RCS-2000 API

Developer Guide

Legal Information

Contents

Chapter 1 Overview

1.1 Introduction

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

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

1.2 Update History

Summary of Changes in Version 3.2.1_April/2023

Version	Summary of Changes
Version 3.2.1_April/2023	1. Extended the API ***genAgvSchedulingTask*** : extended the applicable AMR types; extended the description of a request parameter: **materialLot** (material lot ID); added 2 request parameters: **agvTyp** (device type) and **positionSelStrategy** (in and out rules for searching for racks).
	2. Deleted the API for specifying the container destination: **bindCtnrDestination**.
	3. Added the API for checking whether the container inbound at loader/unloader is allowed: ***applyReturnForValid (Tower Station)*** .
	4. Deleted the API (applyBin) and added the API for applying for inbound position of the container: ***applyReturnForBin (Tower Station)*** .
	5. Extended the API ***genPreScheduleTask*** : added request messages for three different application scenarios: for container inbound at conveyor line via tower station (apply a pre-schedule task every time a container is scanned), for container inbound at conveyor line via tower station (update the total number of pre-

Version	Summary of Changes
	schedule tasks every time a container is scanned), and for container inbound at loader/unloader.
	6. Added the calling flow for pre-schedule and inbound via tower station at the conveyor line: ***Pre-Schedule and Inbound via Tower Station at Conveyor Line*** .
	7. Added the calling flow for inbound via tower station at the loader/unloader: ***Inbound via Tower Station at Loader/Unloader*** .
	8. Added more tower station-related applications: ***Other Tower Station Application Scenarios*** .

Summary of Changes in Version 3.2.0_September/2022

Version	Summary of Changes
Version 3.2.0_September/2022	1. Extended API ***genAgvSchedulingTask*** : added three fields: **ctnrNum** (number of containers), **taskMode** (task mode), and **groupId** (group ID, which is generally used for tower station outbound by groups).
	2. Added the API for applying pre-schedule tasks: ***genPreScheduleTask*** .
	3. Added the API for clearing all the data of roadways, containers, or racks: ***clearRoadWay*** .
	4. Added the API for batch enabling/disabling bins: ***blockStgBin*** .
	5. Extended the API ***agvCallback*** : added four fields: **stgBinCode** (storage bin ID), **ctnrCode** (container No.), **ctnrTyp** (container type), and **eqpCode** (equipment No.) for uploading the data of container carrying; added three fields: **materialLot** (material No.), **roadWayCode** (roadway No.), and **seq** (sequence No. in the roadway) for uploading the roadway information.
	6. Extended the API ***bindNotify*** : added the field **binParam** for uploading the information about binding/unbinding operations
	7. Added APIs related to tower station business. See and the sub chapters for details.

Summary of Changes in Version 3.1.4_May/2021

Version	Summary of Changes
Version 3.1.4_May/2021	1. Edited the request message of API ***genAgvSchedulingTask*** : deleted one field **sceneTyp**; added 3 location types to the field **type** of **positionCodePath**: "07" (container ID), "08" (roadway strategy), and "09" (roadway area).
	2. Edited the request message of API ***bindPodAndBerth*** : added one field **characterValue** (trait value).
	3. Added one API for binding or unbinding a container and a bin: ***bindCtnrAndBin*** .
	4. Added one API for emptying or unblocking the specified area: ***blockArea*** .
	5. Added one API for sending the pre-schedule: ***genPreScheduleTask*** .
	6. Added one API for notifying the third-party platform of the binding or unbinding operation: ***bindNotify*** .

Summary of Changes in Version 3.1.0_July/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 FMR 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 for stopping the specified AMR or all AMRs: ***stopRobot*** .
	7. Added one API for resuming the AMR: ***resumeRobot*** .

Version	Summary of Changes
	8. Added one API for blocking or unblocking the area: .
	9. Edited the API for searching for the AMR status ***queryAgvStatus*** : edited the request URL to "http://[address]:8083/rcms-dps/rest/queryAgvStatus".

Summary of Changes in Version 2.5_May/2020

Version	Summary of Changes
Version 2.5_May/2020	1. Edited the API absolute address to "/cms/services/rest/hikRpcService/[apiName]".
	2. Deleted one API for sending data synchronization requirement to the third-party platform when the information has changed: syncNotify.

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.

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 AMR status, the request URL is "http://[address][:port]/rcms-dps/rest/queryAgvStatus".

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 **Note** The default web server port No. is 8182.
RCS-2000 (AMR status search)	http://IP:PORT/rcms-dps/rest

Platform	baseURL
	Note The default port No. is 8182.

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 fields in the message are named by lower camel case.

JSON format is a subset of JavaScript, 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. See ***Status Code*** (Appendix. A) for detailed error codes and description.

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 Hikrobot system and will be transmitted to the third-party platform for authentication when calling APIs.

Chapter 4 Typical Application

4.1 AMR Carries Rack and Leaves without Rack

In this application scene, the AMR 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 AMR 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

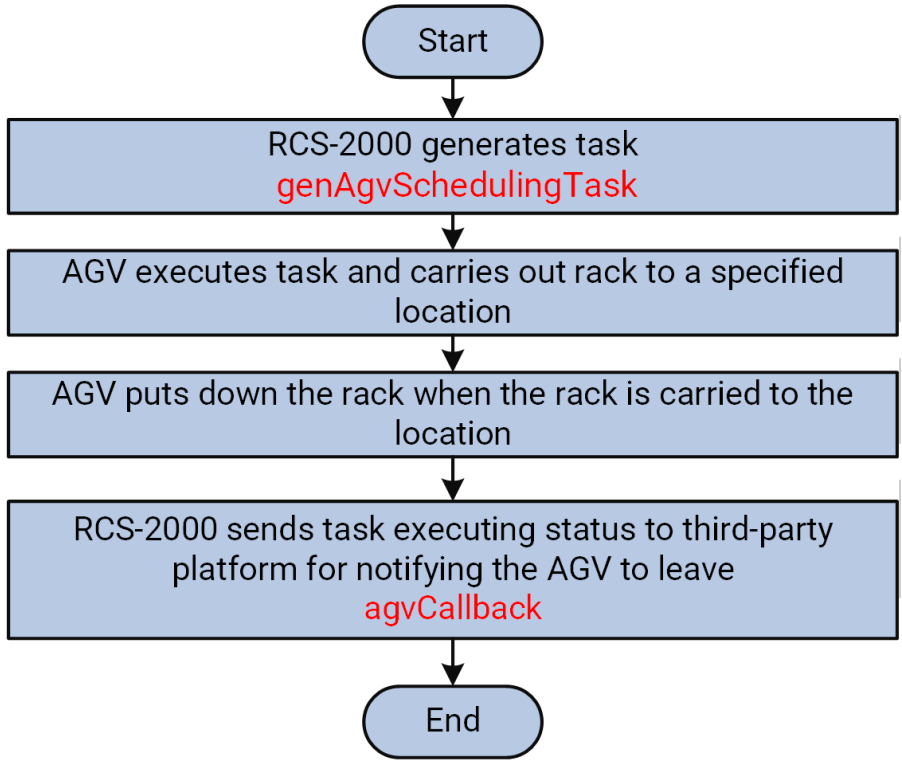

Figure 4-1 Flow of AMR Carrying Rack and Leaving without Rack

1. The third-party platform calls ***genAgvSchedulingTask*** to generate task via the RCS-2000.
   The AMR starts executing the task and carrying a rack to a specified location.
2. The AMR 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 AMR to leave.

