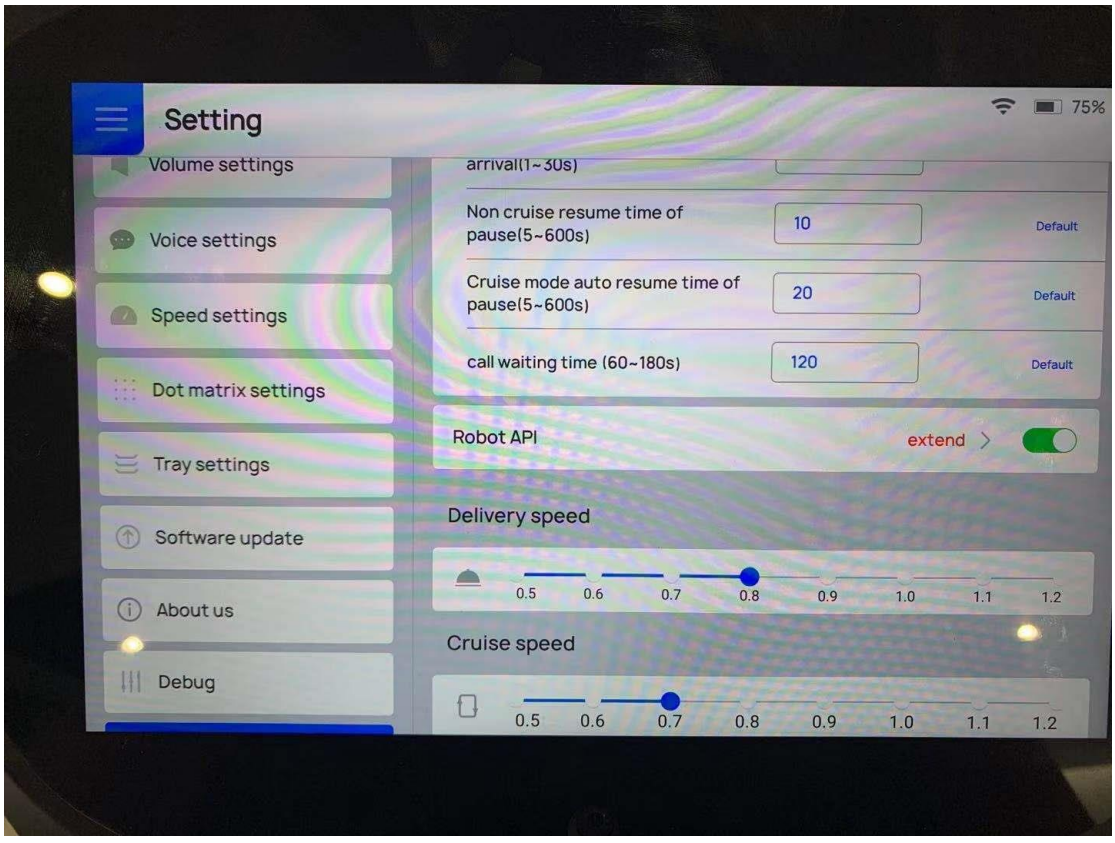
SDK Guidance Document for PuduTech Open Platform for Robotic Services (Nodejs Microservice)

The microservice is provided to developers for deployment which can give obtain the robot control authority by the http protocol.

To start using the Api function of the robot, first open the Advanced setting from robot:



Turn on the Robot API switch

Deployment

The project will output executable files under different platforms (currently support linux, macos, win) After running, it will generate a log (log folder) anddb.json (data file) in the current directory of the executable file.

Startup parameters

- -p Custom port number (default:9050))
- -outl Specify the folder path for log input (Do not specify the default and current directory of executing file)
- -outc Specify the file path of the configuration file output (Do not specify the default and current directory of the executing file)

Example：PuduRobotOpenServer.exe -p 9050

```
Windows file frame：

puduserver  (custom folder)
    -- pudulogs                    (log folder)
        -- *.log
  -- db.json                (data file)
  -- PuduRobotOpenServer.exe  (executable file)
```

Service framework

Developers can deploy microservices on their own servers and call the related-functions and the developer's front-end equipment will interacts with the developer's business server.

Add a device

Create a developer SDK device in PuduTech platform to get the Device ID and Secret, then add the SDK device to the microservice by the “add device” interface

and microservice will connect to Pudu Cloud by the Device ID and Secret. The SDK device ID and Secret can only used in one microservice.

