HIKROBOT

RCS-2000 API

Developer Guide

Legal Information

This Document (hereinafter referred to be "the Document") is the property of Hangzhou Hikrobot Digital Technology Co., Ltd. or its affiliates (hereinafter referred to as "Hikrobot"), and it cannot be reproduced, changed, translated, or distributed, partially or wholly, by any means, without the prior written permission of Hikrobot. Unless otherwise expressly stated herein, Hikrobot does not make any warranties, guarantees or representations, express or implied, regarding to the Document, any information contained herein.

LEGAL DISCLAIMER

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THE DOCUMENT IS PROVIDED "AS IS" AND "WITH ALL FAULTS AND ERRORS". HIKROBOT MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT. IN NO EVENT WILL HIKROBOT BE LIABLE FOR ANY SPECIAL, CONSEQUENTIAL, INCIDENTAL, OR INDIRECT DAMAGES, INCLUDING, AMONG OTHERS, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION OR LOSS OF DATA, CORRUPTION OF SYSTEMS, OR LOSS OF DOCUMENTATION, WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), OR OTHERWISE, IN CONNECTION WITH THE USE OF THE DOCUMENT, EVEN IF HIKROBOT HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES OR LOSS.

Contents

Chapter 1 Overview

1.1 Introduction

This manual provides some open APIs designed on RESTful style for the third-party platform to connect to Robot Control System (hereafter referred as to "RCS-2000") and control Automatic Guided Vehicles (AGVs) for logistics scheduling in the factory. Some typical applications developed by these open APIs are also provided for reference.

The RCS-2000 is a logistics scheduling system used in the smart factory, it can generate and assign tasks to different kinds of AGV to carry the racks, materials, goods, and so on, according to the task conditions. It can also monitor the task executing status to processing the exception situations in time, which makes sure the regular and accurate working of AGVs.

1.2 Update History

Summary of Changes in Version 3.1.0_March/2020

Version	Summary of Changes
Version 3.1.0_July/ 2020	1. Edited the API absolute address to "/rcms/services/rest/ hikRpcService/[apiName]".
	2. Edited the default port No. in the API to 8182.
	3. Updated the API ***genAgvSchedulingTask*** : added the forklift function.
	4. Extended the request message of API ***cancelTask*** : added one parameter **forceCancel** (task cancel mode).
	5. Extended the request message of API ***bindPodAndBerth*** : added one parameter **podDir** (rack direction).
	6. Added one API of stopping the specified AGV or all AGVs: ***stopRobot*** .
	7. Added one API of resuming the AGV: ***resumeRobot*** .
	8. Added one API of blocking or unblocking the area: ***setAreaState*** .
	9. Edited the API of searching for the AGV status ***queryAgvStatus*** : edited the request URL to "http://[address]:8083/rcms-dps/rest/ queryAgvStatus".

Summary of Changes in Version 2.2.3_June/2019

New document.

1.3 Notice

- When the RCS-2000 calls the APIs provided by the third-party platform, the default connection timeout threshold is 30 seconds, and the default returning timeout threshold is 60 seconds. If the timeout is longer than the default thresholds, the RCS-2000 will return the information of connection failed.
  If connecting to the third-party platform failed, you can try again after five seconds, and by default, up to 5 failure connection attempts are allowed.
- When calling an API, make sure the **reqCode** (request IDs) in the request and response message are same.
- If the third-party platform is developed by C# or JAVA language, we have provided demos to quickly start, please contact our technical supports to get the demos.
- The exhaustive request and response parameters are more than that introduced in this manual, the third-party platform can choose required parameters according to the business.
- REST (REpresentational State Transfer) is a protocol design method which abstracts all information as the resources. The abstracted resources are marked by the uniform identifies, i.e., URI (Uniform Resource Identifiers) for simple and extendable management.

Chapter 2 API Description

2.1 API Format

The APIs in this manual are all in URL format, which defines and provides a unique address for resources to access and implement different functions.

The detailed API format definition is shown below:

```
<protocol>://[address][:port][abs_path]
```

protocol

Protocol type that designing APIs based on, in this manual, the protocol type is "http".

address

Domain name or IP address of network device.

port

Port No.: for web server, the default port No. is 8182.

abs_path

An absolute address to define a resource, you can connect to and operate the resource via this address. It varies according to different platform.

- For RCS-2000, the absolute address of its resources is "/rcms/services/rest/hikRpcService/[**apiName**]".
- For the third-party platform, the absolute address of its resources is "/xxx/agv/[**apiName**]".

Note

- The [**apiName**] in the absolute address is used to distinguish the resources and functions, such as genAgvSchedulingTask, whose function is to generate task.
- To simplify the description in this manual, we use **apiName** to replace the complete API format.
- For API of searching for AGV status, the request URL is "http://[address][:port]/rcms-dps/rest/queryAgvStatus", and the port No. is 8083.

Base Access Address

To simplify API calling, you can define the path before **apiName** as the base access address **baseURL**. See the table below for details:

Platform	baseURL
RCS-2000	http://IP:PORT/rcms/services/rest/hikRpcService

Platform	baseURL
	Note The default web server port No. is 8182.
RCS-2000 (AGV status search)	http://IP:PORT/rcms-dps/rest **Note** The default port No. is 8083.
Third-party platform	http://IP:PORT/xxx/agv

2.2 Operation Method

To implement different functions of resources represented by each API, operation method is required. As the APIs in this manual is designed based on HTTP, the operation methods are same as that supported by HTTP.

Method	Description
POST	Create or add resources.
GET	Search or get resources.
PUT	Update or set resources.
DELETE	Delete resources.

Note

In this manual, only the POST operation method is available.

2.3 Message Format

During the development based on the open APIs, the request and response message for communication and interaction is in JSON format, and the nodes in the message are named by lower camel case.

JSON format is a subset of JAVA script, which is a lightweight data format, and this format can be quickly parsed. See the example below.

```
{
  "code": "0",
  "data": "F01169C808C317111G",
  "message": "successful",
  "reqCode": "468513"
}
```

2.4 Others

Time Format

The time appeared in the interaction between device and system adopts ISO8601 format, that is, "YYYY-MM-DD hh:mm:ss". For example, 2019-06-01 08:30:00.

Error Processing

When calling the open APIs, if error occurs, the response message will directly return the error code, you can get the error description and reason according to the returned response message. Currently, only three error codes may be returned in : 0 (succeeded), 1 (failed, incorrect parameter), and 99 (unknown error).

Chapter 3 Security

3.1 Authentication

The authentication of the open API is based on token (**tokenCode**) transmitted during request and response. The token is a string generated by Hikvision system and will be transmitted to the third-party platform for authentication when calling APIs.

Chapter 4 Typical Application

4.1 AGV Carries Rack and Leaves without Rack

In this application scene, the AGV carries a rack from the location A to a specific location B after the third-party platform calling an API to set task parameters and generate task. When arriving at location B, the AGV puts down the rack and directly leave. This application scene is usually available when the time consumption of processing goods on rack is long.