4.2 AMR Carries Rack and Backs with Rack

In this application scene, the AMR 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 AMR keeps carrying the rack and waits until the third-party platform call an API to continue the task. And then the AMR 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

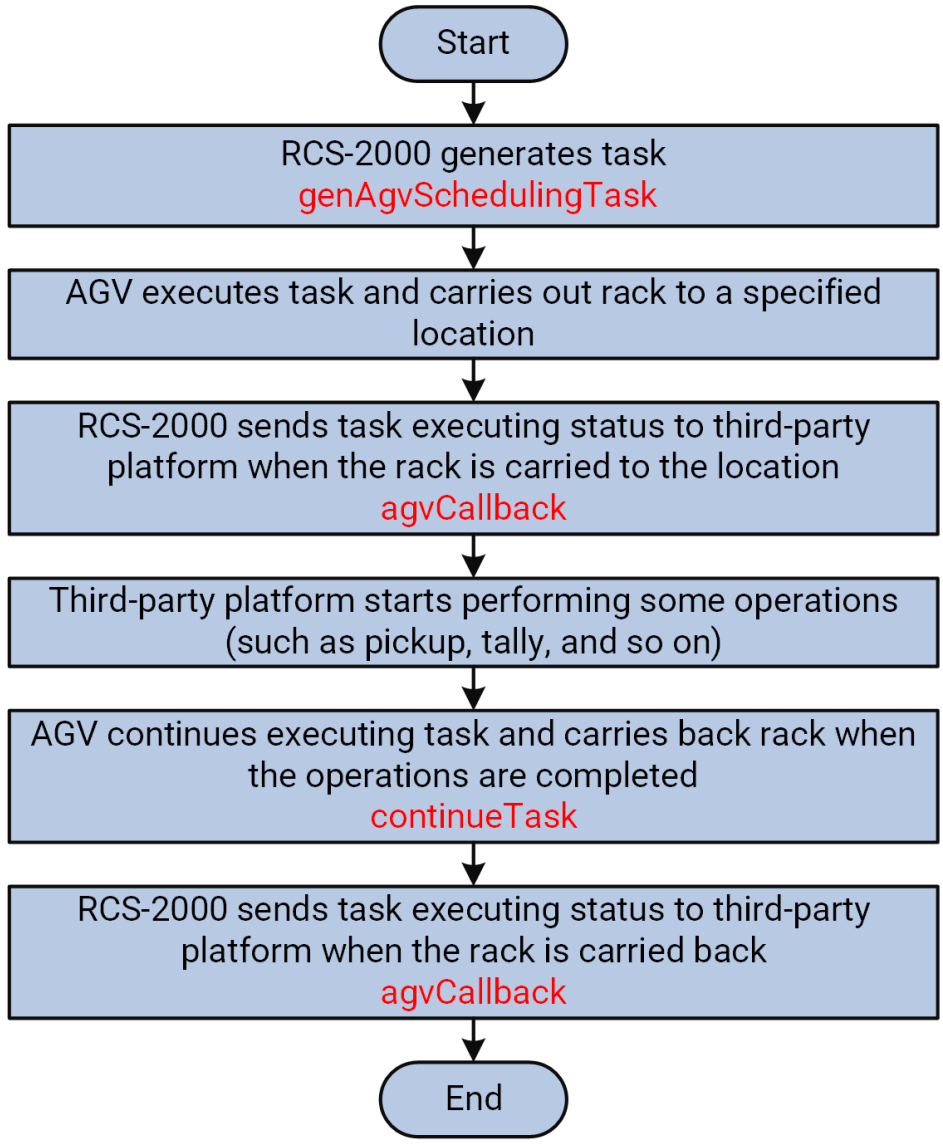

Figure 4-2 Flow of AMR Carrying Rack and Backing with Rack

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

The AMR 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 AMR 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 AMR Receives Materials

The roller AMR 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 FMR, it can convey very heavy material. Here introduces the progress of controlling roller AMR to complete the task of receiving materials.

Steps

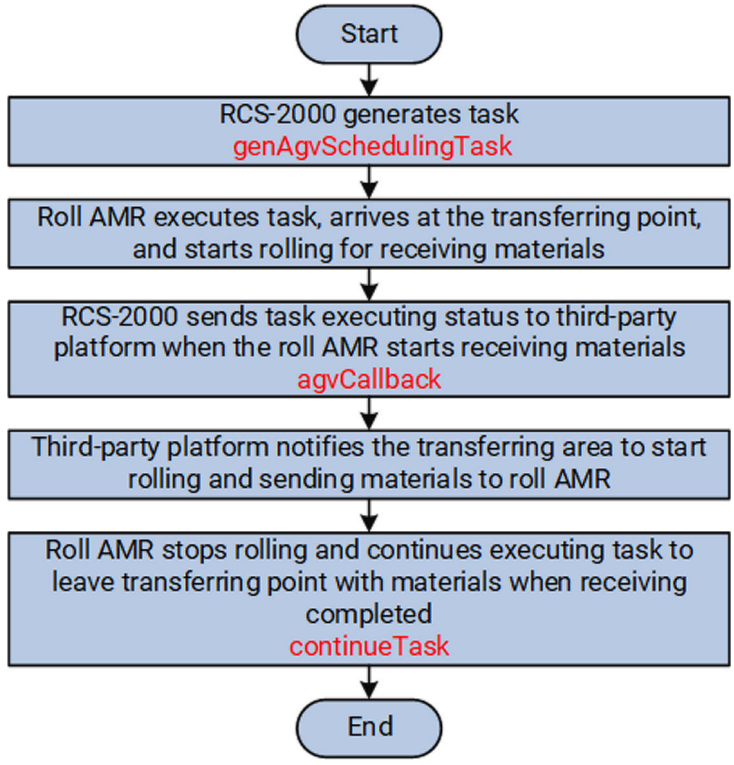

Figure 4-3 Flow of Roller AMR Receiving Materials

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

The roller AMR 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 AMR is ready for receiving materials.
3. The third-party platform notifies the transferring area to start rolling and sending materials to the roller AMR.
4. The third-party platform calls ***continueTask*** to continue executing task when the materials is received by roller AMR..

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

4.4 Pre-Schedule and Inbound via Tower Station at Conveyor Line

Ahead of container inbound to the conveyor line, the third-party platform applies the pre-schedule task and the AMR will be sent by the calling system and be ready for the containers. When the containers arrive at the conveyor line, the inbound task will be applied.

Steps

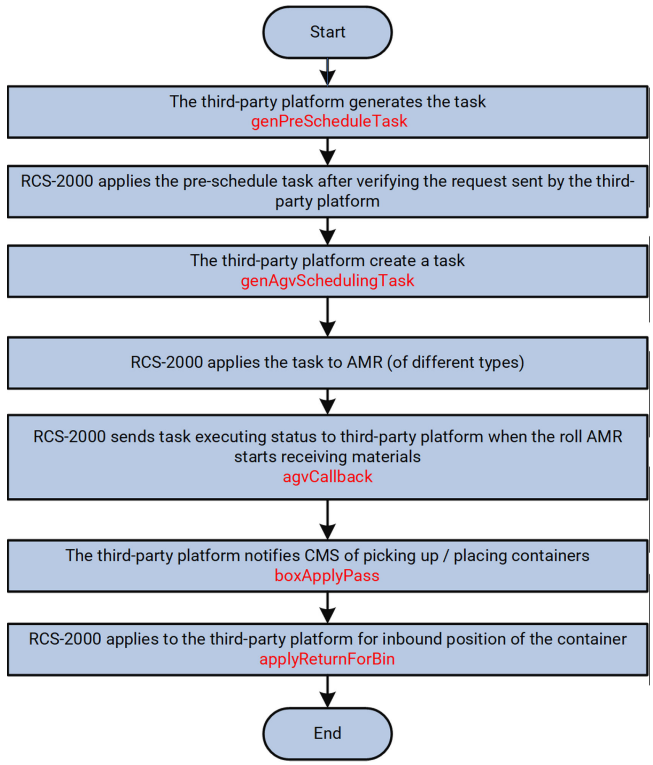

Figure 4-4 Flow of Pre-Schedule and Inbound via Tower Station at Conveyor Line

1. The third-party platform calls ***genPreScheduleTask*** to generate the pre-schedule task and the RCS-2000 applies the pre-schedule task after verifying the request sent by the third-party platform.

2. The third-party platform calls ***genAgvSchedulingTask*** to create a task and the RCS-2000 applies the task to AMR (of different types).

   Note

   The container destination is not obtained in this step. See in ***applyReturnForBin (Tower Station)*** .

3. The RCS-2000 calls ***agvCallback*** to send the task executing status (task starting, getting out of storage section, task canceling, task ending, etc.) to the third-party platform.
4. The third-party platform calls ***boxApplyPass (Tower Station)*** to notify CMS of picking up / placing containers.
5. The RCS-2000 calls ***applyReturnForBin (Tower Station)*** to apply to the third-party platform for inbound position of the container.

