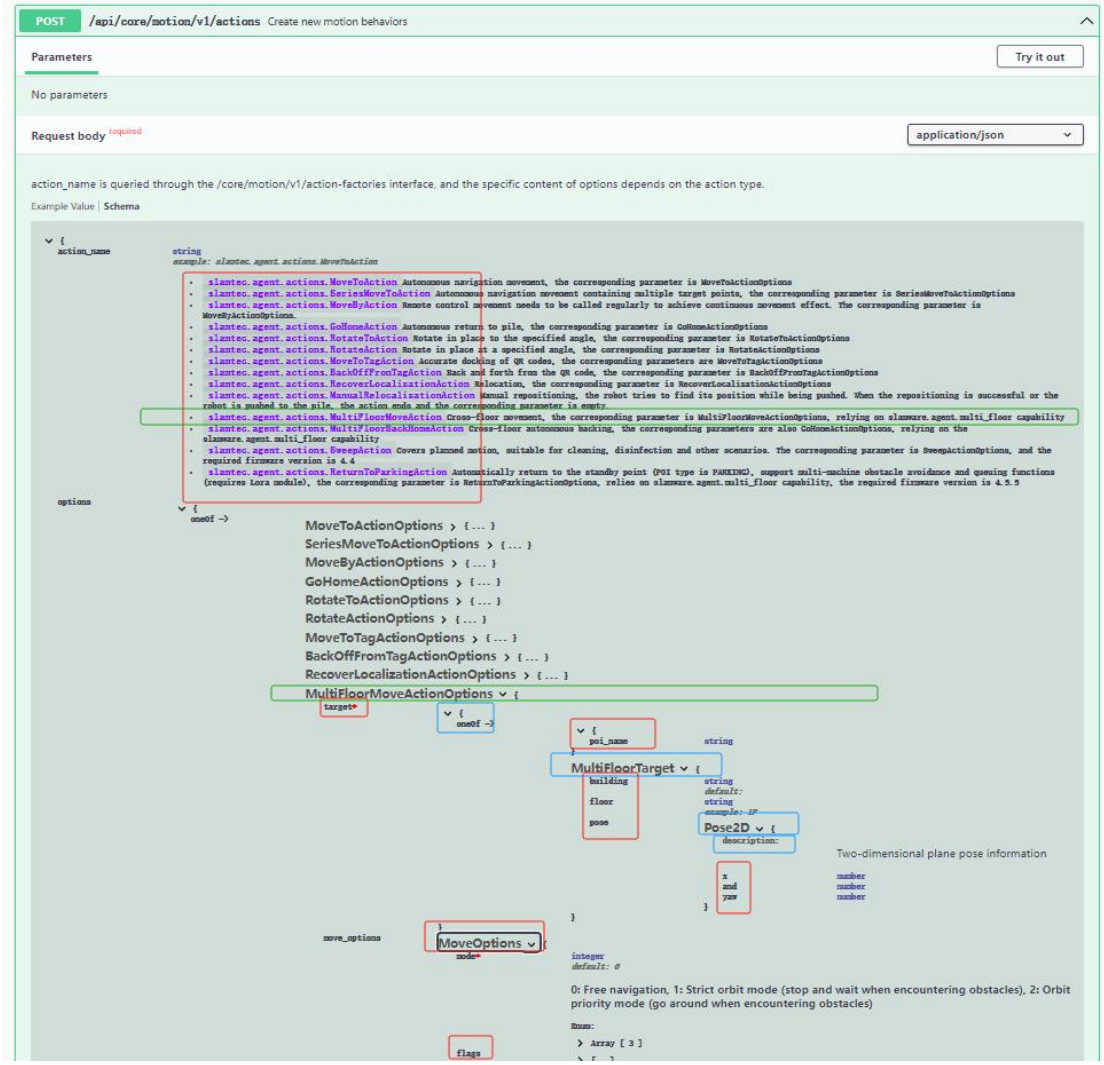
Parameters with red `required` and `*` are required

The parameters in the red box are the field `key` of the Json structure, and the words in the blue box are only auxiliary explanatory text. If there are ellipses in curly brackets item, you can click to view the details.

A pair of green boxes in the figure correspond to the value: such as `action_name` value is a pair of green boxes in the figure corresponding to the value: such as

`action_name` value is: `slamtec.agent.actions.MultiFloorMoveAction` , then the `option` in