Steps

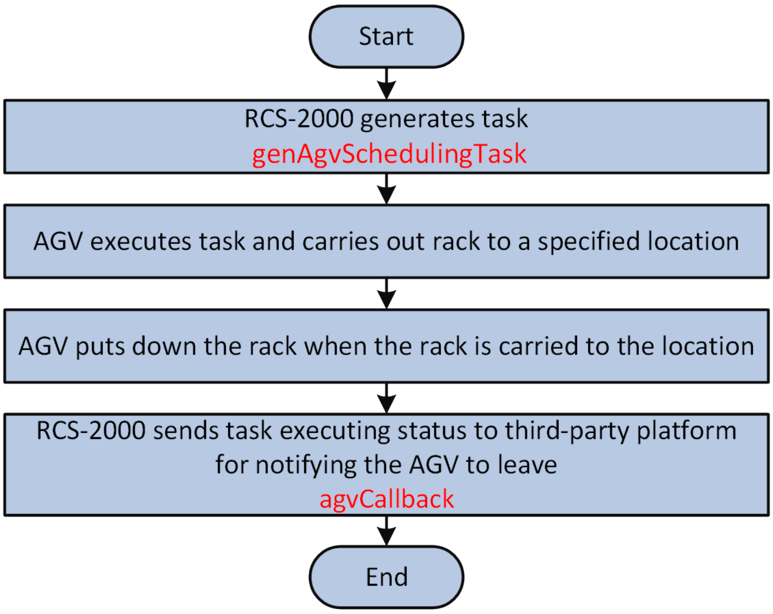

Start
RCS-2000 generates task
genAgvSchedulingTask
AGV executes task and carries out rack to a specified location
AGV puts down the rack when the rack is carried to the location
RCS-2000 sends task executing status to third-party platform for notifying the AGV to leave
agvCallback
End

Figure 4-1 Flow of AGV Carrying Rack and Leaving without Rack

1. The third-party platform calls ***genAgvSchedulingTask*** to generate task via the RCS-2000.

   The AGV starts executing the task and carrying a rack to a specified location.
2. The AGV puts down the rack when it arrived at the location.
3. The RCS-2000 calls ***agvCallback*** to send task executing status to the third-party platform for notifying the AGV to leave.

4.2 AGV Carries Rack and Backs with Rack

In this application scene, the AGV carries a rack from the location A to a specific location B after the third-party platform calling an API to set task parameters and generate task. When arriving at location B, the AGV keeps carrying the rack and waits until the third-party platform call an API to continue the task. And then the AGV carries back the rack whose goods has been processed to location A. This application scene is usually available when the time consumption of processing goods on rack is short.

Steps

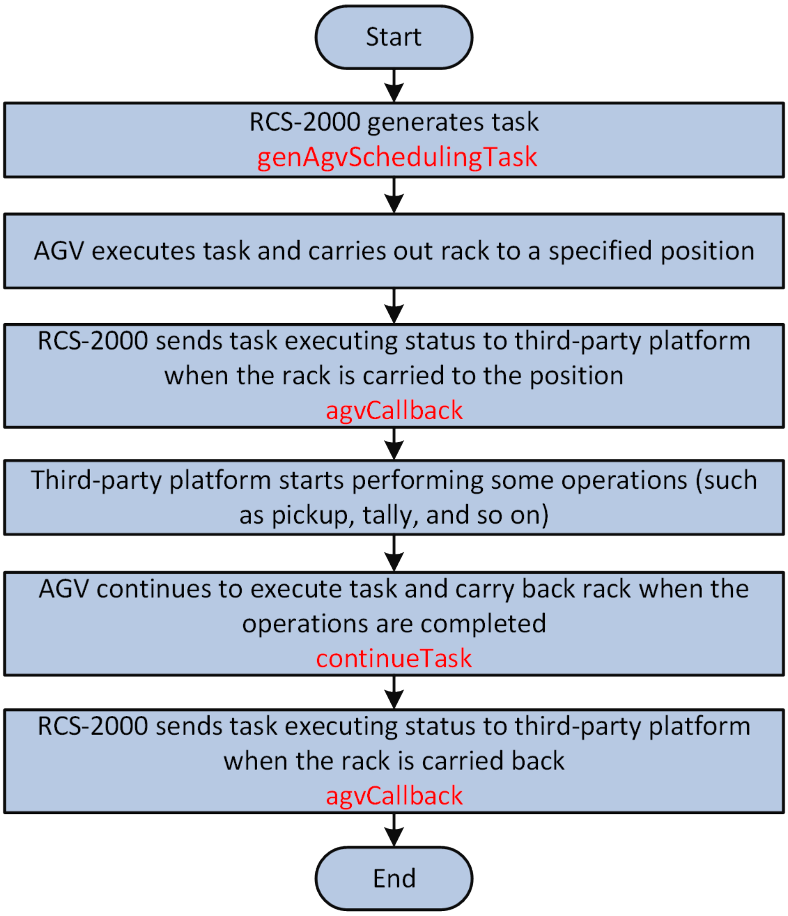

Start
RCS-2000 generates task
genAgvSchedulingTask
AGV executes task and carries out rack to a specified position
RCS-2000 sends task executing status to third-party platform when the rack is carried to the position
agvCallback
Third-party platform starts performing some operations (such as pickup, tally, and so on)
AGV continues to execute task and carry back rack when the operations are completed
continueTask
RCS-2000 sends task executing status to third-party platform when the rack is carried back
agvCallback
End

Figure 4-2 Flow of AGV Carrying Rack and Backing with Rack

1. The third-party platform calls ***genAgvSchedulingTask*** to generate task via the RCS-2000.

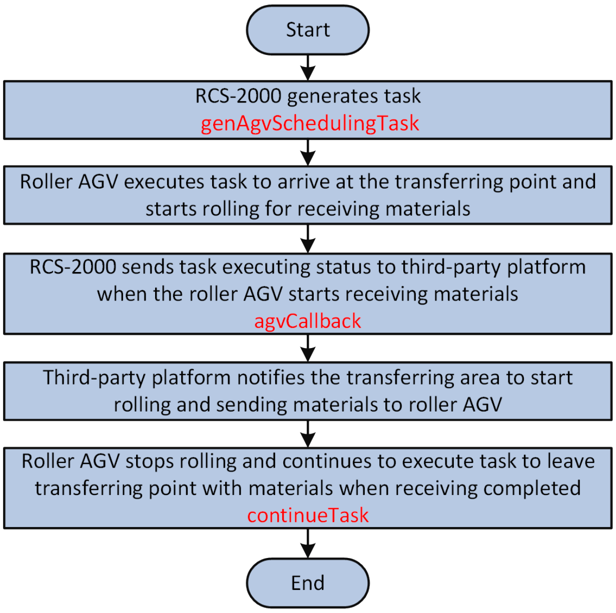
   The AGV starts executing the task and carrying out a rack to a specified location.
2. The RCS-2000 calls ***agvCallback*** to send task executing status to the third-party platform when the rack is carried to the location.
3. The third-party platform starts performing some operations, such as pickup, tally, and so on.
4. The third-party platform calls ***continueTask*** when all operations are completed to continue executing task.

   The AGV carries back the rack.
5. The RCS-2000 calls ***agvCallback*** to send task executing status to the third-party platform when the rack is carried back.

4.3 Roller AGV Receives Materials

The roller AGV is mainly applied to improve the automatic level of factory logistics. It can reduce the length of conveyor belt during long distance transportation; for the workshop which can not use forklift, it can convey very heavy material. Here introduces the progress of controlling roller AGV to complete the task of receiving materials.