To apply for the Device ID and secret: https://cs.pudutech.com/platform_list

Pudu Cloud account: Apply an account in Pudu Cloud and create your store, then add your current robot into your store. You will be able to check the robot data under the account on Pudu Cloud.

Robot group: All the robots in one store will assign to one group, the maps in all the robots must be the same in one group. Users can assign the robots to different groups in the store according to different scenarios.

Binding the device with Robot group: SDK device applied by developer in Pudu Cloud platform account, They can choose which store the device is bound to, after this process, the device will bound to the robot group under the store and they will be able to access the robots.

Function list: Beside binding the store, developers have to select the function of device, this selection will decide if the device can receive the relevant notification. Please help yourselves to select it for your demand.

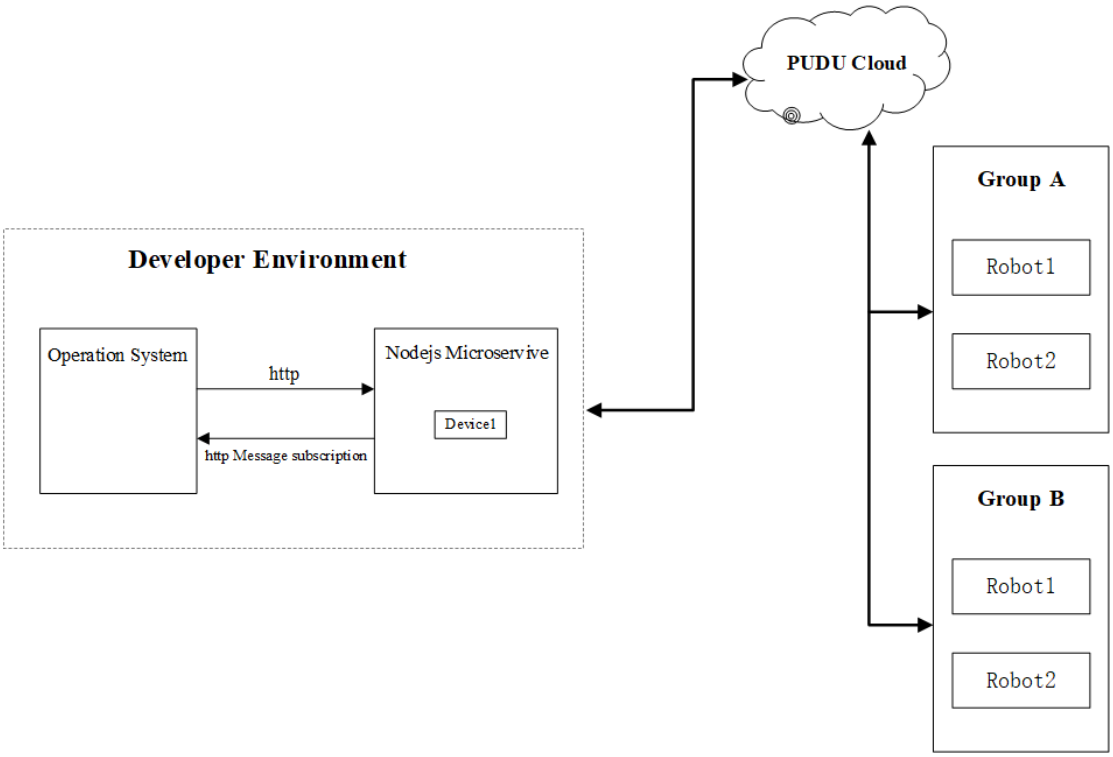
Message subscription address

If developers need to receive the notification from robots， they need to register a http interface by using the “register notification message address” in microservice, then microservice will transfer the message to this interface after receive the message from robots. Please check "Message Notification" for specific notification types and protocols

Service Protocol

Notification

Service Framework



The flow as above:

- Robot send the notification to Pudu Cloud.
- Pudu Cloud forward message to the robot which the device has bound to.
- Robot opens the microservice to forward notification according to the notification interface registered by the developer

Notification is POST, see the protocol below:

Field	Type	Necessity	Description
Msg Id	String	Y	Message ID
Msg Type	String	Y	Message type

Robot Id	String	N	Device ID of robot
Group Id	String	N	Robot Group ID
Device Id	String	Y	Received device ID
data	Object	Y	Data object corresponding to the message type
timestamp	Long	Y	Timestamp ms

Return the content below after receive the message：

Field	Type	Description
success	boolean	result： true

Example：

```
{
"success":true
}
```

Device Status

If the device is logged in at two different places synchronously, the device will keep reconnected, so please log in and use only one microservice for a device key

Data object：

Field	Type	Necessity	Description

status	String	Y	connect : Device connect to server reconnect offline close

Example：

```
{
      "msgId":"xxxxxxxx",
      "msgType":"serviceDeviceStatusChange",
      "deviceId":"Received device ID",
   "timestamp": 1585832325829,
      "data":{
            "status":"connect"
      }
}
```

Order status notification（Apply from Pudu Cloud）

When the task performed by the robot has a food order, the robot will synchronize the message of the order status to the bound device or microservice.