4.5 Inbound via Tower Station at Loader/Unloader

For each container arriving at the loader/unloader, the RCS-2000 needs to check with the third-party platform whether the inbound is allowed. Once it is allowed, the container enters the loader/unloader and waits for inbound. When there are a certain number of containers waiting at the loader/unloader, the RCS-2000 will batch apply to the third-party platform for inbound position of the containers and the tower station will execute the inbound task.

Steps

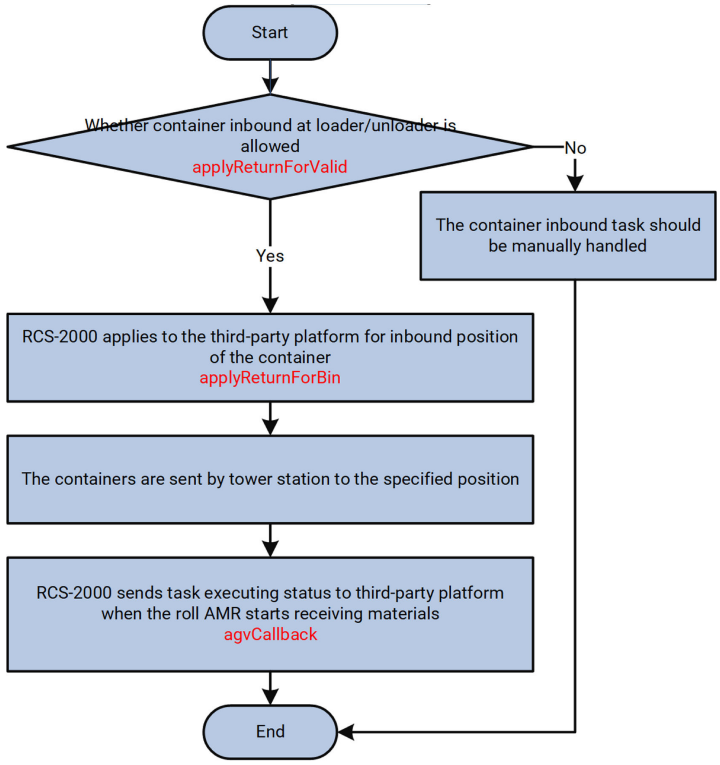

Figure 4-5 Flow of Tower Station Loader/Unloader Inbound

1. The RCS-2000 calls ***applyReturnForValid (Tower Station)*** to check with the third-party platform whether the container inbound at the loader/unloader is allowed.
2. If it is not allowed (usually due to the container problem), the containers will not enter the loader/unloader and should be instead manually handled; if yes, the RCS-2000 calls ***applyReturnForBin (Tower Station)*** to apply to the third-party platform for inbound position of the container.
3. After the inbound position is obtained, the containers will be sent by tower station to the specified position.
4. The RCS-2000 calls ***agvCallback*** to send the task executing status to the third-party platform.

4.6 Other Tower Station Application Scenarios

Other typical application scenarios are as follows:

Scenario	API
1. Container inbound/outbound at buffer rack picking station.	***getOutPod (Tower Station / Buffer Rack Picking Station)*** ; ***returnPod (Tower Station / Buffer Rack Picking Station)*** .
2. Batch/single outbound at conveyor line or loader/unloader.	***genPreScheduleTask*** .
3. Outbound at conveyor line or loader/ unloader by order.	***genCtuGroupTaskBatch (Tower Station)*** .
4. Inbound at conveyor line.	***genPreScheduleTask*** ; ***applyReturnForBin (Tower Station)*** .
5. Inbound at loader/unloader.	***applyReturnForValid (Tower Station)*** ; ***applyReturnForBin (Tower Station)*** .
6. Apply for container picking-up/placing and get task execution notification.	***agvCallback*** .
7. Notify CMS of picking up / placing containers.	***boxApplyPass (Tower Station)*** .

Chapter 5 API Reference

5.1 API List

List of Commonly Used APIs

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

List of Optional APIs

Function	API Name	Provider
Set the task priority.	***setTaskPriority***	RCS-2000
Bind and unbind the rack and location.	***bindPodAndBerth***	RCS-2000
Bind and unbind the 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 the task executing status.	***queryTaskStatus***	RCS-2000
Search for the AMR status.	***queryAgvStatus***	RCS-2000
Stop AMR(s).	***stopRobot***	RCS-2000
Resume a AMR.	***resumeRobot***	RCS-2000
Bind or unbind the container and bin.	***bindCtnrAndBin***	RCS-2000
Empty or unblock the specified area.	***blockArea***	RCS-2000
Send the pre-schedule.	***genPreScheduleTask***	RCS-2000
Clear all the data and notify RCS of updating.	***clearRoadWay***	RCS-2000

Function	API Name	Provider
Container outbound, used for the buffer rack picking station.	***getOutPod (Tower Station / Buffer Rack Picking Station)***	RCS-2000
Container inbound.	***returnPod (Tower Station / Buffer Rack Picking Station)***	RCS-2000
Apply and perform tower station outbound tasks in order.	***genCtuGroupTaskBatch (Tower Station)***	RCS-2000
Container picking-up/placing callback.	***boxApplyPass (Tower Station)***	RCS-2000
Check with the third-party platform whether the container inbound at the loader/unloader is allowed.	***applyReturnForValid (Tower Station)***	Third-party Platform
Apply to the third-party platform for inbound position of the container.	***applyReturnForBin (Tower Station)***	Third-party Platform
Send the alarm to the third-party platform.	***warnCallback***	Third-party Platform
Notify the third-party platform of the binding or unbinding operation.	***bindNotify***	Third-party Platform

5.2 API Provided by RCS-2000

5.2.1 genAgvSchedulingTask

The third-party platform creates a task and Robot Control System (RCS-2000) applies the task to AMR (of different types).

API Definition

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

API Name	genAgvSchedulingTask
Function	The third-party platform creates a task and Robot Control System (RCS-2000) applies the task to AMR.
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 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 type, which is the same as the major task type configured in RCS-2000: • Build-in task type: "F01"-carry and transfer rack, "F02"-empty/full rack exchange, "F03"-carry and transfer by CMR, "F04"-rack outbound, "F05"-rotate rack, "F06"-elevator task. • FMR task type: "F11"-transfer from high bay rack to workstation, "F12"-transfer from workstation to high bay rack, "F13"-transfer from roadway to workstation, "F14"-transfer from workstation to roadway, "F15"-shuttle from high bay rack to workstation, "F16"-shuttle from workstation to high bay rack, "F17"-shuttle from roadway to workstation, "F18"-shuttle from workstation to roadway, "F20"-cross-floor FMR main task.
	ctnrTyp	String	16	Opt.	Container type. This field is dedicated for FMR tasks and tower station tasks, and it is required for FMR tasks.

ctnrCode	String	32	Opt.	Container ID. This field is dedicated for FMR tasks and tower station tasks.
ctnrNum	String	2	Opt.	Number of containers. This field is dedicated for FMR tasks, and it is used in the scenario of palletizing/depalletizing containers via FMR.
taskMode	String	1	Opt.	Task mode: 0 (normal move), 1 (outbound move), 2 (inbound move), 3 (transfer move). The outbound move cannot be interrupted, while the normal move and inbound move mode can be interrupted. When an outbound move is completed, the inbound move or transfer move should be performed.
wbCode	String	32	Opt.	Workstation ID, which consists of letters and digits, and it must be same as that configured by RCS-2000.
positionCodePath	Object[]	50	Opt.	Rack moving pattern, which consists of multiple locations (up to 50 locations are allowed).
⊢ type	String	/	Opt.	Location type: "00"-actual location on map, "01"-location of a specific material batch, "02"-available location of area selection strategy, "03"-rack No., "04"-available location in an area, "05"-bin ID (for FMR / tower station only), "06"-roadway ID, "07"-container ID, "08"-roadway strategy, "09"-roadway area, "10"-roadway bin, "11"-conveyor (operating machine) No., "12"-tower station workstation (loader/unloader) No., "13"-position of roadway rack (used for rack outbound of the roadway).