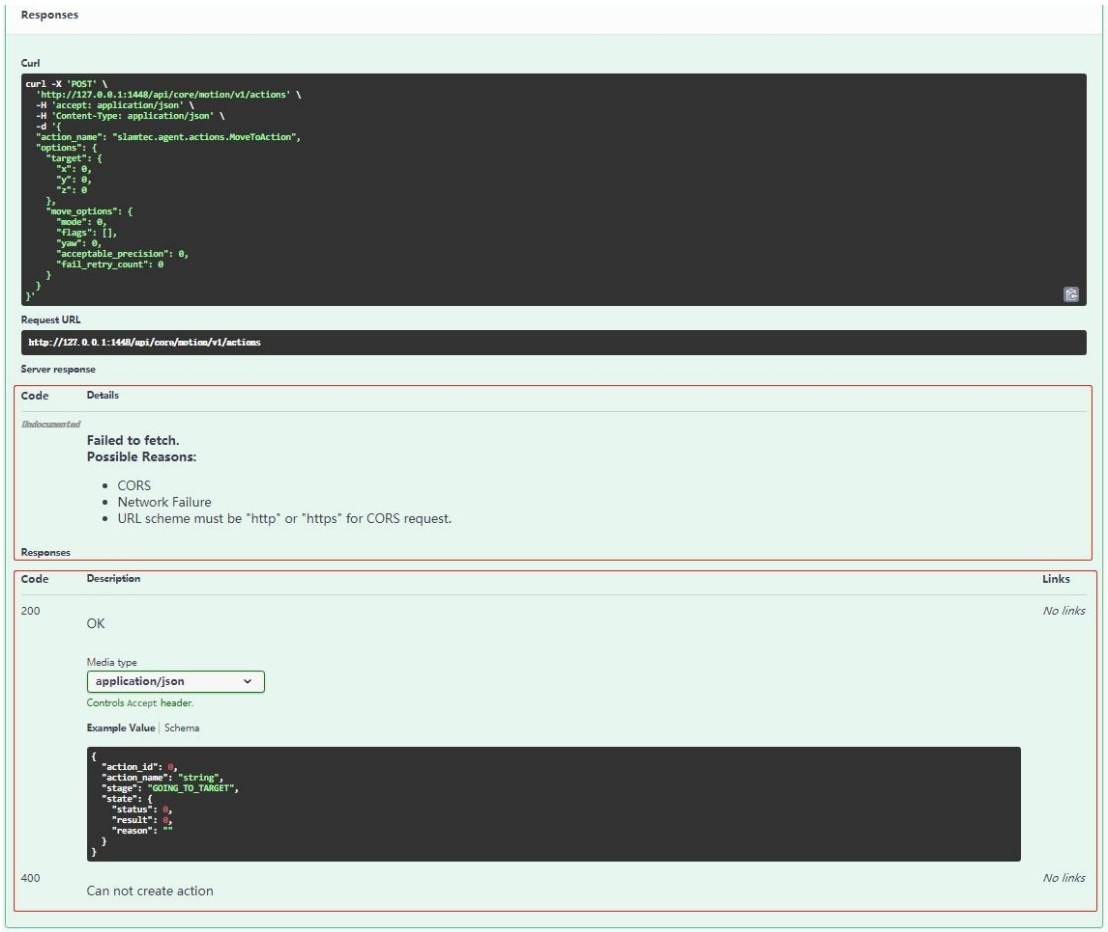
`oneOf` takes the corresponding `MultiFloorMoveActionOptions` structure