Data object：

Field	Type	Necessity	Description
orderState	String	Y	There will be three states respectively; “Start” is to start meal delivery; “Cancel” is to cancel the

			task; “Complete” is to complete the task;
spendTime	Long	Y	The time spent in task execution, the unit is milliseconds, the Start state is 0
ids	Array<Object>	Y	Array of order ID
employeeId	String	N	Employee ID: if the order event is initiated with employee information, the employee ID will be returned

Example:

```
{
      "msgId":"xxxxxxxx",
      "msgType":"notifyRobotOrderState",
      "robotId":" Robot ID",
   "groupId":" Robot Group ID",
      "deviceId":" Received device ID",
   "timestamp": 1585832325829,
      "data":{
            "orderState":"Start/Cancel/Complete", // There will be three states respectively: “Start” is to start meal delivery; “Cancel” is to cancel the task; “Complete” is to complete the task            "ids":[                        // The unique ID of the order
         {
            "id":"11122223334444555",
```

```
            "spendTime": 0       // Time spent from Start to Cancel or Complete,
unit: ms          },
        {
            "id":"11122223334444555",
            "spendTime": 0
        }
            ],
      "employeeId": "emplotee1"
      }
}
```

Message returned:

```
{
      "success":true
}
```

Notification of call task status

When making a call to the robot, the robot will synchronize the call status while performing the call task.

Data object：

Field	Type	Necessity	Description
robotGoState	String	Y	Call task execution status: “Arriving” is starting to go; “Arrived” is having arrived at the call point; “Cancel” refers to the call task is canceled;

destination	Object	Y	Target position

Example:

```
{
    "msgId":"xxxxxxxx",
    "msgType":"notifyGoState",
    "robotId":" Robot ID",
  "groupId":" Robot Group ID",
    "deviceId":" Received device ID",
  "timestamp": 1585832325829,
    "data":{
        "robotGoState":"Arriving/Arrived/Cancel",
        "destination":{
            "name":"A1",
            "type":"transit"
        }
    }
}
```

Message returned:

```
{
    "success":true
}
```

Notification of robot status

When passing the "Query the status of robots in the group" interface, the online robots in the group will notify the Microservice of their status after receiving the

message, and the Microservice will then send the message to the subscribed interface.

Data object：

Field	Type	Necessity	Description
moveState	String	Y	Robot movement status （Please check “Robot movement status notification for more details）
robotState	String	Y	Robot current status： Free： Robot is currently idle, summon the robot by “call interface” and the robot will execute the task immediately. Busy： Robot is in busy， if someone tapping the screen during the delivery mission（The robot will enter into busy state for a while when someone tapping the screen and it will back to idle state if no one touch it after a few seconds）
robotId	String	Y	
chargeStage	String	Y	Battery status（Please check “robot battery

			notification" for more details)
robotPower	int	Y	Battery power
robotPose	Object	Y	Please check "robot posture notification" for more details

Example:

```
{
    "msgId":"xxxxxxxx",
    "msgType":"query_state",
    "robotId":"Robot ID",
  "groupId":"Robot Group ID",
    "deviceId":"Received device ID",
  "timestamp": 1585832325829,
    "data":{
                    "robotState":"Free",
              "robotId":"xxxxxxxx",
              "moveState" : "Arrive",
              "chargeStage" : "Idle",
                  "robotPower":99,
                  "robotPose":{
                        "x":1.0,
                              "y":1.0,
                            "angle":1.0,
                  }
    }
}
```

Message returned:

```
{
    "success":true
}
```

Notification of robot delivery task status（Apply from Pudu Cloud）

Execution status message of the delivery task will be sent to microservice by Pudu Cloud when the robot is delivering （including the task sent by microservice and directly issue on robot）