⊢ positionCode	String	/	Opt.	Location ID, which is predefined with detailed coordinate information on map. It depends on the value of **type**.
podCode	String	16	Opt.	Rack ID, it can be empty if no rack specified.
podDir	String	4	Opt.	Rack direction: "180"-leftward, "0"-rightward, "90"-upward, "-90"-downward; it can be empty if no direction specified. If the destination is not the workstation, this field will be the direction of task destination.
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. Rack type No.: empty rack of the specified type.
materialLot	String	32	Opt.	Material lot ID. You can set **materialLot** and **podCode** at same time to bind the rack with material lot, or you can set **materialLot** and **wbCode** to bind the material lot with the rack at the specified workstation. If you are searching for racks by area/strategy, you can locate the rack with the bound material lot: 0-search for empty racks by area/strategy, 1-search for full racks by area/strategy, other values-search for racks by specified material lots. For roadway tasks, this field is used for configuring the material trait value.
priority	String	3	Opt.	Task priority, range: [1,127], and larger number corresponds to higher priority. If it is not configured, the task template priority takes effect.
agvCode	String	5	Opt.	AMR ID, if it is not configured, the RCS-2000 will automatically select an optimal AMR to execute the task.

	taskCode	String	64	Opt.	Task ID, if it is not configured, the system will generate automatically.
	groupId	String	16	Opt.	Group ID, which is generally used for tower station outbound by groups. Tasks of a same group have higher priority for sharing AMR. If the business requires outbound in order, you should call ***genCtuGroupTaskBatch (Tower Station)*** to perform outbound in order. For LMR (latent mobile robot), the outbound order is according to the group ID: tasks of smaller group ID have higher priority for outbound.
	agvTyp	String	4	Opt.	Device type.
	positionSelStrategy	String	2	Opt.	The in and out rules for searching for racks by area / strategy / material lot: "1" (first-in-first-out by the time the rack arrives at the storage section), "2" (first-in-last-out by the time the rack arrives at the storage section), "9" (first-in-first-out by the time the rack is bound with the material lot), "10" (first-in-last-out by the time the rack is bound with the material lot).
	data	String	2000	Opt.	Custom content in JSON format.
Response	code	String	6	Req.	Status code, see ***Status Code*** (Appendix A) for details.
	message	String	64	Req.	Returned status description, e.g., "successful".
	reqCode	String	64	Req.	Request ID, which is the same as 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 field **positionCodePath** should be configured.				

Sample	**Request**	{
		"reqCode": "468513",
		"taskTyp": "F01",
		"positionCodePath": [{
		"positionCode": "p01",
		"type": "00"			

		<pre>{
		  },
		  {
		  "positionCode": "x02",
		  "type": "02"
		  }],
		  "podCode": "100001",
		  "podDir": "0",
		  "priority": "1"
		}</pre>

	Response	<pre>{
		  "code": "0",
		  "data": "F01169C808C317111G",
		  "message": "successful",
		  "reqCode": "468513"
		}</pre>

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 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. According to the input rack ID, the RCS-2000 will find the corresponding task and continue executing the next sub task.
	podCode	String	16	Opt.	Rack ID. According to the input rack ID, the RCS-2000 will find the corresponding task and continue executing the next sub task.
	agvCode	String	5	Opt.	AMR ID. According to the input AMR ID, the RCS-2000 will find the corresponding task and continue executing the next sub task.
	taskCode	String	64	Opt.	Task ID (UUID). According to the input task ID, the RCS-2000 will find the corresponding task and continue executing the next sub task.
	taskSeq	String	32	Opt.	Sub task No. to be specified to continue executing. If this field is not configured, by default, the next sub task will be executed.
	nextPositionCode	Object	40	Opt.	The location information of sub tasks. This field is required when the task type is externally configured. If this field is not configured, it indicates to perform the next sub task by default.
	⊢ positionCode	String	/	Opt.	Location ID, which is predefined with detailed coordinate information on map. It depends on the value of **type**.
	⊢ type	String	/	Opt.	Location types: "00"-actual location on map, "02"-strategy
Response	code	String	6	Req.	Status code, see ***Status Code*** (Appendix A) for details.

	message	String	64	Req.	Returned status description, e.g., "successful"
	reqCode	String	64	Req.	Request ID, which is the same as that in corresponding request message
Remarks	Only one of **wbCode**, **agvCode**, **taskCode**, and **podCode** should be configured at one time, and the parameter to be configured depends on the trigger type configured in the task template. The parameter **taskCode** is recommended.				

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 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)-AMR puts down the rack at current location and its status turns to "idle", "1"-AMR carries the rack and returns back, this mode is supported by LMR (latent mobile robot) and tower station only.
	matterArea	String	16	Opt.	Inbound area No., it is valid only when the value of **forceCancel** is "1"; if this field is not configured, the inbound area is the storage area.
	agvCode	String	5	Opt.	AMR 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*** (Appendix A) for details.
	message	String	64	Req.	Returned status description, e.g., "successful".
	reqCode	String	64	Req.	Request ID, which is the same as that in corresponding request message
Remarks	One of **taskCode** and **agvCode** must be configured to confirm the task to be canceled or the AMR 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

- For canceling tasks by task ID, if an AMR is executing a task and carrying a rack, when the cancel mode is "0", the AMR will put down the rack at any location, and its status turns to "idle"; when the cancel mode is "1", the AMR will carry the rack and execute inbound, if there is no space of inbound area, the error will be returned and canceling failed.
- For canceling tasks by task ID, if a tower station is executing a task and loading a container, when the cancel mode is "0", the container will remain on the tower station and should be carried away manually; when the cancel mode is "1", the tower station will carry the container to the specified bin or empty bin.
- FMR only supports the cancel mode "0".

5.2.4 setTaskPriority

Set the task priority from 1 to 127, and the larger the value, the higher the priority. The priority takes effect only when the number of AMRs is less than that of tasks and there are multiple tasks with different priorities. The tasks will be assigned to the AMR according to priority order. The task priority may influence the route scheduling, when there is route conflict, the AMR with high-priority task uses the route.

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 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	Object[]	/	Req.	Task 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*** (Appendix A) for details.
	message	String	64	Req.	Returned status description, e.g., "successful"
	reqCode	String	64	Req.	Request ID, which is the same as that in corresponding request message
Remarks	You can set the priority for the tasks that are not assigned to the AMR only; If the task has been assigned to AMR, 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 or unbind the rack and location. When the binding is completed, the location can be searched by rack ID, which is recorded on the position. After unbinding the rack from the location, the rack ID will be cleared from the position.

API Definition

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

API Name	bindPodAndBerth				
Function	Bind or unbind a rack and one location.				
Protocol	REST				
Provider	RCS-2000				
Caller	Third-party platform				
Remarks	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 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 field is not configured, the rack direction is horizontal by default. The position configuration is in priority. If the position direction is configured as horizontal

					and vertical, this field is invalid; if the position direction is configured as all-direction, this field takes effect.
	characterValue	String	64	Opt.	Trait value (only for racks on the roadway). It can be the material type or batch information.
	indBind	String	1	Req.	Bind or unbind: "1"-bind, "0"-unbind.
Response	code	String	6	Req.	Status code, see ***Status Code*** (Appendix A) for details.
	message	String	64	Req.	Returned status description, e.g., "successful".
	reqCode	String	64	Req.	Request ID, which is the same as 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 bindCtnrAndBin

Bind or unbind the container and bin. For tower station (carton transport unit) and FMR tasks, you can call this API to write in the container type and ID to the bin.

Request URL Definition

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

API Name	bindCtnrAndBin
Function	Bind or unbind the container and bin.
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 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.
	ctnrCode	String	30	Req.	Container ID
	ctnrTyp	String	16	Req.	Container type
	stgBinCode	String	32	Opt.	Bin ID. **Note** One of **stgBinCode** and **positionCode** is required.
	binName	String	32	Opt.	Custom bin ID, which will be verified by the RCS-2000. For example, when the system configuration number 10110 is set to true, it indicates that the custom bin ID and bin ID should be one-to-one, if one custom bin ID corresponds to multiple bin ID, an error will be reported.
	characterValue	String	64	Opt.	Trait value (only for the FMR roadway)
	positionCode	String	32	Req.	Position ID, which is predefined with detailed coordinate information on the map, and it is a unique string configured via RCS-2000. This field is used for binding or unbinding containers and the bin of virtual racks. **Note** One of **stgBinCode** and **positionCode** is required.

	indBind	String	1	Req.	Bind or unbind: "1"-bind, "0"-unbind.
Response	code	String	6	Req.	Status code, see ***Status Code*** (Appendix A) for details.
	data	String	2000	Opt.	Custom content to be returned.
	message	String	64	Req.	Returned status description, e.g., "successful".
	reqCode	String	64	Req.	Request ID, which is the same as that in corresponding request message

Sample Codes	**Request**	{
		"reqCode": "12345678",
		"ctnrTyp": "C1",
		"stgBinCode": "p05",
		"indBind": "1"
		}			

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

5.2.7 genPreScheduleTask

Robot Control System (RCS) applies the pre-schedule task after verifying the request sent by the third-party platform. The pre-schedule task is to notify RCS of the actual task in advance so that RCS can send a free AMR to the target position and the AMR is in standby for the upcoming actual task. It improves the scheduling efficiency.