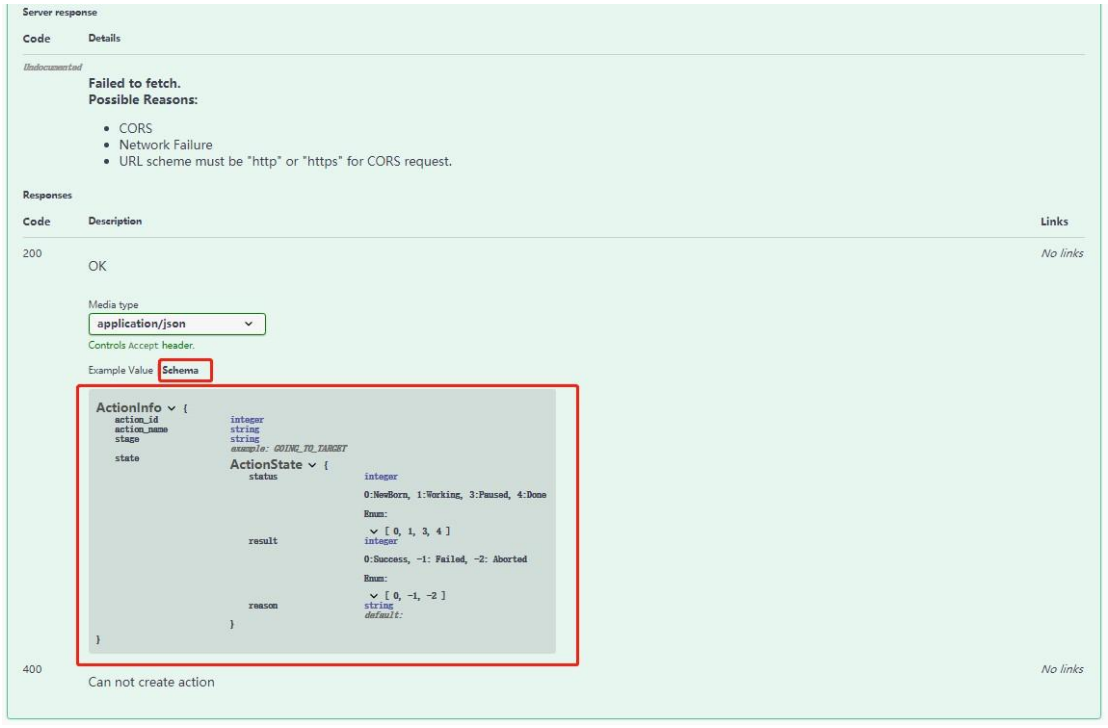


Response Body ***Details***

Response Body shows the `Response Body` that returns 200, and the parsing rule

`Shcema` corresponding to the `Body` or returns other status codes and their descriptions





3. Interface classification

Classification of features in Section 2.1

3.1 system resources

This type of interface can access the system-level resources of the robot, such as reading the power status, restarting the machine, setting system parameters, etc.

3.2 Functions related to slam positioning and mapping

Obtain robot pose, obtain/register charging station, turn on/off mapping, obtain map data, etc.

After completing the mapping and adding the required POIs , you can call the `Get Composite Map` interface to export the map:

```
GET /api/core/slam/v1/maps/stcm
```

3.3 Artifacts manually mark map elements

The following elements can be added to the map

- Virtual tracks(tracks): The robot can travel along a preset track through parameter control.
- Virtual walls(walls): Prohibit robots from entering certain areas.
- Forbidden Area(forbidden_area) : The effect is similar to a virtual wall, supporting automatic escape function, the robot will not enter the restricted area when it is outside, once it is pushed in, it can escape the restricted area to the nearest edge.We recommend use Forbidden area insteady of Virtual walls.
- Elevator area(elevator_area): When applied to a multi-floor environment, you need to add elevator information and merge maps of multiple floors into one file through RS.
- Dangerous area(dangerous_area): there are slopes, narrow roads, etc., which can limit the maximum movement speed of the robot.
- Coverage area (coverage_area): The robot plans a path to cover the entire area, behaving like a sweeper.
- maintenance area (maintenance_area): When the robot reopens the map creation, it will only update the map in the maintenance area.

3.4 motion robot operation control

This type of interface provides: obtaining all motion behaviors supported by the robot; obtaining/terminating/creating new motion behaviors for the robot; querying paths and target points; obtaining/setting motion strategies; enabling/querying manual relocation and other behavior controls.

- Robots need to create new motion behaviors to start moving.

Create a new motion behavior: POST `/api/core/motion/v1/actions`

- During the robot's movement, it is necessary to constantly query the Action status to determine the current movement status.

Query Action status: GET `/api/core/motion/v1/actions/{action_id}`

3.5 firmware upgrade

This type of interface provides functions for upgrading robot firmware and querying related upgrade information.

3.6 statistics of operation data

This type of interface mainly obtains the motion mileage and running time of the robot.

3.7 Android application management (ARM platform only)

This type of interface provides functions for robot installation/uninstallation of apps, as well as obtaining installed apps.

3.8 platform universal chassis and platform-related functions

This type of interface provides functions for obtaining robot system timestamps and obtaining robot event information.

- During the movement of the robot, it may encounter a series of situations such as encountering obstacles, low battery, etc., so the caller needs to constantly obtain event information to grasp the robot situation in real time and `obtain event information: GET /api/platform/v1/events` .

3.9 Multi-floor map management, cross-floor movement

Multi-floor map management, elevator and other functions: such as finding the nearest charging station to the robot, persistently saving the current map, reloading the map, synchronizing the map, etc.

3.10 Delivery service related interface

Note: The delivery-related interface is only available for the whole Robot, and the universal chassis is not supported by default. If you need to use it, please contact FAE.

As a whole, it is divided into three categories: system configuration, cargo management, and task management

This type of interface provides well-integrated functions for robot scenarios such as delivery, guidance, and greeting. If there are similar needs, the deliver service interface can also be used directly. It mainly includes functions such as creating tasks, querying tasks, canceling tasks, pausing/continuing tasks, ending tasks, starting to pick up items, ending to pick up items, completing operations, etc

Two types of map operation instructions

- */api/core/slam map operation interface:*

If you set a map, it will be set to the navigation system memory, and the map will not be persistently saved. You can export the map file.

- */api/multi-floor/map map operation interface:*

Maps can be uploaded or retrieved from the memory of the positioning and navigation system and persisted to disk.

Two types of POI operation instructions

- */api/core/artifact class POI operation interface:*

Poi can be added, deleted, modified, and checked.

- */api/multi-floor/map class POI operation interface:*

Only poi information can be found, where/api/multi-floor/map/v1/pois can get additional building, floor, poi_name, type information.