Steps



Start

RCS-2000 generates task
genAgvSchedulingTask

Roller AGV executes task to arrive at the transferring point and starts rolling for receiving materials

RCS-2000 sends task executing status to third-party platform when the roller AGV starts receiving materials
agvCallback

Third-party platform notifies the transferring area to start rolling and sending materials to roller AGV

Roller AGV stops rolling and continues to execute task to leave transferring point with materials when receiving completed
continueTask

End

Figure 4-3 Flow of Roller AGV Receiving Materials

1. The third-party platform calls ***genAgvSchedulingTask*** to generate task via the RCS-2000.

   The roller AGV executes the task, arrives at the transferring point, and starts rolling to get ready for receiving materials.
2. The RCS-2000 calls ***agvCallback*** to send the task executing status to the third-party platform when the AGV is ready for receiving materials.
3. The third-party platform notifies the transferring area to start rolling and sending materials to the roller AGV.
4. The third-party platform calls ***continueTask*** to continue executing task when the materials is received by roller AGV..

   The roller AGV stops rolling and leaves the transferring point with received materials.

Chapter 5 API Reference

5.1 API List

List of Frequently Used APIs

Function	API Name	Provider
Create task	***genAgvSchedulingTask***	RCS-2000
Continue executing next sub task.	***continueTask***	RCS-2000
Cancel task	***cancelTask***	RCS-2000
Send task executing status to third-party platform	***agvCallback***	Third-party Platform

List of Rarely Used APIs

Function	API Name	Provider
Set task priority	***setTaskPriority***	RCS-2000
Bind and unbind rack and location	***bindPodAndBerth***	RCS-2000
Bind and unbind material batch and rack	***bindPodAndMat***	RCS-2000
Enable or disable the location	***lockPosition***	RCS-2000
Synchronize map information with that in RCS-2000	***syncMapDatas***	RCS-2000
Search relations among rack, location, and material batch	***queryPodBerthAndMat***	RCS-2000
Search for task executing status	***queryTaskStatus***	RCS-2000
Search for AGV status	***queryAgvStatus***	RCS-2000
Stop AGV	***stopRobot***	RCS-2000
Resume AGV	***resumeRobot***	RCS-2000
Block or unblock area	***setAreaState***	RCS-2000
Send alarm to the third-party platform	***warnCallback***	Third-party Platform

5.2 API Provided by RCS-2000

5.2.1 genAgvSchedulingTask

Robot Control System (RCS) creates task and the third-party platform applies the task to AGV.

API Definition

Table 5-1 POST http://[address][:port]/rcms/services/rest/hikRpcService/genAgvSchedulingTask

API Name	genAgvSchedulingTask				
Function	Robot Control System (RCS) creates task and the third-party platform applies the task to AGV.				
Protocol	REST				
Provider	RCS-2000				
Caller	Third-party platform				
Request	**Parameter**	**Data Type**	**Max. Length**	**Req. or Opt.**	**Description**
	reqCode	String	32	Req.	Request ID, if a same request is repeatedly submitted, the request ID must be same.
	reqTime	String	20	Opt.	Request timestamp, format: YYYY-MM-DD hh:mm:ss
	clientCode	String	16	Opt.	Third-party platform ID, e.g., PDA, HCWMS
	tokenCode	String	64	Opt.	Token ID, which is provided by RCS-2000 for third-party platform.
	taskTyp	String	16	Req.	Task types, which is same with the major task type configured in RCS-2000: • Build-in task type: "F01"-carry and transfer rack in factory, "F02"-switch empty rack with full rack in factory, "F03"-carry and transfer by roller AGV, "F04"-carry rack and back with rack in factory, "F05"-rotate rack, "F06"-elevator task. • Fork lift task type: "F11"-transfer from large rack to workstation, "F12"-transfer from workstation to large rack, "F13"-transfer from roadway to workstation, "F14"-transfer

					from workstation to roadway, "F15"-shuttle from large rack to workstation, " F16"-shuttle from workstation to large rack, "F17"-shuttle from roadway to workstation, "F18"-shuttle from workstation to roadway, "F20"-cross-floor forklift main task
	sceneTyp	String	2	Opt.	Scene: 0-normal AGV task, 1-forklift task. **Note** For creating forklift task, this node is required and its value should be "1", when this node is empty or does not exist, it indicates creating normal AGV task.
	ctnrTyp	String	16	Opt.	Container type, this node is required for forklift task.
	ctnrCode	String	32	Opt.	Container ID, this node is required for forklift task.
	wbCode	String	32	Opt.	Workstation ID, which consists of letters and digits, and it must be same with that configured by RCS-2000.
	positionCodePath	Object[]	50	Opt.	Rack moving pattern, which consists of multiple locations (up to 50 locations are allowed). **positionCode**: Location ID, which is predefined with detailed coordinate information on map; the maximum string length is 64 bytes. **type**: Location types: "00"-actual location on map, "01"-location of a specific material batch, "02"-available location of area selection strategy, "03"-rack location, "04"-available location in an area, "05"-bin ID (for forklift only), "06"-roadway ID (for forklift only)
	podCode	String	16	Opt.	Rack ID, it can be empty if no rack specified.
	podDir	String	4	Opt.	Rack directions: "180"-leftward, "0"-rightward, "90"-upward, "-90"-downward; it can be empty if no direction specified.
	podTyp	String	16	Opt.	Rack types: "1"-empty rack of all types (default), "2"-type of rack linked with

					configured workstation (if the linked rack is empty, it also represents empty rack of all types), other values-empty rack with a specific type.
	materialLot	String	32	Opt.	Material batch ID, it can be bound with rack by setting **materialLot** and **podCode** or **positionCode** at same time.
	priority	String	32	Opt.	Task priority, range: [1,127], and larger number corresponds to higher priority. If it is not configured, the task priority takes effect.
	taskCode	String	64	Opt.	Task ID, if it is not configured, the system will generate automatically.
	agvCode	String	16	Opt.	AGV ID, if it is not configured, the RCS-2000 will automatically select an optimal AGV to execute the task
	data	String	2000	Opt.	Custom content, up to 2000 characters are allowed.
Response	code	String	6	Req.	Status code, see ***Status Code*** for details.
	message	String	64	Req.	Returned status description, e.g., "successful"
	reqCode	String	64	Req.	Request ID, which must be same with that in corresponding request message
	data	String	2000	Opt.	Custom content to be returned, such as task ID
Remarks	One of **wbCode** and **positionCodePath** must be configured to confirm the location information in task. If you want to specify multiple locations in the task, e.g., start location and end location, the node **positionCodePath** should be configured.				

Sample	**Request**	{
		  "reqCode": "468513",
		  "taskTyp": "F01",
		  "positionCodePath": [{
		    "positionCode": "p01",
		    "type": "00"
		  },
		  {
		    "positionCode": "x02",
		    "type": "02"
		  }],
		  "podCode": "100001",
		  "podDir": "0",			

		"priority": "1", }

	Response	{
		"code": "0",
		"data": "F01169C808C317111G",
		"message": "successful",
		"reqCode": "468513"
		}

5.2.2 continueTask

Continue executing the next sub task.