Field in data object：

Field	Type	Necessity	Description
trays	Array	Y	Tray arrays
deliveryMode	String	Y	Delivery mode： General：task sent by microservice will be normal delivery mode. Direct：direct delivery mode Birthday：birthday mode Special：special mode

data in trays array

Field	Type	Necessity	Description
destinations	Array	Y	Task on each tray

Data in destinations

Field	Type	Necessity	Description
destination	String	Y	Target point：This value is the target point obtained by "getting the robot's map target point"
type	String	Y	Task type： remote：task sent remotely manual：manually edited tasks on the robot
id	String	N	Task id, If the robot task is executed with the task id by "send delivery task to the robot" The robot will notify the developer of the received id by this field
status	String	Y	Task status Await On The Way Arrived Cancel Complete
estimatedTime	Long	Y	Estimate time to arrive, When the task start at “On The Way”, it will calculate how long to finish this task, unit: ms （it is calculated based on the distance and speed when starting the task and it might be

			inaccurate if encountered any obstacle)
spendTime	Long	Y	The total time it takes for the delivery status to change, it starts to calculate from “On The Way” of first task to the end of this task complete. Unit: ms
completeType	String	N	There are many ways for the robot to complete the task. This field will return how the robot completed the task after it arrived: TIMEOUT：Complete the task overtime after arrival. MANUAL：Manually click to complete. REMOTE：Remotely complete QRCODE: Scan QR code to complete. TRAY_EMPTY：Tray detection is automatically completed when empty

Example：

```
{
    "msgId":"xxxxxxxx",
    "msgType":"notifyDeliveryTask",
    "robotId":"Robot ID",
```

```
  "groupId":"Robot Group ID",
     "deviceId":" Received device ID ",
  "timestamp": 1585832325829,
     "data":{
     "deliveryMode":"General",
          "trays":[
               {
                    "destinations":[{
                                        "destination":"A1",
                                        "type":"remote/manual",
                                        "id":" Task id, will be returned when
the status is synchronized ",
                                   "estimatedTime": 60000,
                                   "spendTime":80000,

"status":"Await/OnTheWay/Arrived/Cancel/Complete"
                                        }]

               },
               {
                    "destinations":[{
                                        "destination":"A1",
                                        "type":"remote/manual",
                                        "id":" Task id, will be returned when
the status is synchronized ",
                                   "estimatedTime": 60000,
                                   "spendTime":80000,

"status":"Await/OnTheWay/Arrived/Cancel/Complete"
                                        }]
               },
               {
                    "destinations":[{
```

```
                                        "destination":"A1",
                                        "type":"remote/manual",
                                        "id":" Task id, will be returned when
the status is synchronized ",
                                "estimatedTime": 60000,
                                "spendTime":80000,

"status":"Await/OnTheWay/Arrived/Cancel/Complete"
                                        }]
            }
        ]

    }
}
```

Message returned:

```
{
    "success":true
}
```

Notification of Robot battery power（Apply from Pudu Cloud）

It will be sent when the battery power changes or the battery status changes.

Data object：

Field	Type	Necessity	Description
power	int	Y	Power percentage: 0 ~ 100
chargeStage	String	Y	Power status:

			Idle: Charging Charge Full Charge Error Contact Charge Error Electric Error Battery Pack Comm Error Over Volt Error Over Electric Error Over Temperature Error Over Time

Example:

```
{
    "msgId":"xxxxxxxx",
    "msgType":"notifyRobotPower",
    "robotId":"Robot ID",
  "groupId":"Robot Group ID",
    "deviceId":" Received device ID ",
  "timestamp": 1585832325829,
    "data":{
          "power":80,
    "chargeStage":"Idle"
    }
}
```

Message returned:

```
{
    "success":true
}
```

Notification of Robot movement status（Apply from Pudu Cloud）

Status notification of the robot in the process of running to a target point. Such as it will be 3 target points “A1”, “A2”, “A3” during a delivery
The robot will start to arrive at "A1", movement status is
Moving->Approaching->Arrive->Idle

The robot starts from "A1" to "A2", encountering obstacles, tasks pause, and robot scheduling during the process. Robot status is：
Moving->Stuck(encounter obstacle)->Moving(bypass obstacle)->Pause(task pause)->Moving(recover moving)->Avoid(two robots cross over and scheduling)->Moving(scheduling over and moving again)->Approaching->Arrive->Idle

If the robot goes from "A2" to "A3", an errors occurs in the process
Moving->Error(the robot sill stop running if any error occurred)->Moving(running moving again after error removed)->Approaching->Arrive->Idle

Data object：

Field	Type	Necessity	Description
state	String	Y	Notification of robot movement status: Idle Moving Stuck: stuck by obstacle Approaching: close to the target point Arrive

			Pause Avoid: Scheduling with the other robots Error
errors	Array	N	Error information data