4.Example of deployment process

This stage is to complete the robot initialization operation, enable the robot, and make it ready for use.

Mainly includes: turning on/off map creation, adding pois, adding restricted areas, exporting maps, and other operations.

Note：If you use the Robostudio deployment method, you can ignore this process.

4.1 Mapping

Turn on/off mapping:

PUT /api/core/slam/v1/mapping/:enable

If enable is set to false in the request body, it will close drawing construction.

The return value `true` indicates that the operation was successful

4.2 Add `POI`

Add POI :

POST /api/core/artifact/v1/pois

The caller should randomly generate a UUID as id, the display_name in metadata is used for interface display, and type is used to distinguish POI types.

When adding POI during the mapping process, it is recommended not to include Pose. At this time, the POI will be created with the current position of the robot, and the sensor observation information will be recorded. Pose adjustment will be performed after the loop is closed.

Please click on `Schema` in the interface documentation for detailed instructions

4.3 Add a virtual wall

Add virtual line segment

POST /api/core/artifact/v1/lines/{usage}

ID is an invalid field when added and can be any value.

Please click on `Schema` in the interface documentation for detailed instructions

4.4 Set Forbidden areas

Add rectangular area

POST /api/core/artifact/v1/rectangle-areas/{usage}

*Different types of rectangular areas require different `metadata`, **please click on the interface document `Schema` to view detailed instructions***

4.5 Export map

Obtain composite map:

```
GET /api/core/slam/v1/maps/stcm
```

Composite map containing all data

The response message is binary ByteFlow and can be directly saved as a stcm file.

4.6 Save the map

1. If you want to save the map exported in 4.5 to the robot through API , please follow the following interface calling sequence to save the map.

a.Upload the map to the robot

Uploaded maps will be persistently saved in the file system, but will not be loaded into Slamware. [Note] When the robot is managed by the cloud, maps downloaded from the cloud will overwrite local maps.

```
POST /api/multi-floor/map/v1/stcm
```

b.Reload the map

```
POST /api/multi-floor/map/v1/stcm/:reload
```

c.Persistently save the current map

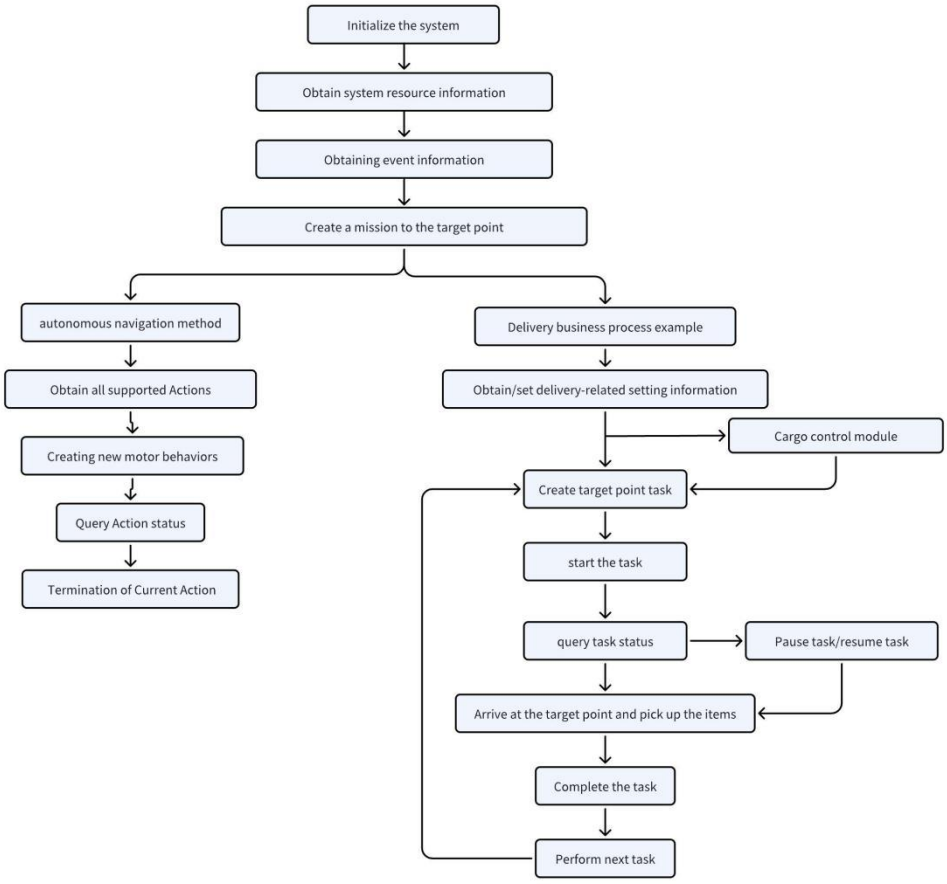
```
POST /api/multi-floor/map/v1/stcm/:save
```

2. After loading the map into the slamware through RoboStudio, the caller only needs to call the synchronous map interface to save the map

a. **Synchronized map**

```
POST /api/multi-floor/map/v1/stcm/:sync
```

5 . Examples of business processes



5.1 Initialize the system

When the robot starts, use the following polling interface to determine whether the system has completed initialization.