API Definition

Table 5-2 POST http://[address][:port]/rcms/services/rest/hikRpcService/continueTask

API Name	continueTask				
Function	Continue executing the next sub task.				
Protocol	REST				
Provider	RCS-2000				
Caller	Third-party platform				
Request	**Parameter**	**Data Type**	**Max. Length**	**Req. or Opt.**	**Description**
	reqCode	String	32	Req.	Request ID, if a same request is repeatedly submitted, the request ID must be same.
	reqTime	String	20	Opt.	Request timestamp, format: YYYY-MM-DD hh:mm:ss
	clientCode	String	16	Opt.	Third-party platform ID, e.g., PDA, HCWMS
	tokenCode	String	64	Opt.	Token ID, which is provided by RCS-2000 for third-party platform.
	wbCode	String	32	Opt.	Workstation ID, it must be same with that configured by RCS-2000.
	podCode	String	16	Opt.	Rack ID
	agvCode	String	16	Opt.	AGV ID, if it is not configured, the RCS-2000 will automatically select an optimal AGV to execute the task.

	taskCode	String	64	Opt.	Task ID (UUID). If it is not configured, it will be automatically generated by RCS-2000, and it is 64 bits UUID.
	taskSeq	String	32	Opt.	Sub task No. to be specified to continue executing. If this node is not configured, by default, the next sub task will be executed.
	nextPositionCode	Object	40	Opt.	The location information of sub tasks. This node is required when the task type is externally configured. **positionCode**: position ID, which is predefined with detailed coordinate information on map. **type**: location types: "00"-actual location on map, "02"-strategy
Response	code	String	6	Req.	Status code, see ***Status Code*** for details.
	message	String	64	Req.	Returned status description, e.g., "successful"
	reqCode	String	64	Req.	Request ID, which must be same with that in corresponding request message
Remarks	One of **wbCode**, **agvCode**, **taskCode**, and **podCode** must be configured to confirm the task ID. The priority order of above four parameters is: **wbCode** > **agvCode** > **taskCode** > **podCode**, that is, only the value of **wbCode** will take effect if the above four parameters are all configured.				

Sample Codes	**Request**	{
		"reqCode": "123",
		"taskCode": "123456",
		"nextPositionCode": {
		"positionCode": "p02",
		"type": "00"
		}
		}			

	Response	{
		"code": "0",
		"message": "successful",
		"reqCode": "123"
		}			

5.2.3 cancelTask

Cancel the task that is executing or is generated but waiting for being executed.

API Definition

Table 5-3 POST http://[address][:port]/rcms/services/rest/hikRpcService/cancelTask

API Name	cancelTask				
Function	Cancel the task that is executing or is generated but waiting for being executed.				
Protocol	REST				
Provider	RCS-2000				
Caller	Third-party platform				
Request	**Parameter**	**Data Type**	**Max. Length**	**Req. or Opt.**	**Description**
	reqCode	String	32	Req.	Request ID, if a same request is repeatedly submitted, the request ID must be same.
	reqTime	String	20	Opt.	Request timestamp, format: YYYY-MM-DD hh:mm:ss
	clientCode	String	16	Opt.	Third-party platform ID, e.g., PDA, HCWMS
	tokenCode	String	64	Opt.	Token ID, which is provided by RCS-2000 for third-party platform.
	forceCancel	String	16	Opt.	Task cancel mode: "0" (default)-AGV puts down the rack at any location, and its status turns to "idle", "1"-AGV carries the rack and executes inbound, this mode is supported by latent mobile robot only.
	matterArea	String	16	Opt.	Inbound area No., it is valid only when the value of **forcecancel** is "1"; if this node is not configured, the inbound area is the warehouse area.
	agvCode	String	16	Opt.	AGV ID, whose executing task will be canceled.
	taskCode	String	64	Opt.	ID (UUID) of task to be canceled.
Response	code	String	6	Req.	Status code, see ***Status Code*** for details.
	message	String	64	Req.	Returned status description, e.g., "successful"
	reqCode	String	64	Req.	Request ID, which must be same with that in corresponding request message

Remarks	**Note** One of **taskCode** and **agvCode** must be configured to confirm the task to be canceled or the AGV to be freed. If both the two parameters are configured, only the value of **agvCode** will take effect.	

Sample Codes	**Request**	{
		"reqCode": "1541954B96B1112",
		"forceCancel": "1",
		"matterArea": "abc",
		"taskCode": "123456"
		}

	Response	{
		"code": "0",
		"message": "successful",
		"reqCode": "1541954B96B1112"
		}

Remarks

- If an AGV is executing a task and carrying a rack, when the task is canceled and the cancel mode is "0", the AGV will put down the rack at any location, and its status turns to "idle"; when the cancel mode is "1", the AGV will carry the rack and execute inbound, if there is no space of inbound area, the error will be returned and canceling failed.
- Forklift only supports the cancel mode "0".

5.2.4 setTaskPriority

Set task priority from 1 to 127, and the larger the value, the higher the priority. The priority takes effect only when the AGVs are not enough and there are multiple tasks with different priorities. The tasks will be assigned to the AGV according to priority order.

API Definition

Table 5-4 POST http://[address][:port]/rcms/services/rest/hikRpcService/setTaskPriority

API Name	setTaskPriority
Function	Set task priority from 1 to 127, and the larger the value, the higher the priority.
Protocol	REST
Provider	RCS-2000
Caller	Third-party platform

Request	**Parameter**	**Data Type**	**Max. Length**	**Req. or Opt.**	**Description**
	reqCode	String	32	Req.	Request ID, if a same request is repeatedly submitted, the request ID must be same.
	reqTime	String	20	Opt.	Request timestamp, format: YYYY-MM-DD hh:mm:ss
	clientCode	String	16	Opt.	Third-party platform ID, e.g., PDA, HCWMS
	tokenCode	String	64	Opt.	Token ID, which is provided by RCS-2000 for third-party platform.
	priorities	List			
	taskCode	String	64	Req.	Task ID (UUID)
	priority	String	32	Req.	Task priority, range: [1,127], the larger the value, the higher the priority
Response	code	String	6	Req.	Status code, see ***Status Code*** for details.
	message	String	64	Req.	Returned status description, e.g., "successful"
	reqCode	String	64	Req.	Request ID, which must be same with that in corresponding request message
	data	String	2000	Opt.	Custom content to be returned, such as task ID
Remarks	**Note** You can set the priority for the tasks that are not assigned to the AGV only; If the task has been assigned to AGV, it is invalid for setting task priority.				

Sample Codes	**Request**	{
		"reqCode": "1234567",
		"priorities": [{
		"priority": "1",
		"taskCode": "1232"
		},
		{
		"priority": "2",
		"taskCode": "3214"
		}]
		}			

	Response	{ "code": "0", "message": "successful",			

		"reqCode": "1234567" }

5.2.5 bindPodAndBerth

Bind and unbind rack and location.