Example:

```
{
      "msgId":"xxxxxxxx",
      "msgType":"notifyRobotMoveState",
      "robotId":"Robot ID",
   "groupId":"Robot Group ID",
      "deviceId":" Received device ID ",
   "timestamp": 1585832325829,
      "data":{
            "state": "Idle",
      "errors": [
        {
          "errorType":"",
          "level":"",
          "detail":""
        }
      ]
      }
}
```

Message returned:

```
{
      "success":true
```

```
}
```

Notification of robot posture（Apply from Pudu Cloud）

Robot will notify its own coordinate and orientation to microservice when start to move. (The coordinate is the map used by the robot)

Data object：

Field	Type	Necessity	Description
x	Double	Y	X coordinate on the map
y	Double	Y	Y coordinate on the map
angle	Double	Y	The positive direction of the x-axis is 0, the positive direction of x-axis rotates from the positive direction of the y-axis to the negative direction of the x-axis, which changes from 0 to 180; The positive x-axis rotation from the negative y-axis to the negative x-axis changes from 0 to -180

Example：

```
{
    "msgId":"xxxxxxxx",
    "msgType":"notifyRobotPose",
    "robotId":"Robot ID",
```

```
  "groupId":"Robot Group ID",
    "deviceId":" Received device ID ",
  "timestamp": 1585832325829,
    "data":{
          "x": 111.1,
    "y": 2222.4,
    "angle": 0.333
    }
}
```

Message returned:

```
{
    "success":true
}
```

Robot QR code message return

（Support V1.0.10 and above）

According to the robot's QR code protocol, when the robot reads the QR code type is 3, the robot will send the customize content on the code to the developer through this notification type。

Message type： notifyQrCodeContent

Data object：

Field	Type	Necessity	Description
content	String	Y	Developer customize content on the QR code

Example：

```
{
```

```
    "msgId":"xxxxxxxx",
    "msgType":"notifyQrCodeContent",
    "robotId":"Robot ID",
  "groupId":"Robot Group ID",
    "deviceId":" Received device ID ",
  "timestamp": 1585832325829,
    "data":{
          "content":"111|1222|4444",
    }
}
```

Message returned:

```
{
    "success":true
}
```

Request Protocol

All the following request protocol links is SDK deploy IP and port splicing, such as http://127.0.0.1:90

Add a device

POST /api/add/device

The device ID cannot be duplicated, and each device can only allow one microservice to log in.

Field	Type	Necessity	Description

deviceName	String	N	Custom device name
deviceId	String	N	Device name registered in Pudu Cloud
deviceSecret	String	N	Device key registered in Pudu Cloud
region	String	N	(V1.2.1)device area: default is cn-shanghai, it based on which area the equipment belongs to for different situation.

request:

```
{
    "deviceName":"Shenzhen Store",
  "deviceId":"",              // Device id, apply from Purdue platform
  "deviceSecret":"",    // Device key, apply from Purdue platform
  "region":"cn-shanghai"
}
```

response

```
{
    "code":0,
  "msg":"",
  "data":{
    "success":true
  }
}
```

Delete device

POST /api/delete/device

request:

```
{
   "deviceId":"xxxxxx"
}
```

response

```
{
      "code":0,
   "msg":"",
   "data":{
      "success": true
   }
}
```

Modify device name

POST /api/device/name

request:

```
{
      "name":"Shenzhen Store",
   "deviceId":"xxxxxx"
}
```

response

```
{
      "code":0,
```

```
  "msg":"",
  "data":{
    "success": true
  }
}
```

Get all devices

GET /api/devices

response

```
{
    "code": 0,
    "msg": "",
    "data": {
        "devices": [
            {
                "name": "Guangdong Store",
                "deviceId": "xxxxxxxx"
            },
            {
                "name": "Shanghai Store",
                "deviceId": "xxxxxxxx"
            }
        ]
    }
}
```

Get the bound robot groups

GET /api/robot/groups?device=device id

Robot groups are added and modified on the cloud platform. Each store come with one robot group by default. Users can add multiple groups under the same

store and assign robots to different groups. The robots in each group must use the same map.

response

```
{
    "code": 0,
    "msg": "",
    "data": {
        "robotGroups": [
            {
                "id": "groups_1",
                "name":"group name",
                "shop_name": "shop name this group belongs to"
            },
            {
                "id": "groups_2",
                "name":"组名",
                "shop_name": "shop name this group belongs to"
            }
        ]
    }
}
```