Only when each component is enabled ， robot can enter normal business logic.

```
GET /api/core/system/v1/capabilities
```

5.2 Obtain system resource information

By using the API , you can read and set up system-level resources of the robot, which can control and manage the robot more efficiently. Through the API, you can obtain key data such as device information, device health status, power status, etc., to ensure the normal operation and reasonable use of the robot. In addition, you can also shut down or restart the robot, and obtain and modify system parameters to meet specific usage requirements.

Get the robot power status:

```
GET /api/core/system/v1/power/status
```

Obtain device information:

```
GET /api/core/system/v1/robot/info
```

Obtain device health status information:

```
GET /api/core/system/v1/robot/health
```

Obtain system parameters:

```
GET /api/core/system/v1/parameter
```

Set system parameters:

```
PUT /api/core/system/v1/parameter
```

Turn off or restart the robot:

```
POST /api/core/system/v1/power/:shutdown
```

5.3 Obtain POI information

Get the list of POI information set in the map, that is, the target points that the robot can go to.

```
GET  /api/multi-floor/map/v1/pois
```

Through parameters, you can get the POI list of specified floors, buildings, POI types, and POI groups. Get all POIs without parameters.

Parameter name	Type	Required	Description
floor	string	No	Floor name
building	string	No	Building name
type	string	No	POI type
group	string	No	Poi grouping

5.4 Obtaining event information

When the robot is turned on, it obtains the current status, charging status, operating status, possible obstacles, elevator entry and exit, health status alarm and other situations of the robot. It is recommended that the caller polling this interface to know

the situation encountered by the robot in real time.

Get event information:

GET /api/platform/v1/events

The robot notifies the calling of its own events, and the upper computer can broadcast voice or engage in other interactions. Enabling different plugins will expand different event types.
Robot event information, type will be expanded with new definitions in different scenarios, and the caller only needs to handle the events they care about. GeneralEventType is a general event, ElevatorEventType is an event related to entering and leaving elevators, and DeliveryEventType is a delivery-related event.

Please click on** Schema **in the interface documentation for detailed instructions

5.5 The user starts operating the robot

When the user operates the robot, first call the following interface `enable_task_execution` set to false to prohibit the robot from moving, when the user completes the operation, the `enable_task_execution` is set to true to allow the robot to move, at this time the robot has a task to perform the task, no task is back to the charging station or return to the type of PARKING POI .

PUT /api/delivery/v1/tasks/:task_execution

The request body example is as follows

```
Java
{
  "enable_task_execution": false
}
```

5.6 Autonomous Navigation Process Example

This example shows an example where the caller creates an action to control the robot to move to a specified POI.

5.6.1 Obtain all supported Actions

GET /api/core/motion/v1/action-factories

- `Slamtec.agent.actions. MoveToAction` Autonomous navigation move
- `Slamtec.agent.actions.MultiFloorMoveAction` autonomous navigation movement, support cross-floor, POI target points, multi-level scheduling
- `Slamtec.agent.actions.MultiFloorBackHomeAction` Cross-floor

autonomous return charging station

- `Slamtec.agent.actions. SeriesMoveToAction` contains autonomous navigation moves with multiple target points
- `Slamtec.agent.actions. MoveByAction` remote movement, need to be called regularly to achieve continuous motion effect
- `Slamtec.agent.actions. GoHomeAction` autonomous return charging station
- `Slamtec.agent.actions. RotateToAction` Rotate in place to a specified angle
- `Slamtec.agent.actions. RotateAction`Rotate in place to a specified angle
- `Slamtec.agent.actions. MoveToTagAction` QR code precise docking
- `Slamtec.agent.actions.BackOffFromTagAction` Back from QR code to prevent collision.
- `Slamtec.agent.actions. RecoverLocalizationAction` relocation
- `Slamtec.agent.actions.ManualRelocalizationAction` Manual relocation
- `Slamtec.agent.actions. SweepAction` covers the planning movement, suitable for cleaning, disinfection and other scenarios, the required firmware version is 4.4
- `Slamtec.agent.actions.ReturnToParkingAction` Autonomous return to the standby point (POI type is PARKING), supports multi-machine obstacle avoidance and queuing function (requires Lora module), the required firmware version is 4.5.5

5.6.2 Creating new motor behaviors

POST /api/core/motion/v1/actions

action_name select `slamtec.agent.actions.MultiFloorMoveAction` , different motion behaviors can be selected by the action type obtained in the 5.6.1