API Definition

Table 5-5 POST http://[address][:port]/rcms/services/rest/hikRpcService/bindPodAndBerth

API Name	bindPodAndBerth				
Function	Bind and unbind rack and location.				
Protocol	REST				
Provider	RCS-2000				
Caller	Third-party platform				
Remarks	**Note** When unbinding, to avoid misoperation, both parameters **podCode** and **positionCode** should be configured for verification.				
Request	**Parameter**	**Data Type**	**Max. Length**	**Req. or Opt.**	**Description**
	reqCode	String	32	Req.	Request ID, if a same request is repeatedly submitted, the request ID must be same.
	reqTime	String	20	Opt.	Request timestamp, format: YYYY-MM-DD hh:mm:ss
	clientCode	String	16	Opt.	Third-party platform ID, e.g., PDA, HCWMS
	tokenCode	String	64	Opt.	Token ID, which is provided by RCS-2000 for third-party platform.
	podCode	String	16	Req.	Rack ID
	positionCode	String	32	Req.	Position ID, which is predefined with detailed coordinate information on map, and it is a unique string configured via RCS-2000.
	podDir	String	6	Opt.	Rack direction: "0"-horizontal (default), "1"-vertical. If this node is not configured, the rack direction is horizontal by default.

	indBind	String	1	Req.	Bind or unbind: "1"-bind, "0"-unbind.
Response	code	String	6	Req.	Status code, see ***Status Code*** for details.
	message	String	64	Req.	Returned status description, e.g., "successful"
	reqCode	String	64	Req.	Request ID, which must be same with that in corresponding request message

Sample Codes	**Request**	{
		"reqCode": "12345678",
		"podCode": "100001",
		"positionCode": "p05",
		"podDir": "0",
		"indBind": "1"
		}			

	Response	{
		"code": "0",
		"message": "successful",
		"reqCode": "12345678"
		}			

5.2.6 bindPodAndMat

Bind and unbind material batch and rack.

API Definition

Table 5-6 POST http://[address][:port]/rcms/services/rest/hikRpcService/bindPodAndMat

API Name	bindPodAndMat				
Function	Bind and unbind material batch and rack.				
Protocol	REST				
Provider	RCS-2000				
Caller	Third-party platform				
Remarks	**Note** When unbinding, to avoid misoperation, both parameters **podCode** and **materialLot** should be configured for verification.				
Request	**Parameter**	**Data Type**	**Max. Length**	**Req. or Opt.**	**Description**

	reqCode	String	32	Req.	Request ID, if a same request is repeatedly submitted, the request ID must be same.
	reqTime	String	20	Opt.	Request timestamp, format: YYYY-MM-DD hh:mm:ss
	clientCode	String	16	Opt.	Third-party platform ID, e.g., PDA, HCWMS
	tokenCode	String	64	Opt.	Token ID, which is provided by RCS-2000 for third-party platform.
	podCode	String	16	Req.	Rack ID
	materialLot	String	32	Req.	Material batch ID
	indBind	String	1	Req.	Bind or unbind: "1"-bind, "0"-unbind
Response	code	String	6	Req.	Status code, see ***Status Code*** for details.
	message	String	64	Req.	Returned status description, e.g., "successful"
	reqCode	String	64	Req.	Request ID, which must be same with that in corresponding request message
	data	String	2000	Opt.	Custom content to be returned, such as task ID

Sample Codes	**Request**	{
		"reqCode": "1541954B96B1112",
		"podCode": "100001",
		"materialLot": "123",
		"indBind": "1"
		}			

	Response	{
		"code": "0",
		"data": "",
		"message": "successful",
		"reqCode": "1541954B96B1112"
		}			

5.2.7 lockPosition

Enable or disable the location. When the location is disabled, it cannot be found in the area.

API Definition

Table 5-7 POST http://[address][:port]/rcms/services/rest/hikRpcService/lockPosition

API Name	lockPosition				
Function	Enable or disable the location. When the location is disabled, it cannot be found in the area.				
Protocol	REST				
Provider	RCS-2000				
Caller	Third-party platform				
Request	**Parameter**	**Data Type**	**Max. Length**	**Req. or Opt.**	**Description**
	reqCode	String	32	Req.	Request ID, if a same request is repeatedly submitted, the request ID must be same.
	reqTime	String	20	Opt.	Request timestamp, format: YYYY-MM-DD hh:mm:ss
	clientCode	String	16	Opt.	Third-party platform ID, e.g., PDA, HCWMS
	tokenCode	String	64	Opt.	Token ID, which is provided by RCS-2000 for third-party platform.
	positionCode	String	32	Req.	Position ID, which is predefined with detailed coordinate information on map, and it is a unique string configured via RCS-2000.
	indBind	String	1	Req.	Enable or disable: "1"-enable, "0"-disable
Response	code	String	6	Req.	Status code, see ***Status Code*** for details.
	message	String	64	Req.	Returned status description, e.g., "successful"
	reqCode	String	64	Req.	Request ID, which must be same with that in corresponding request message
	data	String	2000	Opt.	Custom content to be returned, such as task ID

Sample Codes	**Request**	{
		"reqCode": "1541954B96B1112",
		"positionCode": "p02",
		"indBind": "1"
		}			

	Response	{ "code": "0",			

		"message": "successful", "reqCode": "1541954B96B1112" }

5.2.8 syncMapDatas

Synchronize map information with that in RCS-2000.

API Definition

Table 5-8 POST http://[address][:port]/rcms/services/rest/hikRpcService/syncMapDatas

API Name	syncMapDatas				
Function	Synchronize map information with that in RCS-2000.				
Protocol	REST				
Provider	RCS-2000				
Caller	Third-party platform				
Request	**Parameter**	**Data Type**	**Max. Length**	**Req. or Opt.**	**Description**
	reqCode	String	32	Req.	Request ID, if a same request is repeatedly submitted, the request ID must be same.
	reqTime	String	20	Opt.	Request timestamp, format: YYYY-MM-DD hh:mm:ss
	clientCode	String	16	Opt.	Third-party platform ID, e.g., PDA, HCWMS
	tokenCode	String	64	Opt.	Token ID, which is provided by RCS-2000 for third-party platform.
	mapDataCode	String	32	Opt.	Unique location code on map
	mapShortName	String	32	Req.	Alias of the map that needs to be synchronized
	dataTyp	String	6	Opt.	Map element type, when this node is not configured, all location codes of the map will be synchronized.
Response	code	String	6	Req.	Status code, see ***Status Code*** for details.
	message	String	64	Req.	Returned status description, e.g., "successful"

	reqCode	String	64	Req.	Request ID, which must be same with that in corresponding request message
	data	List			
	cooX	String	8	Req.	X-coordinate of location code, unit: mm
	cooY	String	8	Req.	Y-coordinate of location code, unit: mm
	dataTyp	String	2	Req.	Map element types: "1"-storage section, "10"-workstation, "11"-charge station, "20"-interim storage area, "55"-roadway storage area
	direction	String	8	Opt.	Working station direction, in which a staff faces a rack to pickup: "180"-leftward, "0"-rightward, "90"-upward,"-90"-downward.
	mapCode	String	16	Req.	Map ID
	mapDataCode	String	32	Req.	Unique location code on map
	positionCode	String	32	Req.	Position ID, which is predefined with detailed coordinate information on map, and it is a unique string configured via RCS-2000.
	berthType	String	2	Opt.	Storage section types: "1"-outer storage section, "2"-inner storage section, "3"-normal storage section. **Note** This node is required when the value of **dataTyp** is 1.

Sample Codes	**Request**	{
		 "reqCode": "1541954B96B1112",
		 "reqTime": "",
		 "clientCode": "",
		 "tokenCode": "",
		 "mapDataCode": "xxxxxx",
		 "mapShortName": "xxxxxx",
		 "dataTyp": ""
		}			

	Response	{
		 "code": "0",
		 "message": "successful",
		 "reqCode": "1541954B96B1112",
		 "data": [{
		 "berthType": "3",			

```
    "cooX": "17000.0",
    "cooY": "18000.0",
    "dataTyp": "1",
    "direction": "0",
    "mapCode": "AA",
    "mapDataCode": "011724AA012414",
    "positionCode": "011724AA012414"
  },
  {
    "berthType": "3",
    "cooX": "11000.0",
    "cooY": "21999.0",
    "dataTyp": "10",
    "direction": "0",
    "mapCode": "AA",
    "mapDataCode": "007586AA015172",
    "positionCode": "104"
  }]
}
```

5.2.9 queryPodBerthAndMat

Search the relations among rack, location, and material batch.