API Definition

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

API Name	genPreScheduleTask
Function	Robot Control System (RCS) applies the pre-schedule task after verifying the request sent by the third-party platform.
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 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.	Task position.
	nextTask	String	32	Req.	Pre-schedule task time period. Remaining seconds before the actual task is generated: "-1" (clear the pre-schedule tasks of a position).
	agvTyp	String	256	Req.	AMR type, with which the complete the actual task.
	waitTime	String	32	Opt.	Waiting time.
	priority	String	3	Opt.	Task priority, range: [1,127], and larger number corresponds to higher priority. If it is not configured, the task template priority takes effect.
	useableLayers	String	/	Opt.	Number of available empty bins (for the tower station only) on an AMR. When docking with the conveyor line, the tower station with one empty bin can be dispatched; when docking with the loader/unloader, only the empty tower station with the same number of bins can be dispatched.
	cacheCount	String	/	Opt.	Number of cache containers (for the loader/unloader and conveyor line). The number of pre-schedule tasks can be calculated via the value of **useableLayers** and **cacheCount**. And the calculation result has higher priority than **preTaskQty**.
	update	Number		Opt.	Whether to update the pre-schedule tasks:

					0 (each time add a new pre-schedule task to the total number of tasks) 1 (refresh and calculate the total number of pre-schedule tasks by **cacheCount** and **useableLayers**).
	preTaskQty	Number	/	Opt.	Number of pre-schedule tasks. When the value of **update** is 1, the number of pre-schedule tasks will be updated to the value of this filed.
Response	code	String	6	Req.	Status code, see ***Status Code*** (Appendix A) for details.
	message	String	64	Req.	Returned status description, e.g., "successful".
	reqCode	String	64	Req.	Request ID, which is the same as that in corresponding request message

Sample	**Request**	For container inbound at conveyor line via tower station (apply a pre-schedule task every time a container is scanned):
		{
		"reqCode": "123",
		"positionCode":"pos1", //The pre-schedule task position/bin.
		"nextTask": "300", //The actual task is expected to be applied within 5 minutes. If not, the pre-schedule AMR will be free.
		"agvTyp": "1",
		"priority": "",
		"useableLayers": "1", //Each pre-schedule corresponds to a container and requires a tower station bin.
		"cacheCount": "1", //Each pre-schedule corresponds to a container
		"update": "0" //Each time add a new pre-schedule task to the total number of tasks.
		}
		For container inbound at conveyor line via tower station (update the total number of pre-schedule tasks every time a container is scanned):
		{
		"reqCode": "123",
		"positionCode": "pos1", //The pre-schedule task position/bin.
		"nextTask": "300", //The actual task is expected to be applied within 5 minutes. If not, the pre-schedule AMR will be free.
		"agvTyp": "1",
		"priority": "",
		"useableLayers": "1", //Each pre-schedule corresponds to a container and requires a tower station bin.
		"cacheCount": "4", //4 containers currently at the conveyor line, which requires 4 pre-schedule tasks.
		"update": "1", //Refresh and calculate the total number of pre-schedule tasks				

		(cacheCount/useableLayers). }

		For container inbound at loader/unloader:
		{
		"reqCode": "123",
		"positionCode": "pos1", //The pre-schedule task position/bin.
		"nextTask": "300", //The actual task is expected to be applied within 5 minutes. If not, the pre-schedule AMR will be free.
		"agvTyp": "1",
		"priority": "",
		"useableLayers": "6", //Number of available empty bins (for the tower station only)docking with loader/unloader.
		"cacheCount": "12", //12 cached containers requiring loader/unloader inbound.
		"update": "1", //Refresh and calculate the total number of pre-schedule tasks (cacheCount/useableLayers).
		}

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

5.2.8 clearRoadWay

Clear all the data of roadways, containers, or racks and notify RCS of updating the information. Transport roadway: cannot be cleared when the outbound lock capacity is not 0; when the outbound capacity is set to 0, the inbound capacity will be recalculated. FMR roadway: cannot be cleared when the outbound and inbound lock capacity is not 0; when the configuration is completed, the capacity will be recalculated.

API Definition

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

API Name	clearRoadWay
Function	Clear all the roadway data and notify RCS of updating roadway information.
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 request is submitted repeatedly, the request ID must be the 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.
	roadWayCode	string	16	Req.	Roadway ID.
Response	code	string	6	Req.	Status code, see ***Status Code*** (Appendix A) for details.
	message	string	64	Req.	Returned status description.
	reqCode	string	64	Req.	Request ID.

5.2.9 lockPosition

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

API Definition

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

API Name	lockPosition				
Function	Enable or disable the position. 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 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*** (Appendix A) for details.
	message	String	64	Req.	Returned status description, e.g., "successful".
	reqCode	String	64	Req.	Request ID, which is the same as 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"
		}			

5.2.10 stopRobot

Stop the specified AMR or all AMRs.

API Definition

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

API Name	stopRobot
Function	Stop the specified AMR or all AMRs.

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 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 AMRs to be stopped: "-1" (all AMRs). **Note** • One of **robotCount** and **robots** is required. • To stop all AMRs, set **robotCount** to "-1", while to stop parts of AMRs, input the corresponding AMR ID list in **robots**.
	mapShortName	String	32	Opt.	Alias of the map where the AMR locates. **Note** This field is required when the value of **robotCount** is "-1".
	robots	String[]	16	Opt.	List of AMR IDs. **Note** • One of **robotCount** and **robots** is required. • To stop all AMRs, set **robotCount** to "-1", while to stop parts of AMRs, input the corresponding AMR ID list in **robots**.
Response	code	String	6	Req.	Status code, see ***Status Code*** (Appendix A) for details.

	message	String	64	Req.	Returned status description, e.g., "successful".
	reqCode	String	64	Req.	Request ID, which is the same as that in corresponding request message

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

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

5.2.11 syncMapDatas

Synchronize map information with that in RCS-2000.

API Definition

Table 5-11 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 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 field is not configured, all location codes of the map will be synchronized.
Response	code	String	6	Req.	Status code, see ***Status Code*** (Appendix A) for details.
	message	String	64	Req.	Returned status description, e.g., "successful".
	reqCode	String	64	Req.	Request ID, which is the same as 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 field 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" }] }			

5.2.12 blockArea

Empty or unblock the specified area. Emptying an area is to remove all AMRs from the area and block the area. When the area is blocked, all AMRs in the area will stop and AMRs outside the area are not allowed to enter.

