RoboKit NetProtocol Document

Seer Robotics

目录

第一章: 概述

第二章: 机器人状态 API

第三章: 机器人控制 API

第四章: 机器人任务 API

第五章: 机器人配置 API

第六章: 机器人核心 API

Author: Ye Yangsheng

Date: 2017-06-14

Version: v1.2.1

Website: http://www.seer-robotics.com

Email: support@seer-robotics.com

Generated by gitboook 3.2.2 at Mon Jul 10 2017 12:25:03 GMT+0000 (UTC)

概述

RoboKit 向用户开放机器人操作的相关 API。整套 API 采用 TCP request/response 问答的方式，机器人作为服务器接受客户端的请求并向客户端作出响应。API 请求由头部和数据区组成，头部为普通定长的字节，用于标识数据包基本信息以及数据区的长度及类型等，数据区为序列化后的 `JSON` 数据，根据头部中的类型信息进行反序列化可以得到相应的 `JSON` 对象。API 的响应也由头部和数据区组成，头部为根据请求得到的定长字节(响应头部与对应请求头部的关系见下文)，数据区也为序列化后的 `JSON` 数据。

API 测试及例程

我们将 API 测试工具开源在 Github，Source，该工具使用 Qt 编写，其源码可以作为例程参考。编译后的单 Windows 可执行文件也可在 Release 中下载。

API 类别

API 分为六类，分别为机器人状态 API、机器人控制 API、机器人任务 API、机器人配置 API、机器人核心 API、其他 API。

1. 机器人状态 API：用于查询机器人各种状态量，如位置、速度、报警信息等。
2. 机器人控制 API：用于向机器人发送开环控制指令，如前进速度、转向速度，重定位等。
3. 机器人任务 API：用于向机器人发送任务级别的控制指令，如去目标点、返航等。
4. 机器人配置 API：用于设置机器人的参数，以及下载、上传地图等。
5. 机器人核心 API：用于启停机器人、更换运行脚本等。
6. 其他 API：杂项，如喇叭控制等。

每个类别使用不用的端口，相互独立，因此对不同的类别可以使用不同的传输协议。

端口

类别	端口	允许连接数
机器人状态 API	19204	10
机器人控制 API	19205	1
机器人任务 API	19206	1
机器人配置 API	19207	1
机器人核心 API	19208	1
其他API	19210	1

机器人将对每个建立的 TCP 连接进行 KeepAlive 保活，对已死的连接会及时的清除。

机器人若收到任何不符合本协议报文头部的错误数据包，将会主动关闭该 TCP 连接（不会回复任何信息）。

其他情况下，机器人不会主动断开连接，API 使用者应自己控制连接何时中断。

对于仅允许建立一个连接的端口，如果该端口已经有连接建立，那么机器人将不响应其他任何建立连接的请求，直到先建立的连接断开。

对于每个连接，机器人都是以一问一答的形式处理的，即未作出响应前是不会处理下一个请求的，因此建议使用时等收到上一次请求的响应后再发送下一个请求。

API 报文结构

顺序如下:

名称	内容	长度（**byte**）	描述
报文同步头	0x5A	1 (`uint8_t`)	报文同步头, 用于确定报文头部的开始
协议版本	1	1 (`uint8_t`)	协议的主版本号, 对于 v1.x.x 版本的协议均填 0x01
序号		2 (`uint16_t`)	请求及响应的序号, 0 ~ 65535, 请求包与响应包的这个字段是相同的, API 使用者自行填入序号。 机器人对每个请求的响应都使用这个序号。 这个序号是为了便于用户区分响应是对应于哪个请求的
数据区长度		4 (`uint32_t`)	数据区长度, 即 JSON 序列化数据的长度
报文类型(编号)		2 (`uint16_t`)	标识报文的类型, 即 API 的编号, 具体见下文中各个 API 的章节
保留区域		6 (`uint8_t[6]`)	保留区, 用于未来可能的扩展
数据区		取决于头部中的数据区长度	JSON 序列化的数据内容

```cpp
// C++ 的包头结构体如下
#include <stdint.h>
struct ProtocolHeader {
    uint8_t  m_sync;
    uint8_t  m_version;
    uint16_t m_number;
    uint32_t m_length;
    uint16_t m_type;
    uint8_t  m_reserved[6];
};
```