API Definition

Table 5-9 POST http://[address][:port]/rcms/services/rest/hikRpcService/queryPodBerthAndMat

API Name	queryPodBerthAndMat				
Function	Search the relations among rack, location, and material batch.				
Protocol	REST				
Provider	RCS-2000				
Caller	Third-party platform				
Remarks	**Note** At least one of the following parameters should be configured: **podCode**, **materialLot**, **positionCode**, and **mapShortName**.				
Request	**Parameter**	**Data Type**	**Max. Length**	**Req. or Opt.**	**Description**
	reqCode	String	32	Req.	Request ID, if a same request is repeatedly submitted, the request ID must be same.

	reqTime	String	20	Opt.	Request timestamp, format: YYYY-MM-DD hh:mm:ss
	clientCode	String	16	Opt.	Third-party platform ID, e.g., PDA, HCWMS
	tokenCode	String	64	Opt.	Token ID, which is provided by RCS-2000 for third-party platform.
	podCode	String	16	Opt.	Rack ID
	materialLot	String	32	Opt.	Material batch ID
	positionCode	String	16	Opt.	Position ID, which is predefined with detailed coordinate information on map, and it is a unique string configured via RCS-2000.
	areaCode	String	16	Opt.	Area ID
	mapShortName	String	16	Opt.	Map alias
Response	code	String	6	Req.	Status code, see ***Status Code*** for details.
	message	String	64	Req.	Returned status description, e.g., "successful"
	reqCode	String	64	Req.	Request ID, which must be same with that in corresponding request message
	data	Object			
	areaCode	String	16	Opt.	Area ID
	materialLot	String	64	Opt.	Material batch ID
	podCode	String	16	Req.	Rack ID
	mapDataCode	String	32	Req.	Unique location code on map
	positionCode	String	32	Req.	Position ID, which is predefined with detailed coordinate information on map, and it is a unique string configured via RCS-2000.

Sample Codes	**Request**	{
		  "reqCode": "1541954B96B1110",
		  "mapShortName": "test"
		}			

	Response	{ "code": "0", "message": "successful",			

```
 "reqCode": "1541954B96B1110",
 "data": [{
   "podCode": "100001",
   "mapDataCode": "P02",
   "positionCode": "P02"
 },
 {
   "podCode": "100002",
   "mapDataCode": "P03",
   "positionCode": "P03"
 }]
}
```

5.2.10 queryTaskStatus

Search for the task executing status according to the task ID or AGV ID, or search for executing statuses of multiple tasks in batch.

API Definition

Table 5-10 POST http://[address][:port]/rcms/services/rest/hikRpcService/queryTaskStatus

API Name	queryTaskStatus				
Function	Search for the task executing status according to the task ID or AGV ID, or search for executing statuses of multiple tasks in batch.				
Protocol	REST				
Provider	RCS-2000				
Caller	Third-party platform				
Request	**Parameter**	**Data Type**	**Max. Length**	**Req. or Opt.**	**Description**
	reqCode	String	32	Req.	Request ID, if a same request is repeatedly submitted, the request ID must be same.
	reqTime	String	20	Opt.	Request timestamp, format: YYYY-MM-DD hh:mm:ss
	clientCode	String	16	Opt.	Third-party platform ID, e.g., PDA, HCWMS
	tokenCode	String	64	Opt.	Token ID, which is provided by RCS-2000 for third-party platform.
	taskCodes	String[ ]	64	Opt.	Array of task IDs, and at least one of **taskCodes** and **agvCode** should be configured.

	agvCode	String	16	Opt.	AGV ID, and at least one of **taskCodes** and **agvCode** should be configured
Response	code	String	6	Req.	Status code, see ***Status Code*** for details.
	data	Object			
	taskCode	String	64	Req.	Task ID (UUID)
	taskTyp	String	16	Req.	Task type
	taskStatus	String	2	Req.	Task status: "0"-sending exception, "1"-created, "2"-executing, "3"-sending, "4"-canceling, "5"-canceled, "6"-resending, "9"-completed, "10"-interrupted. Common values are: "0", "1", "2", "5", "9".
	agvCode	String	16	Opt.	AGV ID, this node exists only when the task has been assigned to.
	message	String	64	Req.	Returned status description, e.g., "successful"
	reqCode	String	64	Req.	Request ID, which must be same with that in corresponding request message.

Sample Codes	**Request**	{
		"reqCode": "1541954B96B1110",
		"taskCodes": ["123", "234"]
		}			

	Response	{
		"code": "0",
		"message": "successful",
		"reqCode": "1541954B96B1110",
		"data": [{
		"taskCode": "234",
		"taskStatus": "2",
		"taskTyp": "F01"
		},
		{
		"taskCode": "123",
		"taskStatus": "9",
		"taskTyp": "F01"
		}]
		}			

5.2.11 queryAgvStatus

Search for AGV status, including the AGV battery.

API Definition

Table 5-11 POST http://[address]:8083/rcms-dps/rest/queryAgvStatus

API Name	queryAgvStatus				
Function	Search for AGV status, including the AGV battery.				
Protocol	REST				
Provider	RCS-2000				
Caller	Third-party platform				
Remarks	• API calling frequency: number of AGVs is less than 100: 5 seconds; number of AGVs is between 100 and 200: 10 seconds; number of AGVs is between 200 and 300: 15 seconds. • The request URI is "http://[address]:8083/rcms-dps/rest/queryAgvStatus"				
Request	**Parameter**	**Data Type**	**Max. Length**	**Req. or Opt.**	**Description**
	reqCode	String	32	Req.	Request ID, if a same request is repeatedly submitted, the request ID must be same.
	reqTime	String	20	Opt.	Request timestamp, format: YYYY-MM-DD hh:mm:ss
	clientCode	String	16	Opt.	Third-party platform ID, e.g., PDA, HCWMS
	tokenCode	String	64	Opt.	Token ID, which is provided by RCS-2000 for third-party platform.
	mapShort Name	String	32	Opt.	Alias of the map where the AGV locates.
Response	code	String	6	Req.	Status code, see ***Status Code*** for details.
	data	Object			
	robotCode	String	16	Req.	AGV ID
	robotDir	String	4	Req.	AGV direction, range: [-180,360] degrees
	robotIp	String	64	Opt.	AGV IP address
	battery	String	4	Req.	AGV battery, range: [0,100]
	posX	String	8	Req.	AGV X-coordinate, unit: millimeter
	posY	String	8	Req.	AGV Y-coordinate, unit: millimeter
	mapCode	String	32	Req.	ID of map where the AGV locates

	speed	String	6	Req.	AGV current speed, unit: mm/s
	status	String	6	Req.	AGV status, see ***AGV Status*** for details.
	exclType	String	1	Req.	Whether the AGV is excluded: "1"-excluded, "0"-not excluded. **Note** No task will be assigned to the excluded AGV.
	stop	String	1	Req.	Whether the AGV is stopped: "0"-no, "1"-yes
	podCode	String	16	Opt.	Carried rack ID
	podDir	String	6	Opt.	Direction of move with rack
	path	String[ ]	300	Opt.	Task execution path, unit: millimeter, format: [x-coordinate, y-coordinate, direction], e.g., ["[x,y,dir]","[x,y,dir]", "[x,y,dir]"]
	message	String	64	Req.	Returned status description, e.g., "successful"
	reqCode	String	64	Req.	Request ID, which must be same with that in corresponding request message