Get the robots in the robot group

GET /api/robots?device=device id&group_id=robot group id

response

```
{
    "code": 0,
    "msg": "",
    "data": {
        "robots": [
```

```
            {
                "id": "robot1",
                "name": "XB", // Robot name, name can be edited in Pudu
Cloud
                "shop_name": "shop name"
            },
            {
                "id": "robot2",
                "name": "XB", // Robot name
                "shop_name": "shop name"
            }
        ]
    }
}
```

Obtain the target point data in map

GET /api/destinations?device=device id&robot_id=robot id&page_size=200&page_index=1

To get the target point data, you can first obtain the id of the online robot in the group through the status query interface, and then query the relevant data through the online robot.

request:

Field	Type	Necessity	Description
deviceId	String	Y	Device ID of the microservice
robotId	String	Y	Robot ID

page_size	int	N	Paging to get the maximum number of each page, minimum:10, maximum:300
page_index	int	N	Which page, starting from page 1

response:

Field	Type	Necessity	Description
name	String	Y	Table number
type	String	Y	Table type table outlet/parking point transit; dishwashing;
total	int	Y	Total target points
pageSize	int	Y	Page size
pageIndex	int	Y	Which page, starting from page 1


```
{
    "code": 0,
    "msg": "",
    "data": {
        "total":434,
        "pageSize":200,
        "pageIndex":1,
        "destinations": [
            {
```

```
                "name": "1",
                "type": "table"
            },
            {
                "name": "2",
                "type": "table"
            },
            {
                "name": "home1",
                "type": "dining_outlet"
            },
            {
                "name": "recycling 1",
                "type": "transit"
            },
            {
                "name": "dishwashing
A",
                "type": "dishwashing"
            }
        ]
    }
}
```


Register the notification address

POST /api/notify/url

Register the message listener address, the notification will return the message through the registered interface.

Field	Type	Necessity	Description

url	String	Y	Register listening interface

request：

```
{
    "url":"http://127.0.0.1:60001/api/"
}
```

response

```
{
    "code": 0,
    "msg": "",
    "data": {
        "success": true
    }
}
```

Robot status in the group

POST /api/robot/state

Send status query instructions to all robots in the group, after the robot receives the instruction, it will return its own current state which only the robots online will return it. The microservice will send out the status messages of the robots in the group.

Field	Type	Necessity	Description
deviceId	String	Y	Device ID added by the microservice
groupId	String	Y	Robot group ID

request:

```
{
   "deviceId":"device id",
   "groupId":"Robot group    ID"
}
```

response

```
{
      "code": 0,
      "msg": "",
      "data": {
            "success": true
      }
}
```

Get the status of robots in the group

GET /api/robot/state?device_id=device ID&group_id=group ID&timeout=5&count=5

Send status query instructions to all robots in the group, the microservice will wait for the status message of the corresponding robot, and return the obtained data when it times out.

Field	Type	Necessity	Description
device_id	String	Y	Device ID added by the microservice
group_id	String	Y	Robot group ID
timeout	int	N	It will need a timeout to wait for the result Since

			getting the status of multiple robots have to wait for their response, default: 5s, value range: 2-10
count	int	N	Return the result after the robot data obtained

response

Field	Type	Necessity	description
moveState	String	Y	The movement status of the robot (please see "Notification of Robot Movement Status" for more details)
robotState	String	Y	Current state of the robot: Free: Robot is currently idle, summon the robot by “call interface” and the robot will execute the task immediately. Busy: Robot is in busy, if someone tapping the screen during the delivery mission（The robot will enter into busy state for a while when someone tapping the screen and it

			will back to idle state if no one touch it after a few seconds)
robotId	String	Y	
chargeStage	String	Y	Battery status（Please check “robot battery notification” for more details)
robotPower	int	Y	Battery power
robotPose	Object	Y	Please check “robot posture notification” for more details


```
{
    "code": 0,
    "msg": "",
    "data": {
        "state": [
          {
            "robotState":"Free",
            "robotId":"xxxxxxxx",
            "moveState" : "Arrive",
            "chargeStage" : "Idle",
                "robotPower":99,
                "robotPose":{
                    "x":1.0,
                        "y":1.0,
                    "angle":1.0,
                }
          },
          {
```

```
                "robotState":"Busy",
                "robotId":"xxxxxxxx",
                "moveState" : "Pause",
                "chargeStage" : "Idle",
                    "robotPower":99,
                    "robotPose":{
                        "x":1.0,
                                    "y":1.0,
                                "angle":1.0,
                    }
            },
        ]
    }
}
```

Get the status of the designate robot

GET /api/robot/status?device_id=device ID&robot_id=xxxxx

If the robot is not online, the request will time out.