如上表所示, 报文头部的长度为 `1 + 1 + 2 + 4 + 2 + 6 = 16` 字节, 请求与响应的报文头部长度是相同的。如果某个报文类型没有数据区, 那么数据区长度的内容为 0x00000000。报文类型(编号)必须是下文API编号中的一种, 机器人若收到未定义的报文类型将会响应一个错误。若报文头部格式错误无法解析, 机器人将断开连接, 不做任何响应。若数据区解析错误, 机器人将认为这是一个错误的请求（参见对错误请求的响应）。

注 *1:* 每个报文头部正确的请求都会有一个响应, 编号用于将请求与响应对应起来, 一般情况下, 响应的编号为请求编号加 *10000 (0x2710)*

注 *2:* 机器人任何情况下都不会主动发送数据

注 *3:* 机器人若收到不属于某个端口的报文类型, 将会认为是错误的请求并响应 *60000*

注 *4:* 保留区域的 *6* 个字节必须填充不可省略, 一般用 *0x000000000000* 填充即可

请求数据区示例

以下文中的 2002 (0x07D2) 号请求为例:

2002(0x07D2) 号为重定位请求, 参数为 x 坐标, y 坐标, 标准 `JSON` 对象如下:

```
{
    "x": 10.0,
    "y": 3.0,
    "angle": 0
}
```

在报文中，需要将上面的 `JSON` 对象序列化并塞入报文的数据区，并将序列化后的长度填入报文头部的数据区长度中。

序列化后的数据为 `{"x":10.0,"y":3.0,"angle":0}` ，转化为十六进制为 `7B 22 78 22 3A 31 30 2E 30 2C 22 79 22 3A 33 2E 30 2C 22 61 6E 67 6C 65 22 3A 30 7D` ，长度为28(0x1c)个字节，因此头部为 `5A 01 00 01 00 00 00 1C 07 D2 00 00 00 00 00 00` ，将头部与数据区拼接在一起得到一个请求如下(共 44 字节):

```
5A 01 00 01 00 00 00 1C 07 D2 00 00 00 00 00 00
7B 22 78 22 3A 31 30 2E 30 2C 22 79 22 3A 33 2E 30 2C 22 61 6E 67 6C 65 22 3A 30 7D
```

响应数据区示例

以下文中的 11004 (0x2AFC) 号响应为例:

机器人当前的位置为 `x = 6.0, y = 2.0, angle = 1.57, confidence = 0.9` ，表示成 `JSON` 对象如下:

```
{
    "ret_code": 0,
    "x": 6.0,
    "y": 2.0,
    "angle": 1.57,
    "confidence": 0.9
}
```

序列化后的数据为 `{"ret_code":0,"x":6.0,"y":2.0,"angle":1.57,"confidence":0.9}` ，转化为十六进制为 `7B 22 72 65 74 5F 63 6F 64 65 22 3A 30 2C 22 78 22 3A 36 2E 30 2C 22 79 22 3A 32 2E 30 2C 22 61 6E 67 6C 65 22 3A 31 2E 35 37 2C 22 63 6F 6E 66 69 64 65 6E 63 65 22 3A 30 2E 39 7D` ，长度为60(0x3C)字节，因此响应的头部为 `5A 01 00 01 00 00 00 3C 2A FC 00 00 00 00 00 00` 拼接后数据如下(共 76 字节):

```
5A 01 00 01 00 00 00 3C 2A FC 00 00 00 00 00 00
7B 22 72 65 74 5F 63 6F 64 65 22 3A 30 2C 22 78 22 3A 36 2E 30 2C 22 79 22 3A 32 2E 30 2C
22 61 6E 67 6C 65 22 3A 31 2E 35 37 2C 22 63 6F 6E 66 69 64 65 6E 63 65 22 3A 30 2E 39 7D
```

坐标系

在 RoboKit 中，存在两套坐标系，分别为世界坐标系和机器人坐标系。 世界坐标系为右手坐标系，即地图的坐标系，与地图和定位相关的操作均使用世界坐标系。如机器人的位置 (x, y, r) 即描述的是的机器人在世界坐标系中的 x 坐标，y 坐标，r 朝向角。示意图见下文