The request body example is as follows, indicating that it moves to A101 in a way that is accurate to the point and accurate to the angle (yaw value is radians 1.0).

```Java
{
    "action_name":"slamtec.agent.actions.MultiFloorMoveAction",
    "options":{
        "target":{
            "poi_name":"A101"
        },
        "move_options":{
```

```
            "mode":2,
            "flags":[
                "with_yaw",
                "precise"
            ],
            "yaw":1,
            "acceptable_precision":0,
            "fail_retry_count":0
        }
    }
}
```

1. Mode: default is 0
    - `0` : Free navigation
    - `1` : Force following track mode (stop and wait in case of obstacles)
    - `2` : Track priority mode (orbiting when encountering obstacles)
2. flags:
    - `Precise` Precise to the point mode, making the robot more accurate to the point
    - `with_yaw` Precise to the angular mode, the value of the yaw field will only take effect if the flag is included
    - `fail_retry_count` Specify the number of retries after a search failure, apply default configuration if not specified
    - `find_path_ignoring_dynamic_obstacles` Ignore dynamic obstacles when searching for paths, suitable for crowded and narrow areas
3. yaw：

    The orientation of the robot after reaching the target point, accurate to the angle
4. acceptable_precision：

    The acceptable range to the taget point. When the target point is occupied, the distance between the robot and the target point is considered successful within this range. The default value is 0.1 meters or 0.18 meters, which does not affect the robot's accuracy to the point.
5. fail_retry_count：

    Number of failed retries

5.6.3 Query Action Status

When creating an action, a action_id will be returned, and the current status of the action will be queried based on this id. During the operation of the robot, it is necessary to view the Action status in real time through polling this interface.

GET /api/core/motion/v1/actions/{action_id}

5.6.4 Termination of Current Action

DELETE /api/core/motion/v1/actions/:current

5.7 Delivery business process example (delivey delivery service required)

Note: The delivery-related interface is only available for the whole robot, and the universal chassis is not supported by default. If you need to use it, please contact FAE.

The interface `endpoint` prefix is `/api/delivery`

As a whole, it is divided into three categories: system configuration, cargo management, and task management

5.7.1 user starts operating the robot

When the user operates the robot, first call the following interface `enable_task_execution` set to false to prohibit the robot from moving, when the user completes the operation, the `enable_task_execution` is set to true to allow the robot to move, at this time the robot has a task to perform the task, no task is back to the pile or return to the type of PARKING POI .

PUT /api/delivery/v1/tasks/:task_execution

The request body example is as follows

```Java
{
  "enable_task_execution": false
}
```

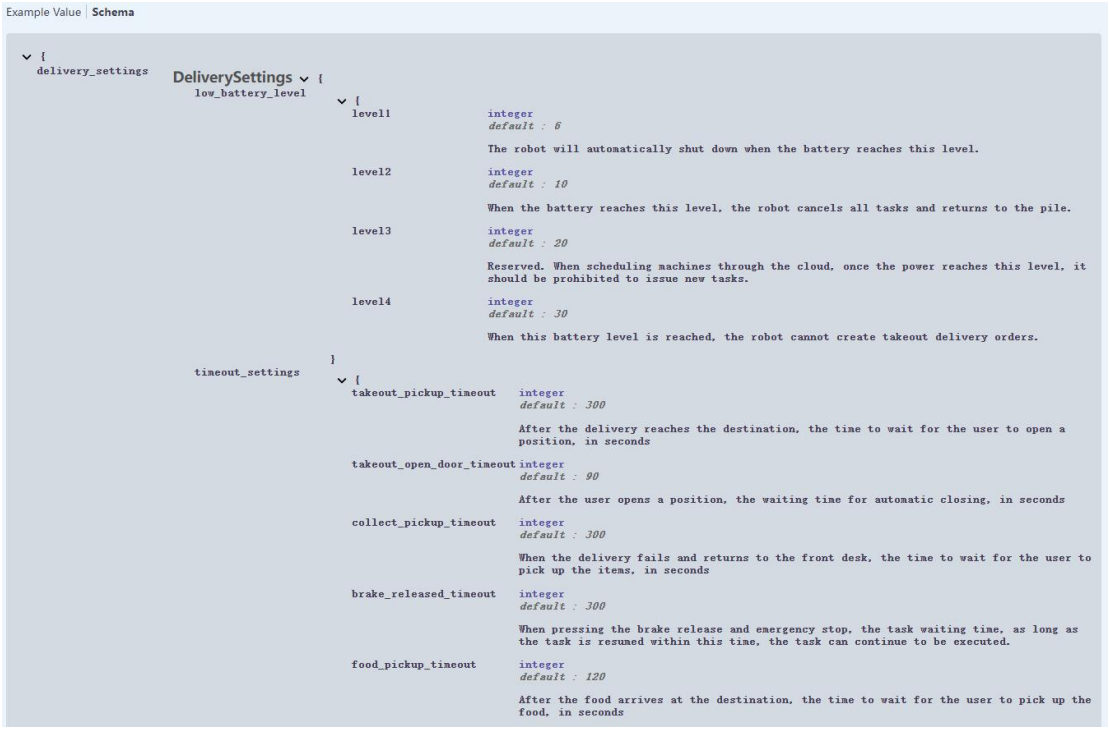
5.7.2 Get settings related to shipping

Users can obtain the low battery scenario configuration and timeout scenario configuration of the robot through the APP.