Field	Type	Necessity	Description
device_id	String	Y	Device ID added by the microservice
robot_id	String	Y	Robot ID

response

Field	Type	Necessity	Description

moveState	String	Y	The movement status of the robot (please see "Notification of Robot Movement Status" for more details
robotState	String	是	Current state of the robot: Free: Robot is currently idle, summon the robot by “call interface” and the robot will execute the task immediately. Busy: Robot is in busy, if someone tapping the screen during the delivery mission（The robot will enter into busy state for a while when someone tapping the screen and it will back to idle state if no one touch it after a few seconds）
robotId	String	Y	
chargeStage	String	Y	Battery status（Please check “robot battery notification” for more details）
robotPower	int	Y	Battery power

robotPose	Object	Y	Please check “robot posture notification” for more details


```
{
    "code": 0,
    "msg": "",
    "data": {
        "robotState":"Free",
        "robotId":"xxxxxxxx",
        "moveState" : "Arrive",
        "chargeStage" : "Idle",
        "robotPower":99,
        "robotPose":{
           "x":1.0,
                    "y":1.0,
                    "angle":1.0,
        }
    }
}
```


Call the robot by a target point on the map

POST /api/robot/call

The map target point can be obtained by "Getting the robot's map target point data" by directly using the map target point to summon the robot. If a call task is sent to a robot in the Free state, the robot task can execute the task immediately, If a call task is sent to a robot in the Busy state, the robot will execute the call task after entering the Free state.

Field	Type	Necessity	Description
deviceId	String	Y	Device ID added by the microservice
robotId	String	Y	The designate robot ID
destination	Object	Y	target

request：

```
{
   "deviceId":"device id",
   "robotId":"Robot ID",
   "destination":{
      "name": "home1", // target point name
      "type": "dining_outlet"
    }
}
```

response

```
{
      "code": 0,
      "msg": "",
      "data": {
            "success": true
      }
}
```

Cancel call task through target point

POST /api/robot/cancel/call

Cancel the call task: The device can only cancel the call task sent by itself. After canceling the call task, the robot will "return home" and return to the parking point.

Field	Type	Necessity	Description
deviceId	String	Y	Device ID added by the microservice
robotId	String	Y	Designate robot ID
destination	Object	Y	target

request：

```
{
      "robotId":" The id of the robot that successfully called ",
    "deviceId":"device id",
    "destination":{                  // Location for cancel call
        "name": "home1",
        "type": "dining_outlet"
    }
}
```

response

```
{
      "code": 0,
      "msg": "",
      "data": {
            "success": true
      }
}
```

Send a delivery task to the robot

POST /api/robot/delivery/task

When the robot is at the edit interface, this command can remotely push the delivery task instead of manually clicking the robot. If it is not in this interface, the command will fail.

Field	Type	Necessity	Description
deviceId	String	Y	Device ID added by the microservice
robotId	String	Y	Designate robot ID
type	String	Y	new：new task that can be received in the edit task interface； modify: the robot in the delivery task can forcibly modify the delivery task
deliverySort	String	Y	auto：the robot sorts the delivery task from the nearest point. Fixed**:** delivery according to the order of tasks
executeTask	boolean	Y	Whether to perform the task directly: true：the robot will execute the task directly after receiving the command.

			false: only the relevant tasks will be entered in the robot delivery task interface, and you need to manually click to start or call the Start command of the "send operation instructions to the machine" interface to start
trays	Array	Y	Tray list

Trays list object

Field	Type	Necessity	Description
destinations	Array	Y	Target on each tray

Destinations object

Field	Type	Necessity	Description
destination	String	Y	destination
id	String	Y	Task id will be returned when the delivery status is notified, you can leave it blank

request:

```
{
    "robotId":" id of the robot that successfully called ",
  "deviceId":"device id",
```

```
  "type":"new/modify",
  "deliverySort":"auto/fixed",
  "executeTask": false,
    "trays":[
            {
                "destinations":[{
                                    "destination":"A1",
                                    "id":" Task id, will be returned when
the status is synchronized "
                                }]
            },
            {
                "destinations":[{
                                    "destination":"A2",
                                    "id":" Task id, will be returned when
the status is synchronized "
                                }]
            },
            {
                "destinations":[{
                                    "destination":"A3",
                                    "id":" Task id, will be returned when
the status is synchronized "
                                }]
            }
        ]
}
```

response

```
{
    "code": 0,
    "msg": "",
```

```
    "data": {
        "success": true
    }
}
```

Send operating instructions to the robot.