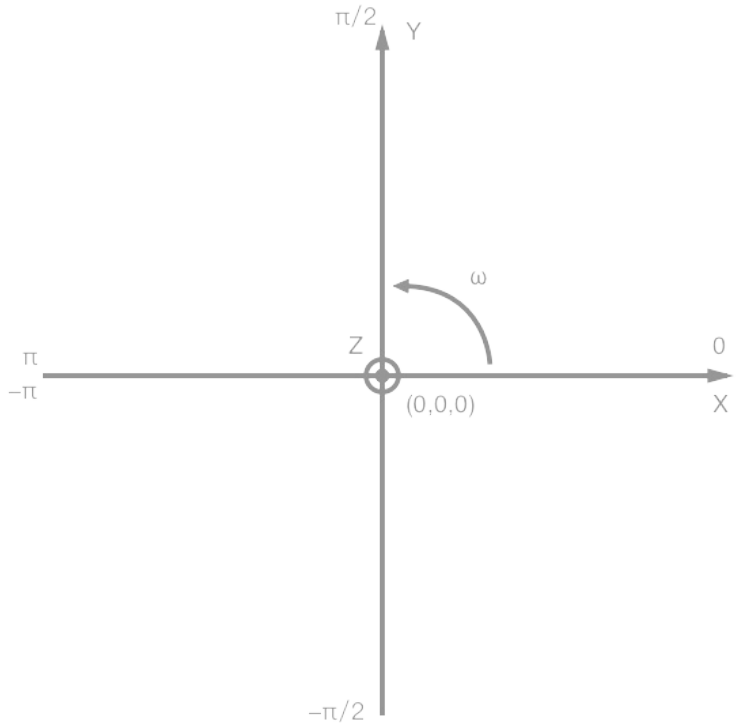
对于机器人本身的运动则需要参考机器人坐标系，机器人坐标系为右手坐标系，详情见下文

世界坐标系

世界坐标系为右手坐标系

原点

在 RoboShop 中打开一张地图即可看到该地图的世界坐标系原点，通过坐标变换工具也可以任意改变地图中原点的位置。

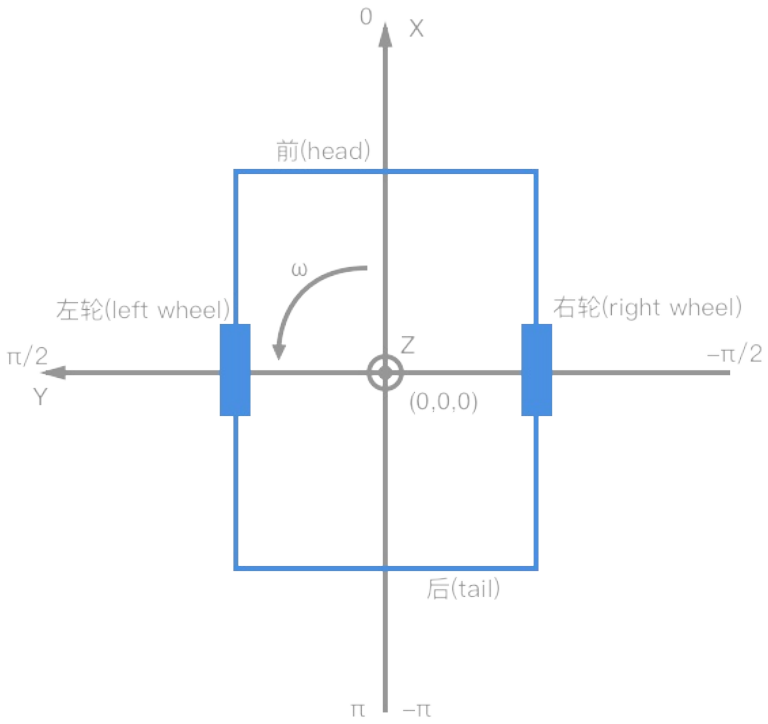


机器人坐标系

机器人坐标系为右手坐标系

原点

对于以差动方式运动的机器人来说，机器人坐标系的原点即为两轮轴距的中心点，如下图所示:



注意: 当我们说到机器人速度的时候, 一般情况下, ***vx*** 为机器人在机器人坐标系下的 ***x*** 轴方向的速度, ***vy*** 为机器人在机器人坐标系下的 ***y*** 轴方向的速度, ***w*** 为机器人在机器人坐标系下的角速度