API Definition

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

API Name	blockArea				
Function	Empty or unblock the specified area. Emptying an area is to remove all objects from the area and block the area.				
Protocol	REST				
Provider	RCS-2000				
Caller	Third-party platform				
Remarks	• You can configure the area via RCS-2000 • When the area is emptied, the AMRs in the area leaves, and the AMRs to enter the area will bypass.				
Request	**Parameter**	**Data Type**	**Max. Length**	**Req. or Opt.**	**Description**
	reqCode	String	32	Req.	Request ID, if a 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 emptied or unblocked
	indBind	String	1	Req.	Empty or unblock the area: "1"-empty, "0"-unblock.
	pause	String	1	Req.	Whether to pause emptying the area: "0" (no, default value) , 1 (yes). When the area is already emptied, whether to stop all AMRs on the map: "0" (no, default value) , 1 (yes).
	controlMod	String	2	Req.	Scheduling mode: "0" (move out of the area, default value), "1" (move to the interim park area), "2" (move to the specified area), "-1" (blocking mode, when an area is blocked,

					AMRs in this area will stop and no other AMR is allowed to enter this area).
	targetArea	String	16	Opt.	The specified area ID. This field is required when the value of **controlMod** is "2".
	noticeThird	String	2	Req.	Whether to notify the third-party platform when the area is emptied: 0 (no), 1 (yes).
Response	code	String	6	Req.	Status code, see ***Status Code*** (Appendix A) for details.
	message	String	64	Req.	Returned status description, e.g., "successful".
	reqCode	String	64	Req.	Request ID, which is the same as that in corresponding request message

Sample Codes	**Request**	{
		"reqCode": "1541954B96B1112",
		"matterArea": "2",
		"indBind": "1",
		"pause": "0",
		"controlMod": "2",
		"targetArea":"HF002"
		}			

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

5.2.13 resumeRobot

Resume the AMR, the resumed AMR 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 AMR, the resumed AMR 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 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 AMRs to be resumed: "-1" (all AMRs). **Note** • One of **robotCount** and **robots** is required. • To resume all AMRs, set **robotCount** to "-1", while to resume parts of AMRs, input the corresponding AMR ID list in **robots**.
	mapShortName	String	32	Opt.	Alias of the map where the AMR locates. **Note** This field is required when the value of **robotCount** is "-1".
	robots	String[]	16	Opt.	List of AMR IDs. **Note** • One of **robotCount** and **robots** is required. • To resume all AMRs, set **robotCount** to "-1", while to resume parts of AMRs, input corresponding AMR list via field **robots**.
Response	code	String	6	Req.	Status code, see ***Status Code*** (Appendix A) for details.
	message	String	64	Req.	Returned status description, e.g., "successful".
	reqCode	String	64	Req.	Request ID, which is the same as 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 queryAgvStatus

Search for AMR status, including the AMR battery.

API Definition

Table 5-14 POST http://[address]:8182/rcms-dps/rest/queryAgvStatus

API Name	queryAgvStatus				
Function	Search for AMR status, including the AMR battery.				
Protocol	REST				
Provider	RCS-2000				
Caller	Third-party platform				
Remarks	• API calling frequency: number of AMRs is less than 100: 5 seconds; number of AMRs is between 100 and 200: 10 seconds; number of AMRs is between 200 and 300: 15 seconds. • The request URI is "http://[address]:8182/rcms-dps/rest/queryAgvStatus"				
Request	**Parameter**	**Data Type**	**Max. Length**	**Req. or Opt.**	**Description**
	reqCode	String	32	Req.	Request ID, if a 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.
	mapShortName	String	32	Opt.	Alias of the map where the AMR locates.
Response	code	String	6	Req.	Status code, see ***Status Code*** (Appendix A) for details.
	data	Object			
	robotCode	String	5	Req.	AMR ID
	robotDir	String	4	Req.	AMR direction, range: [-180,360] degrees
	robotIp	String	64	Opt.	AMR IP address
	battery	String	4	Req.	AMR battery, range: [0,100]
	posX	String	8	Req.	AMR X-coordinate, unit: millimeter
	posY	String	8	Req.	AMR Y-coordinate, unit: millimeter
	mapCode	String	32	Req.	ID of map where the AMR locates
	speed	String	6	Req.	AMR current speed, unit: mm/s
	status	String	6	Req.	AMR status, see ***AMR Status*** for details.
	exclType	String	1	Req.	Whether the AMR is excluded: "1"-excluded, "0"-not excluded. **Note** No task will be assigned to the excluded AMR.
	stop	String	1	Req.	Whether the AMR 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 is the same as 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"
		}]
		}			

5.2.15 queryTaskStatus

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

API Definition

Table 5-15 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 AMR 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 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	5	Opt.	AMR ID, and at least one of **taskCodes** and **agvCode** should be configured.
Response	code	String	6	Req.	Status code, see ***Status Code*** (Appendix A) 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	5	Opt.	AMR ID, this field exists when the task has been assigned to.
	message	String	64	Req.	Returned status description, e.g., "successful".
	reqCode	String	64	Req.	Request ID, which is the same as 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"
		}]
		}			

5.2.16 queryPodBerthAndMat

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

API Definition

Table 5-16 POST http://[address][:port]/rcms/services/rest/hikRpcService/queryPodBerthAndMat

API Name	queryPodBerthAndMat				
Function	Search the relations among rack, position, and material batch.				
Protocol	REST				
Provider	RCS-2000				
Caller	Third-party platform				
Remarks	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 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*** (Appendix A) for details.
	message	String	64	Req.	Returned status description, e.g., "successful".

	reqCode	String	64	Req.	Request ID, which is the same as 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"
		}]
		}			

5.2.17 bindPodAndMat

Bind/unbind the material batch and rack.

API Definition

Table 5-17 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	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 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*** (Appendix A) for details.
	message	String	64	Req.	Returned status description, e.g., "successful".
	reqCode	String	64	Req.	Request ID, which is the same as 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"
		}

5.2.18 blockStgBin

Batch enable/disable bins. The batch operations will succeed or fail together. The bins that generate initialized racks are all enabled. When managing the bins, the third-party platform should ensure that the bin status is consistent with that of RCMS-2000.

API Definition

Table 5-18 POST http://[address][:port]/rcms/services/rest/hikRpcService/blockStgBin

API Name	blockStgBin				
Function	Batch enable/disable bins.				
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 request is submitted repeatedly, the request ID must be the same.
	reqTime	string	20	Opt.	Expiry date, 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.
	data	object []		Req.	Item type: object.
	⊢ stgBinCode	string	32	Req.	Bin ID.

	⊢ action	string	2	Req.	Operation type: "0" (enable bin), "1" (disable bin).
Response	reqCode	string	64	Req.	Request ID.
	code	string	6	Req.	Status code, see ***Status Code*** (Appendix A) for details.
	message	string	64	Req.	Returned status description.

5.2.19 getOutPod (Tower Station / Buffer Rack Picking Station)

Container outbound, used for the buffer rack picking station.

API Definition

Table 5-19 POST http://[address][:port]/rcms/services/rest/hikTpsService/getOutPod

API Name	getOutPod				
Function	Container outbound, used for the buffer rack picking station.				
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 request is submitted repeatedly, the request ID must be the same.
	reqTime	String	20	Opt.	Expiry date, format: yyyy-MM-dd Hh:mm:ss (date-time). Provided by the third-party platform.
	clientCode	String	16	Opt.	Third-party platform ID, e.g., PDA, HCWMS, which is provided to the third-party platform by TPS.
	tokenCode	String	64	Opt.	Token ID, which is provided by RCS-2000, which is provided to the third-party platform by TPS.

	taskTyp	String	2	Req.	Task type: "0" (TPS tasks), "1" (tower station tasks).
	data	Object []	/	Req.	Up to 1000 items are allowed.
	⊢ taskCode	String	64	Req.	Task ID.
	⊢ ctnrCode	String	32	Req.	Container ID.
	⊢ binCode	String	32	Opt.	Bin ID.
	⊢ wbCode	String	32	Req.	Workstation ID.
	⊢ agvTyp	String	16	Opt.	Device type.
	⊢ priority	String	3	Opt.	Task priority, range: [1-127], and the larger the value, the higher the priority.
Response	code	String	4	Req.	Status code, see ***Status Code*** (Appendix A) for details.
	message	String	512	Req.	Returned status description.
	reqCode	String	64	Req.	Request ID.
	data	String	2000	Opt.	Task ID.

5.2.20 returnPod (Tower Station / Buffer Rack Picking Station)

Container inbound, which is the returning route of the outbound task, mainly for the buffer rack picking station.