Sample Codes

Request

```
{
 "reqCode": "1541954B96B1112",
 "reqTime": "2020-04-03 10:08:06",
 "mapShortName": "test"
}
```

Response

```
{
 "code": "0",
 "message": "successful",
 "reqCode": "1541954B96B1112",
 "data": [{
  "robotCode": "1001",
  "robotDir": "180",
  "battery": "80",
  "posX": "1.0",
  "posY": "2.0",
  "mapCode": "AA",
  "speed": "100",
  "status": "1",
  "exclType": "0",
  "stop": "1",
  "podCode": "200001",
  "podDir": "90",
  "path": [
   "[10000,20000,90]",
   "[20000,30000,-90]",
      "[20000,30000,180]",
      "[30000,40000,0]"
    ],
  },
  {
    "robotCode": "1001",
    "robotDir": "180",
    "battery": "80",
    "posX": "1.0",
    "posY": "2.0",
    "mapCode": "AA",
    "speed": "100",
    "status": "1",
    "exclType": "0",
    "stop": "1",
    "podCode": "200001",
    "podDir": "90",
  }]
}
```

5.2.12 stopRobot

Stop the specified AGV or all AGVs.

API Definition

Table 5-12 POST http://[address][:port]/rcms/services/rest/hikRpcService/stopRobot

<table>
<tr><td><b>API Name</b></td><td colspan="5">stopRobot</td></tr>
<tr><td><b>Function</b></td><td colspan="5">Stop the specified AGV or all AGVs.</td></tr>
<tr><td><b>Protocol</b></td><td colspan="5">REST</td></tr>
<tr><td><b>Provider</b></td><td colspan="5">RCS-2000</td></tr>
<tr><td><b>Caller</b></td><td colspan="5">Third-party platform</td></tr>
<tr><td rowspan="4"><b>Request</b></td><td><b>Parameter</b></td><td><b>Data Type</b></td><td><b>Max. Length</b></td><td><b>Req. or Opt.</b></td><td><b>Description</b></td></tr>
<tr><td>reqCode</td><td>String</td><td>32</td><td>Req.</td><td>Request ID, if a same request is repeatedly submitted, the request ID must be same.</td></tr>
<tr><td>reqTime</td><td>String</td><td>20</td><td>Opt.</td><td>Request timestamp, format: YYYY-MM-DD hh:mm:ss</td></tr>
<tr><td>clientCode</td><td>String</td><td>16</td><td>Opt.</td><td>Third-party platform ID, e.g., PDA, HCWMS</td></tr>
</table>

<table>
<tr><td></td><td>tokenCode</td><td>String</td><td>64</td><td>Opt.</td><td>Token ID, which is provided by RCS-2000 for third-party platform.</td></tr>
<tr><td></td><td>robotCount</td><td>String</td><td>64</td><td>Opt.</td><td>The number of AGVs to be stopped, "-1"-all AGVs.</td></tr>
<tr><td></td><td>mapShort Name</td><td>String</td><td>32</td><td>Opt.</td><td>Alias of the map where the AGV locates.
<b>Note</b>
This node is required when the value of <b>robotCount</b> is "-1".</td></tr>
<tr><td></td><td>robots</td><td>String[ ]</td><td>16</td><td>Opt.</td><td>List of AGV IDs.</td></tr>
<tr><td><b>Response</b></td><td>code</td><td>String</td><td>6</td><td>Req.</td><td>Status code, see <b><i>Status Code</i></b> for details.</td></tr>
<tr><td></td><td>message</td><td>String</td><td>64</td><td>Req.</td><td>Returned status description, e.g., "successful"</td></tr>
<tr><td></td><td>reqCode</td><td>String</td><td>64</td><td>Req.</td><td>Request ID, which must be same with that in corresponding request message</td></tr>
<tr><td><b>Sample Codes</b></td><td><b>Request</b></td><td colspan="4"><pre>{
  "reqCode": "1541954B96B1112",
  "robotCount": "2",
  "robots": [
    "1001",
    "1002"
  ]
}</pre></td></tr>
<tr><td></td><td><b>Response</b></td><td colspan="4"><pre>{
  "code": "0",
  "message": "successful",
  "reqCode": "1541954B96B1112"
}</pre></td></tr>
</table>

5.2.13 resumeRobot

Resume the AGV, the resumed AGV will continue executing the uncompleted task.

API Definition

Table 5-13 POST http://[address][:port]/rcms/services/rest/hikRpcService/resumeRobot

API Name	resumeRobot
Function	Resume the AGV, the resumed AGV will continue executing the uncompleted task.

Protocol	REST				
Provider	RCS-2000				
Caller	Third-party platform				
Request	**Parameter**	**Data Type**	**Max. Length**	**Req. or Opt.**	**Description**
	reqCode	String	32	Req.	Request ID, if a same request is repeatedly submitted, the request ID must be same.
	reqTime	String	20	Opt.	Request timestamp, format: YYYY-MM-DD hh:mm:ss
	clientCode	String	16	Opt.	Third-party platform ID, e.g., PDA, HCWMS
	tokenCode	String	64	Opt.	Token ID, which is provided by RCS-2000 for third-party platform.
	robotCount	String	64	Opt.	The number of AGVs to be resumed, "-1"-all AGVs.
	mapShortName	String	32	Opt.	Alias of the map where the AGV locates. **Note** This node is required when the value of **robotCount** is "-1".
	robots	String[]	16	Opt.	List of AGV IDs.
Response	code	String	6	Req.	Status code, see ***Status Code*** for details.
	message	String	64	Req.	Returned status description, e.g., "successful"
	reqCode	String	64	Req.	Request ID, which must be same with that in corresponding request message

Sample Codes	**Request**	{
		  "reqCode": "1541954B96B1112",
		  "robotCount": "2",
		  "robots": [
		    "1001",
		    "1002"
		  ]
		}			

	Response	{ "code": "0", "message": "successful",			

		"reqCode": "1541954B96B1112" }

5.2.14 setAreaState

Block or unblock the specified area.