POST /api/robot/action

Send relevant instructions to the robot instead of manual operation

Field	Type	Necessity	Description
action	String	Y	Start：To start the delivery task, it needs to take effect in the delivery task interface; Complete：To complete the delivery task, the instruction will only take effect when the arrival interface is displayed; Cancel All Delivery: The instructions only take effect when the robot is performing the delivery task（if the task was not completed, the instruction will invalid）
robotId	String	Y	Robot ID
deviceId	String	Y	Device ID added by the microservice

request:

```
{
      "robotId":" id of the robot that successfully called ",
   "deviceId":"device id",
   "action":"Start/Complete/CancelAllDelivery"
}
```

response

```
{
      "code": 0,
      "msg": "",
      "data": {
            "success": true
      }
}
```

Get the map of the robot

GET /api/robot/map?device_id=device ID&robot_id=Robot ID

To obtain the map information of the robot, you can use the information to draw the robot map and the coordinate. The coordinates and orientation of the robot are based on the map; each group only needs to be obtained from one of the robot because we recommend that the maps in the group are consistent.

Field	Type	Necessity	Description
robot_id	String	Y	Robot ID
device_id	String	Y	Device ID added by the microservice

response

Field	Type	Necessity	Description
map	Object	Y	Map object
elements	Array	Y	Elements list of map

Elements object

Field	Type	Necessity	Description
type	String	Y	Element type

Related fields of the element type：

Type: source：

The target point on the map. Including dining table, recycling cabinet, dishwashing spot, picking spot, temporary parking

Field	Type	Description
vector	double[3]	[x,y,z] x,y is coordinate z is direction, but this is field temporarily 0
name	string	Name, not used yet
mode	string	table dining_outlet transit

		dishwashing parking
id	string	Usually means table number and cabinet
group	string	Group name, blank means the group unassigned

Type: track

Map paths can be drawn from this data

Field	Type	description
vector	double[4]	[x1,y1,x2,y2] means a straight line between (x1,y1) and (x2,y2)
width	double	Track width
maxSpeed	double	Maximum speed
id	string	A track may consist of multiple straight lines connected end to end, same

		id means same track

Type: cycle

Cruising track

Field	Type	Description
vector	double[]	Composed of multiple adjacent nodes, such as [1,2,3,4,3,2,1]


```
{
    "code": 0,
    "msg": "",
    "data": {
      "map":{
        "elements": [
          {
              "type": "track",
              "vector": [
                0.0930061,
                -0.371943,
                -13.0171,
                -0.783049
              ],
              "width": 0,
              "maxSpeed": 0,
              "id": "0"
          }
        ]
      }
```

```
    }
}
```

Send order to the robot tray

（support v1.0.10 or above）

POST /api/robot/tray/orders

Set the order list for the robot's tray. The order can only be entered when the robot is on the task editing interface, because there is only one target point for each trays and the target points in the order must be the same.

request：

Field	Type	Necessity	Description
deviceId	String	Y	Device id
orders	Array	Y	Order list
robotId	String	Y	Robot id
trayIndex	Int	Y	Specify which level of tray to set the order, and the robot tray input interface starts from top to bottom 1. 0 means no tray is specified and select the tray by default.

Orders array object

Field	Type	Necessity	Description

tableNo	String	Y	Table number （targets in the protocol list must be the same）
tableName	String	Y	Table name, must be same with the target point on the map （targets in the protocol list must be the same）
name	String	Y	Dish name
amount	float	Y	Quantity
id	String	Y	The unique id of the order which the robot will synchronize the order status during running


```
{
 "deviceId":"beeper8",
 "robotId":"Robot ID",
 "orders":[
   {
      "tableNo":"2",
      "tableName":"A2",
      "name":"milk tea",
      "amount":1.0,
      "id":"43243243242353643"
    },
   {
      "tableNo":"3",
      "tableName":"A3",
      "goodsName":"milk tea 1",
      "amount":2.0,
```

```
        "id":"4fdsafddsafd3643"
      }
  ]
}
```

response

```
{
      "code": 0,
      "msg": "",
      "data": {
            "success": true
      }
}
```

Error code

Code	description
10001	Failed to request external service
10002	Request for extranet service error
10003	Missing device
10004	IoT request failed

10005	Parameter error
20004	Pager error
20006	Cancel call failed
20007	Call failed
20008	Call failed, the call target point does not exist
20009	Call failed, the robot turned off the call function
20010	Call failed, another pager has called the location