API Definition

Table 5-20 POST http://[address][:port]/rcms/services/rest/hikTpsService/returnPod

API Name	returnPod
Function	Container inbound, which is the returning route of the outbound task, mainly for the buffer rack picking station.
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 request is submitted repeatedly, the request ID must be the same. Provided by the third-party platform.
	reqTime	String	20	Opt.	Expiry date, format: yyyy-MM-dd HH:mm:ss, provided by the third-party platform.
	clientCode	String	16	Opt.	Third-party platform ID, e.g., PDA, HCWMS, which is provided to the third-party platform by TPS.
	tokenCode	String	64	Opt.	Token ID provided by RCS-2000, which is provided to the third-party platform by TPS.
	taskCode	String	64	Req.	Task ID of the current workstation. When there are multiple task IDs, you can upload any one of them for inbound.
	agvTyp	String	16	Opt.	Device type.
	returnPodStrategy	String	32	Opt.	Inbound strategy ID (or big area@small area, with multiple possibilities and different priorities. When the areas are contained in more than one strategy, the strategy in which the areas are prior will be selected.)
	taskTyp	String	2	Req.	Task type: "5" (container inbound).
	ctnrCode	String	32	Req.	Container ID.
	binCode	String	32	Req.	ID of target bin that the container should be carried back to.
	srcBinCode	String	32	Req.	ID of bin that the container should be carried from.
	wbCode	String	32	Opt.	Workstation ID (required for initializing inbound).
Response	code	String	4	Req.	Status code, see ***Status Code*** (Appendix A) for details.
	message	String	512	Req.	Returned status description.

	reqCode	String	64	Req.	Request ID.
	data	String	2000	Opt.	Task ID.

5.2.21 genCtuGroupTaskBatch (Tower Station)

Used for tower station outbound tasks to apply and perform tasks in order. Outbound in order does not support multi-workstations and double deep storage.

API Definition

Table 5-21 POST http://[address][:port]/rcms/services/rest/hikRpcService/genCtuGroupTaskBatch

API Name	genCtuGroupTaskBatch				
Function	Used for tower station outbound tasks to apply and perform tasks in order.				
Protocol	REST				
Provider	RCS-2000				
Caller	Third-party platform				
Remarks	When the tower station outbound is in groups, it will call ***genAgvSchedulingTask*** to transmit **groupId**.				
Request	**Parameter**	**Data Type**	**Max. Length**	**Req. or Opt.**	**Description**
	reqCode	String	32	Req.	Request ID. If a request is submitted repeatedly, the request ID must be the same.
	reqTime	String	20	Opt.	Expiry date, 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.
	taskTyp	String	16	Req.	Task type (default: B10).
	seqTyp	number	3	Req.	Outbound order type value: "1" (ordered between groups and within a group), "2" (ordered between groups and unordered within a group),

					"3" (unordered between groups and ordered within a group).
	taskGroups	Object []	/	Req.	Task information list.
	⊢ taskCode	String	64	Req.	Task ID.
	⊢ ctnrCode	String	32	Req.	Container ID.
	⊢ ctnrTyp	String	16	Req.	Container Type.
	⊢ wbCode	String	32	Req.	Workstation ID.
	⊢ wbTyp	number	2	Req.	Workstation type value: "0" (normal workstation), "1" (conveyor workstation), "2" (loader/unloader workstation).
	⊢ groupId	number		Req.	Task group ID or order between groups.
	⊢ sequence	number		Req.	Order within a group.
Response	code	String	4	Req.	Status code, see ***Status Code*** (Appendix A) for details.
	message	String	512	Req.	Returned status description.
	reqCode	String	64	Req.	Request ID.

5.2.22 boxApplyPass (Tower Station)

Container picking-up/placing callback. When docking with the third-party conveyor, the task template has set the picking/placing application. After CMS applies to the third-party platform, the third-party platform can call this API to notify CMS of picking up / placing containers.

API Definition

Table 5-22 POST http://[address][:port]/rcms/services/rest/hikRpcService/boxApplyPass

API Name	boxApplyPass
Function	Container picking-up/placing callback.

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.
	reqTime	String	20	Opt.	Request time.
	clientCode	String	16	Opt.	Third-party platform ID, e.g., PDA, HCWMS, which is provided to the third-party platform by RCS-2000.
	tokenCode	String	64	Opt.	Token ID, which is provided to the third-party platform by RCS-2000.
	taskCode	String	64	Req.	Task ID.
	type	String	2	Req.	Task type: "1" (picking request approved), "2" (placing request approved).
Response	code	String	4	Req.	Status code, see ***Status Code*** (Appendix A) for details.
	message	String	512	Req.	Returned status description.

5.3 API Provided by Third-Party

5.3.1 agvCallback

Send the task executing status to the third-party platform.

API Definition

Table 5-23 POST http://[address][:port]/xxx/agv/agvCallbackService/agvCallback

API Name	agvCallback
Function	Send the task executing status to the third-party platform.
Protocol	REST

Provider	Third-party platform				
Caller	RCS-2000				
Remarks	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 request is repeatedly submitted, the request ID must be same.
	reqTime	String	Opt.	Opt.	Request timestamp, format: YYYY-MM-DD hh:mm:ss
	cooX	String	8	Opt.	X-coordinate of location code, unit: mm. This field exists when the task has been completed.
	cooY	String	8	Opt.	Y-coordinate of location code, unit: mm. This field exists when the task has been completed.
	currentPositionCode	String	32	Req.	Current position ID For the status of task starting, it indicates the task starting point. For the status of getting out of storage section, it indicates the task starting point. For the status of task canceling, it indicates the workstation No. For the status of task ending, it indicates the destination point. For the request of picking-up/placing, it indicates the point of container to be picked-up/placed.
	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 field exists when the task has been completed.

	stgBinCode	String	32	Opt.	Storage bin ID. It is valid only for FMR and tower station tasks.
	method	String	16	Req.	Name to describe the task executing status: "start"-task started, "outbin"-executing, "end"-task completed, "cancel"-cancel task, "ctu"-cancel the request of picking up/placing container. It is provided by RCS-2000.
	podCode	String	16	Opt.	Rack ID, this field exists when the rack is carried.
	podDir	String	4	Opt.	Rack direction: "180"-leftward, "0"-rightward, "90"-upward, "-90"-downward; this field exists when the task has been completed.
	materialLot	String	32	Opt.	The material No.
	robotCode	String	5	Req.	AMR 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 as that configured by RCS-2000. This field exists when the task has been completed, and it is same as parameter **wbCode** of API ***genAgvSchedulingTask*** .
	ctnrCode	String	32	Opt.	Container No.
	ctnrTyp	String	2	Opt.	Container type.
	roadWayCode	String	16	Opt.	Roadway No.
	seq	String	2	Opt.	The sequence No. in the roadway: 0 (roadway end).
	eqpCode	String	32	Opt.	Equipment No., such as the loader-unloader and conveyor.
Response	code	String	6	Req.	Status code, see ***Status Code*** (Appendix A) for details.
	message	String	64	Req.	Returned status description, e.g., "successful".
	reqCode	String	64	Req.	Request ID, which is the same as 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 AMR stopping to the third-party platform. The alarm sending frequency is 10 seconds per time.

API Definition

Table 5-24 POST http://[address][:port]/service/rest/agvCallbackService/warnCallback

API Name	warnCallback
Function	RCS-2000 sends the severe alarm which causes AMR stopping to the third-party platform.
Protocol	REST
Provider	Third-party platform
Caller	RCS-2000

<table>
<tr><td>Remarks</td><td colspan="5">• This API is configured in the business notification configuration of RCS-2000. Up to three third-party services can be configured as the notification receiver (method), the default third-party service is 1001.
• 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.</td></tr>
<tr><td rowspan="11">Request</td><td>Parameter</td><td>Data Type</td><td>Max. Length</td><td>Req. or Opt.</td><td>Description</td></tr>
<tr><td>reqCode</td><td>String</td><td>32</td><td>Req.</td><td>Request ID, if a 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>
<tr><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>data</td><td colspan="4">Object</td></tr>
<tr><td>⊢ robotCode</td><td>String</td><td>5</td><td>Req.</td><td>AMR ID</td></tr>
<tr><td>⊢ beginTime</td><td>String</td><td>64</td><td>Req.</td><td>Alarm start time</td></tr>
<tr><td>⊢ warnContent</td><td>String</td><td>64</td><td>Req.</td><td>Alarm content, see <u>AMR Status</u> (Appendix B) for details.</td></tr>
<tr><td>⊢ taskCode</td><td>String</td><td>64</td><td>Opt.</td><td>Task No.</td></tr>
<tr><td rowspan="3">Response</td><td>code</td><td>String</td><td>6</td><td>Req.</td><td>Status code, see <u>Status Code</u> (Appendix A) for details.</td></tr>
<tr><td>message</td><td>String</td><td>64</td><td>Req.</td><td>Returned status description, e.g., "successful".</td></tr>
<tr><td>reqCode</td><td>String</td><td>64</td><td>Req.</td><td>Request ID, which is the same as that in corresponding request message.</td></tr>
<tr><td>Sample Codes</td><td>Request</td><td colspan="4"><pre>{
 "reqCode": "1541954B96B1112",
 "data:": [{
  "robotCode": "1001",
  "beginTime": "2020-04-02 23:12:12",</pre></td></tr>
</table>

		<pre>"warnContent": "Platform disconnected",
		"taskCode": "C002WWQQRR"
		},
		{
		"robotCode": "1002",
		"beginTime": "2020-04-02 23:12:12",
		"warnContent": "Guidance alarm",
		"taskCode": "C002WWQQRR33"
		}]
		}</pre>

	Response	<pre>{
		"code": "0",
		"message": "successful",
		"reqCode": "1541954B96B1112"
		}</pre>

5.3.3 bindNotify

Notify the third-party platform of the binding or unbinding operation between racks and locations, material batches and racks, or containers and bins.