API Definition

Table 5-14 POST http://[address][:port]/rcms/services/rest/hikRpcService/setAreaState

API Name	setAreaState				
Function	Block or unblock the specified area.				
Protocol	REST				
Provider	RCS-2000				
Caller	Third-party platform				
Remarks	**Note** • You can configure the area via RCS-2000 • When the area is blocked, the AGVs in the area will stop, and the AGVs to enter the area will bypass.				
Request	**Parameter**	**Data Type**	**Max. Length**	**Req. or Opt.**	**Description**
	reqCode	String	32	Req.	Request ID, if a same request is repeatedly submitted, the request ID must be same.
	reqTime	String	20	Opt.	Request timestamp, format: YYYY-MM-DD hh:mm:ss
	clientCode	String	16	Opt.	Third-party platform ID, e.g., PDA, HCWMS
	tokenCode	String	64	Opt.	Token ID, which is provided by RCS-2000 for third-party platform.
	matterArea	String	16	Req.	ID of the area to be blocked or unblocked
	indBind	String	1	Req.	Block or unblock: "1"-block, "0"-unblock
Response	code	String	6	Req.	Status code, see ***Status Code*** for details.
	message	String	64	Req.	Returned status description, e.g., "successful"

	reqCode	String	64	Req.	Request ID, which must be same with that in corresponding request message

Sample Codes	**Request**	{
		  "reqCode": "1541954B96B1112",
		  "matterArea": "2",
		  "indBind": "1"
		}			

	Response	{
		  "code": "0",
		  "message": "successful",
		  "reqCode": "1541954B96B1112"
		}			

5.3 API Provided by Third-Party

5.3.1 agvCallback

Send task executing status to the third-party platform.

API Definition

Table 5-15 POST http://[address][:port]/xxx/agv/agvCallbackService/agvCallback

API Name	agvCallback				
Function	Send task executing status to the third-party platform.				
Protocol	REST				
Provider	Third-party platform				
Caller	RCS-2000				
Remarks	**Note** The exhaustive request parameters are more than parameters listed in the following table, the third-party platform can choose required parameters according to the business.				
Request	**Parameter**	**Data Type**	**Max. Length**	**Req. or Opt.**	**Description**
	reqCode	String	32	Req.	Request ID, if a same request is repeatedly submitted, the request ID must be same.

	reqTime	String	20	Req.	Request timestamp, format: YYYY-MM-DD hh:mm:ss
	cooX	String	8	Opt.	X-coordinate of location code, unit: mm. This node exists only when the task has been completed.
	cooY	String	8	Opt.	Y-coordinate of location code, unit: mm. This node exists only when the task has been completed.
	currentPositionCode	String	32	Req.	Current position ID
	data	String	2000	Opt.	Custom content
	mapCode	String	16	Opt.	Map ID
	mapDataCode	String	32	Opt.	Location code on map, and it is unique. This node exists only when the task has been completed.
	method	String	16	Req.	Name to describe the task executing status: "start"-task started, "outbin"-executing, "end"-task completed, "cancel"-cancel task; it is provided by RCS-2000
	podCode	String	16	Opt.	Rack ID, this node exists only the rack is carried.
	podDir	String	4	Opt.	Rack directions: "180"-leftward, "0"-rightward, "90"-upward, "-90"-downward; this node exists only when the task has been completed.
	robotCode	String	16	Req.	AGV ID
	taskCode	String	64	Req.	Current task ID (UUID)
	wbCode	String	32	Opt.	Workstation ID, which consists of letters and digits, and it must be same with that configured by RCS-2000. This node exists only when the task has been completed, and it is same with parameter **wbCode** of API ***genAgvSchedulingTask*** .
Response	code	String	6	Req.	Status code, see ***Status Code*** for details.
	message	String	64	Req.	Returned status description, e.g., "successful"
	reqCode	String	64	Req.	Request ID, which must be same with that in corresponding request message

	data	String	2000	Opt.	Custom content to be returned, such as task ID

Sample Codes	**Request**	{
		  "reqCode": "1541954B96B1112",
		  "reqTime": "2019-04-03 10:08:06",
		  "cooX": "3000",
		  "cooY": "21999",
		  "currentPositionCode": "p02",
		  "mapCode": "AA",
		  "mapDataCode": "002069AA015172",
		  "method": "end",
		  "podCode": "100001",
		  "robotCode": "6001",
		  "taskCode": "test169E0F39740116Q",
		  "wbCode": "p02"
		}			

	Response	{
		  "code": "0",
		  "message": "successful",
		  "reqCode": "1541954B96B1112",
		  "data":""
		}			

5.3.2 warnCallback

RCS-2000 sends the severe alarm which causes AGV stopping to the third-party platform. The alarm sending frequency is 10 seconds per time.

API Definition

Table 5-16 POST http://[address][:port]/service/rest/agvCallbackService/warnCallback

API Name	warnCallback
Function	RCS-2000 sends the severe alarm which causes AGV stopping to the third-party platform.
Protocol	REST
Provider	Third-party platform
Caller	RCS-2000
Remarks	**Note** The request URL should be "http://[address][:port]/service/rest/agvCallbackService/warnCallback", of which the "http://[address][:port]/service/rest" can be configured

	in the system parameters configuration of RCS-2000, and the corresponding configuration numbers are: 10012, 10013, and 10014.				
Request	**Parameter**	**Data Type**	**Max. Length**	**Req. or Opt.**	**Description**
	reqCode	String	32	Req.	Request ID, if a same request is repeatedly submitted, the request ID must be same.
	reqTime	String	20	Opt.	Request timestamp, format: YYYY-MM-DD hh:mm:ss
	clientCode	String	16	Opt.	Third-party platform ID, e.g., PDA, HCWMS
	tokenCode	String	64	Opt.	Token ID, which is provided by RCS-2000 for third-party platform.
	data	Object			
	robotCode	String	16	Req.	AGV ID
	beginTime	String	64	Req.	Alarm start time
	warnContent	String	64	Req.	Alarm content
	taskCode	String	64	Opt.	Task No.
Response	code	String	6	Req.	Status code, see ***Status Code*** for details.
	message	String	64	Req.	Returned status description, e.g., "successful"
	reqCode	String	64	Req.	Request ID, if a same request is repeatedly submitted, the request ID must be same.

Sample Codes	**Request**	{
		 "reqCode": "1541954B96B1112",
		 "data:": [{
		  "robotCode": "1001",
		  "beginTime": "2020-04-02 23:12:12",
		  "warnContent": "Platform disconnected",
		  "taskCode": "C002WWQQRR"
		 },
		 {
		  "robotCode": "1002",
		  "beginTime": "2020-04-02 23:12:12",
		  "warnContent": "Guidance alarm",
		  "taskCode": "C002WWQQRR33"				

		}] }

	Response	{
		"code": "0",
		"message": "successful",
		"reqCode": "1541954B96B1112"
		}

Appendix A. Appendixes

A.1 Status Code

The status code returned in the response message are defined in the table below.

Status Code Description

code	Description
0	Succeeded.
1	Incorrect parameters.
6	No need to resend (the task of the same **reqCode** is not completed).
99	Unknown error, you can try again.
100	The task does not exist.

A.2 AGV Status

The AGV common statuses are list in the table below.

AGV Common Status

status	Description
1	Task completed
2	Executing task
3	Abnormal task
4	Idle task
5	Robot stopped
6	Lifting shelf status
7	Charging status
8	Battery arcing in progress
9	Fully charged, entering maintenance mode
11	Carried item not recognized
12	Excessive shelf angle divergence

status	Description
13	Motion library exception
14	Unable to recognize product code
15	Product code mismatch
16	Lift abnormal
17	Charging post abnormal
18	No increase in current
20	Angle error in charging directive
21	Platform decentralisation directive error
23	External force, unloading
24	Misaligned shelf
25	Trolley not in designated zone
26	Decentralisation failed
27	Uneven shelf
28	Lift battery current too low
29	Wide reversing angle
30	No rack detected
31	Failed to lock zone
33	Rotation request temporarily failed
34	Unable to recognize coordinates to switch maps

See Far, Go Further