GET /api/delivery/v1/settings

Please click on `Schema` in the interface documentation for detailed instructions

Response Body



5.7.3 Set the timeout for the task

PUT /api/delivery/v1/settings/timeout

The request body example is as follows, food_pickup_timeout represents the waiting time after reaching the target point, in seconds.

```
Java
{
  "food_pickup_timeout": 0
}
```

5.7.4 operation Cargo

If it is a H2 hotel delivery robot, the APP opens/closes the cabin door through the cargo interface. If there is no cargo, please ignore this operation.

PUT /api/delivery/v1/cargos/{cargo_id}/boxes/{box_id}/{op}

5.7.5 create tasks

After the user puts in the item and closes the cabin door, call the following interface to create a task.

POST /api/delivery/v1/tasks

Currently supported Task types are `TAKEOUT (delivery), GUIDE (Guide), FOOD_DELIVERY (delivery), RECYCLE (return Disc), RETURN (return), DISINFECT (disinfection)`

If you want to create multiple tasks at once, call the following interface

POST /api/delivery/v1/tasks/:batch

5.7.6 start the mission

When the user completes the operation, the `enable_task_execution` is set to true to allow the robot to move. At this time, the robot has a task to perform the task, and if there is no task, it will return to the charging station or return to the POI of type PARKING.

PUT /api/delivery/v1/tasks/:task_execution

The request body example is as follows

```Java
{
  "enable_task_execution": true
}
```

5.7.7 Suspend/Resume Task

PUT /api/delivery/v1/tasks/:task_execution

Parameter name	Type	Required	Description
enable_task_execution	boolean	Yes	True: Continue the task False: pause task

The request body example is as follows

```Java
{
  "enable_task_execution": false
```

```
}
```

5.7.8 query task status

During the robot's task execution process, the APP needs to regularly query the task status to switch its own interface

```
GET /api/delivery/v1/stage
```

- `DEVICE_ERROR` Equipment failure, the chassis reported an Error message, the robot can not move, the caller should display the fault page.
- `GOING_TO_TASK_POINT` On the way to the task point, some tasks (such as return, delivery) need to stop at some task points halfway, and then go to the target point after completing the operation.
- `ARRIVED_AT_TASK_POINT` When the robot reaches the task point, it waits for the operation to complete or times out before continuing to the next stage.
- `ON_DELIVERING` Going to the target point, in order to be compatible with the name, it is not necessarily a delivery task.
- `ARRIVED_AT_TARGET` Reach the final target point.
- `ON_RETURNING` Returning, when the robot has a default docking point, this status indicates that the robot is going to that docking point.
- `GOING_HOME` is returning to the charging station.
- `IDLE` idle, the state the robot is in when it is at the default docking point or charging station

In addition, the polling event interface is needed to query some emergencies, such as path being blocked, receiving new tasks, etc., which can refer **to 5.4 to obtain event information**

```
GET /api/platform/v1/events
```

5.7.9 start taking items

When the APP queries that the `Stage` is `ARRIVED_AT_TARGET` , the pick up interface should be displayed.

If the robot has Cargo, the "open cabin" button needs to be displayed. After the user clicks it, the following interface is called

PUT /api/delivery/v1/tasks/:start_pickup

Then call the cargo interface to open the cabin door

5.7.10 complete the retrieval

After the user completes the pick up action, call the interface to notify the robot

PUT /api/delivery/v1/tasks/:end_pickup

5.7.11 the next mission

The APP calls the following interfaces to allow the robot to move autonomously, if there is a next task and continue the next task , or return to the charging station or standby point (POI type is PARKING) if there is no one

PUT /api/delivery/v1/tasks/:task_execution

5.8 Automatic recharge

POST /api/core/motion/v1/actions

action_name select `slamtec.agent.actions. MultiFloorBackHomeAction`

The request body example is as follows.

```
Java
{

"action_name":"slamtec.agent.actions.MultiFloorBackHomeAction",
    "options":{}
}
```

5.9 Robot abnormal recovery

Set the robot pose to the specified POI , which is generally used for recovery operations after an abnormality occurs, such as *restoring robot positioning information*

Note: This interface is usually called after positioning is lost. If the robot is pushed

back to the charging station, there is no need to call this interface, the system will automatically perform recovery actions.

```
PUT /api/multi-floor/localization/v1/pose
```

5.10 Motion Strategy

The motion strategy is a series of internal parameters of Slamware, involving various aspects such as motion speed and obstacle avoidance behavior. Different strategies can be applied to different scenarios. Generally, the default strategy can be used. The minimum firmware version required 4.2.4