API Definition

Table 5-25 POST http://[address][:port]/service/rest/bindNotify

API Name	bindNotify				
Function	Notify the third-party platform of the binding or unbinding operation between racks and locations, material batches and racks, or containers and bins.				
Protocol	REST				
Provider	Third-party platform				
Caller	RCS-2000				
Remarks	• This API is configured in the business notification configuration of RCS-2000. Up to three third-party services can be configured as the notification receiver (method), the default third-party service is 1001. • The request URL should be "http://[address][:port]/service/rest/bindNotify", of which the "http://[address][:port]/service/rest" can be configured in the system parameters configuration of RCS-2000.				
Request	**Parameter**	**Data Type**	**Max. Length**	**Req. or Opt.**	**Description**

	reqCode	String	32	Req.	Request ID, if a 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.
	method	String	16	Req.	Method name: "bindPodAndBerth" (bind and unbind rack with location), "bindPodAndMat" (bind and unbind material batch and rack), "bindCtnrAndBin" (bind and unbind container and bin).
	indBind	String	2	Req.	Bind or unbind: "1" (bind), 0 (unbind).
	bindParam	Object[]	/	Req.	Up to 2000 characters are allowed.
	⊢ berthCode	String	32	Opt.	Position ID, which is predefined with detailed coordinate information on map. It is valid when the value of **method** is "bindPodAndBerth".
	⊢ ctnrCode	String	32	Opt.	Container ID. It is valid when the value of **method** is "bindCtnrAndBin".
	⊢ ctnrType	String	32	Opt.	Container type. It is valid when the value of **method** is "bindCtnrAndBin".
	⊢ materialLot	String	32	Opt.	Material batch ID. It is valid when the value of **method** is "bindPodAndMat".
	⊢ podCode	String	32	Opt.	Rack ID. It is valid when the value of **method** is "bindPodAndBerth" or "bindPodAndMat".
	⊢ stgBinCode	String	32	Opt.	Bin ID. It is valid when the value of **method** is "bindCtnrAndBin".
Response	code	String	6	Req.	Status code, see ***Status Code*** (Appendix A) for details.
	message	String	64	Req.	Returned status description, e.g., "successful".

	reqCode	String	64	Req.	Request ID, which is the same as that in corresponding request message.

Sample Codes	**Request**	{
		"reqCode": "1541954B96B1112",
		"reqTime": "2020-09-17 13:48:58",
		"method": "bindPodAndBerth",
		"indBind": "0",
		"bindParam": [{
		"berthCode": "AB1",
		"podCode": "111111"
		}]
		}			

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

5.3.4 applyReturnForValid (Tower Station)

Used for container inbound task at the loader/unloader before entry, to check with the third-party platform whether the container inbound at the loader/unloader is allowed. If it is allowed, the containers will be transported into the loader/unloader; if not (usually due to the container problem), the containers will not enter the loader/unloader and should be instead manually handled.

API Definition

Table 5-26 POST http://[address][:port]/service/rest/applyReturnForValid

API Name	applyReturnForValid
Function	This API is only to check if the container inbound at the loader/unloader is allowed. If it is allowed, the RCS-2000 will apply for the inbound position of the container in ***applyReturnForBin (Tower Station)*** .
Protocol	REST
Provider	Third-party platform
Caller	RCS-2000

Request	Parameter	Data Type	Max. Length	Req. or Opt.	Description
	reqCode	String	32	Req.	Request ID. If a request is repeatedly submitted, the request ID must be same.
	reqTime	String	20	Opt.	Request timestamp, format: YYYY-MM-DD hh:mm:ss
	wbCode	String	32	Req.	Workstation ID.
	ctnrCode	String	30	Req.	Container ID.
Response	code	String	6	Req.	Status code, see ***Status Code*** (Appendix A) for details.
	message	String	64	Req.	Returned status description, e.g., "successful".
	reqCode	String	64	Req.	Request ID, which is the same as that in corresponding request message.

Sample Codes	**Request**	{
		"reqCode": "1541954B96B1112",
		"reqTime": "2020-09-17 13:48:58",
		"wbCode": "132435",
		"ctnrCode": "46576879"
		}			

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

5.3.5 applyReturnForBin (Tower Station)

Used for container inbound tasks at the loader/unloader or conveyor line, to apply to the third-party platform for the inbound position. The notification of applying for the inbound bin and the docking path with the third-party platform should be set in the B05, B06, and B08 task template. The returned data should follow the format listed in the response data description below.

API Definition

Table 5-27 POST http://[address][:port]/service/rest/applyReturnForBin

API Name	applyReturnForBin				
Function	Used for container inbound tasks at the loader/unloader or conveyor line, to apply to the third-party platform for inbound position of the container.				
Protocol	REST				
Provider	Third-party platform				
Caller	RCS-2000				
Request	**Parameter**	**Data Type**	**Max. Length**	**Req. or Opt.**	**Description**
	reqCode	String	32	Req.	Request ID. If a request is repeatedly submitted, the request ID must be same.
	reqTime	String	20	Opt.	Request timestamp, format: YYYY-MM-DD hh:mm:ss
	data	Object	/	Req.	Array of container IDs.
	ctnrCode	String	64	Req.	Container ID.
	wbCode	String	32	Req.	Workstation ID.
	groupCode	String	30	Req.	Group ID.
Response	code	String	6	Req.	Status code, see ***Status Code*** (Appendix A) for details.
	message	String	64	Req.	Returned status description, e.g., "successful".
	reqCode	String	64	Req.	Request ID, which is the same as that in corresponding request message.
	data	Object	/	Req.	Array of container information.
	ctnrCode	String	/	Req.	Container ID.
	ctnrType	String	/	Req.	Container type.
	binCode	String	/	Req.	Bin ID.

Sample Codes	**Request**	{
		"reqCode": "1541954B96B1112",
		"reqTime": "2020-09-17 13:48:58",
		"data": [{
		"ctnrCode": "46576879"			

		}], "wbCode": "132435" "groupCode": "AB1" }

	Response	{
		"code": "0",
		"message": "successful",
		"reqCode": "1541954B96B1112"
		"data": [{
		"ctnrCode": "46576879",
		"ctnrType": "23",
		"binCode": "98765"
		}

Appendix A. Status Code

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

Status Code Description

code	Description
0	Succeeded.
1	Failed. Refer to the field **message** for detailed error information.
6	No need to resend (the task of the same **reqCode** is not completed).
99	Unknown error, try again.

Appendix B. AMR Status

The AMR common statuses are list in the table below.

AMR Common Status

status	Description
1	Task completed.
2	Executing task
3	Abnormal task
4	Idle task
5	Robot stopped.
6	Lifting the rack.
7	Charging status
8	Curve movement
9	Full charge maintenance
11	Rack not recognized
12	Rack angle deflected
13	Motion library exception
14	Rack code unrecognized
15	Rack code mismatch
16	Lifting exception
17	Charging station exception
18	Battery not charging
20	Charging direction error
21	Platform command error
23	Abnormal unloading
24	The rack position deviated.
25	Robot not in the block zone
26	Retry putting down failed

status	Description
27	Incorrect rack location
28	Low battery for lifting
29	Robot reversing angle deflected
30	Lifting without rack
31	Blocking zone failed.
33	Rotation request temporarily failed.
34	Map switching code unrecognized