Get all the motion strategy can be supported:

```
GET /api/core/motion/v1/strategies
```

Get the current motion strategy:

```
GET /api/core/motion/v1/strategies/:current
```

Set motion strategy:

```
PUT /api/core/motion/v1/strategies/:current
```

5.11 Frequently Asked Questions about Equipment Health Status

Description	Trigger condition	Trigger rear chassis action	Level	BigInt	Display information	Remove method
Emergency stop switch	Emergency stop switch trigger	Emergency stop, speed no longer responds	ERROR	0x02010100	"system emergency stop"	Release emergency stop switch recovery
Low	Batter	No	WA	0x010	"power low"	Battery

battery alarm	y less than 15%		RNING	20100		higher than 15%
Low battery alarm	Battery less than 5%	Automatic shutdown, all power off			"power low"	Battery higher than 5%
Brake release	Brake release button triggered	The motor loosens the shaft, and the speed no longer responds	ERROR	0x02010700	"motor brake released"	Brake release button recovery
Drive motor alarm	Drive motor driver alarm	Depending on the driver firmware, the shaft will be loose when overcurrent alarms occur.	WARNING	0x01030100	"motor alarm"	The chassis firmware will try to clear the drive alarm, and then check whether there is an alarm again. If there is no alarm again, it will automatically clear the health
Motor odomete	When the motor is	No	WARNING	0x01030010x	"motor[y]odometry alarm"	There is a speed , or brake release

r alarm	stopped (no speed, or speed is 0), the motor moves more than 500mm					trigger
platform watchdog trigger	Platform firmware watchdog triggered, firmware restarted	After firmware restart, the speed no longer responds	ERROR	0x02030400	"watchdog overflow"	Manual remove
Magnetic sensor trigger	Magnetic sensor trigger	stop immediately, speed no longer responds	ERROR	0x0204050x	"magtape[x] triggered"	Emergency stop switch trigger, or manually clear
Magnetic sensor communication	Magnetic sensor communication error	stop immediately, speed no longer responds	FATAL	0x0404060x	"magnetic[x]:y."	Check whether the connection cable is reliably connected, check whether the sensor is

error						damaged, manually clear the error, and restart the chassis if necessary
Tof cliff communication error	Tof cliff communication error	stop immediately, speed no longer responds	FATAL	0x0404020x	"cliff[x]:y."	Check whether the connection cable is reliably connected, check whether the sensor is damaged, manually clear the error, and restart the chassis if necessary
Collision sensor error	The collision continues to trigger and walks forward 200mm	stop immediately, speed no longer responds	ERROR	0x0204010x	"bumper sensor error"	Collision signal remove
TOF Cliff signal error	TOF cliff continuously triggers and walks	stop immediately, speed no longer respon	ERROR	0x0204020x	"cliff sensor error"	TOF cliff signal remove

	forward 200mm	ds				
Low positioning error			ERROR	0x02010900	"Low localization due to great environmental changes, because visual coarse poses received" Or "Low localization quality"	
Relocation error			ERROR	0x02010800	"Relocalization has failed last time, clear the error to move"	
Online slam reboot Location abnormally	Online slam restart	No action	ERROR	0x02010600	"slamware has rebooted, clear the error to move"	Pushing the robot back to the charging pile will reply (if it appears multiple times in the same area, please rebuild the map).

6. QR code precise docking

Please refer to the "QR code precise docking deployment manual" for the QR code precise docking deployment process.

This article only explains how to use API calls

6.1 Obtain deployed POI information

GET /api/core/artifact/v1/pois

Find the POI with type TAG, find the required POI according to the display_name, record the pose and tag_ids, and fill in the MoveToTagAction parameter. If tag_ids information is missing, please check the POI deployment process.

6.2 . Create precise docking motion behavior

POST /api/core/motion/v1/actions

action_name select `slamtec.agent.actions. MoveToTagAction`

The request body example is as follows

```Java
{
    "action_name": "slamtec.agent.actions.MoveToTagAction",
    "options": {
        "target": {
            "x": 0.590,
            "y": 0.110,
            "yaw": -3.130
        },
        "tag_ids": [0,50],
        "relative_pose_to_tag":{
            "x":0.4,
            "y":0.0
        }
    }
}
```

The target and tag_ids data recorded in 6.1.

relative_pose_to_tag field can be left blank, x represents the longitudinal distance from the center of the QR code, y represents the lateral deviation from the center of the QR code .

If not filled in, the QR code docking defaults to precise_move_to_tag safe_distance_to_tag (7.5cm) as the default value of x, and the default value of y is 0.

6.3 . After the operation is completed, call the backward action first to avoid collision

POST /api/core/motion/v1/actions

action_name select `slamtec.agent.actions. BackOffFromTagAction`

```
Java
{
    "action_name":"slamtec.agent.actions.BackOffFromTagAction",
    "options":{}
}
```