API 使用注意事项

1. JSON 中的类型为 Javascript 中的类型, 可以是数字(number)、字符串(string)、布尔值(boolean)、数组(array)、对象(object)、null。
2. 所有未特别说明的数值的单位均为国际单位, 如 m、rad, m/s 等。
3. JSON 数据区字段若标注 TODO, 表示该API暂不支持。
4. API响应的编号为请求的编号 + 10000 (0x2710), 请求与响应一一对应, 除了对错误请求的响应。
5. 请不要发送任何非 ACSII 字符，特别是地图名相关的操作，含有中文字符的地图名将会被认为是非法的。

机器人状态 **API**

机器人状态 API 用于查询机器人的所有状态量。

如果机器人的超时参数 timeOut 设置了大于 0 的值, 那么当 19204 端口没有任何一个连接时，机器人将马上静止，等待超时时间后机器人将返回 ReturnPoint。如果在超时时间内又有连接连上 19204 端口，机器人将不会返回 ReturnPoint, 但机器人仍然会静止，需要人工恢复任务。

API 列表

请求

编号	名称	描述
1000	robot_status_info_req	查询机器人信息
1002	robot_status_run_req	查询机器人的运行状态信息(如运行时间、里程等)
1003	robot_status_mode_req	查询机器人运行模式
1004	robot_status_loc_req	查询机器人位置
1005	robot_status_speed_req	查询机器人速度
1006	robot_status_block_req	查询机器人被阻挡状态
1007	robot_status_battery_req	查询机器人电池状态
1008	robot_status_brake_req	查询机器人抱闸状态
1009	robot_status_laser_req	查询机器人激光雷达数据
1010	robot_status_path_req	查询机器人路径数据
1011	robot_status_area_req	查询机器人当前所在区域
1012	robot_status_emergency_req	查询机器人急停状态
1013	robot_status_io_req	查询机器人IO数据
1020	robot_status_task_req	查询机器人任务状态, 任务站点, 任务相关路径等
1021	robot_status_reloc_req	查询机器人重定位状态
1022	robot_status_loadmap_req	查询机器人地图载入状态
1025	robot_status_slam_req	查询机器人扫图状态
1050	robot_status_alarm_req	查询机器人告警状态
1100	robot_status_all1_req	查询批量数据1
1101	robot_status_all2_req	查询批量数据2
1102	robot_status_all3_req	查询批量数据3
1111	robot_status_init_req	查询机器人初始化状态
1300	robot_status_map_req	查询机器人载入的地图以及储存的地图
1301	robot_status_station	查询机器人当前载入地图中的站点信息
1400	robot_status_params_req	查询机器人参数
...	...	...

响应

编号	名称	描述
11000	robot_status_info_res	对 1000 请求的响应
11002	robot_status_run_res	对 1002 请求的响应
11003	robot_status_mode_res	对 1003 请求的响应
11004	robot_status_loc_res	对 1004 请求的响应
11005	robot_status_speed_res	对 1005 请求的响应
11006	robot_status_block_res	对 1006 请求的响应
11007	robot_status_battery_res	对 1007 请求的响应
11008	robot_status_brake_res	对 1008 请求的响应
11009	robot_status_laser_res	对 1009 请求的响应
11010	robot_status_path_res	对 1010 请求的响应
11011	robot_status_area_res	对 1011 请求的响应
11012	robot_status_emergency_res	对 1012 请求的响应
11013	robot_status_io_res	对 1013 请求的响应
11020	robot_status_task_res	对 1020 请求的响应
11021	robot_status_reloc_res	对 1021 请求的响应
11022	robot_status_loadmap_res	对 1022 请求的响应
11025	robot_status_slam_res	对 1025 请求的响应
11050	robot_status_alarm_res	对 1050 请求的响应
11100	robot_status_all1_res	对 1100 请求的响应
11101	robot_status_all2_res	对 1101 请求的响应
11102	robot_status_all3_res	对 1102 请求的响应
11111	robot_status_init_res	对 1111 请求的响应
11300	robot_status_map_res	对 1300 请求的响应
11301	robot_status_station_res	对 1301 请求的响应
11400	robot_status_params_res	对 1400 请求的响应
…	…	…

查询机器人信息

请求

- 编号: 1000 (0x03E8)
- 名称: robot_status_info_req
- 描述: 查询机器人信息
- JSON 数据区: 无

请求示例

数据区长度为0, 只有包头:

```
5A 01 00 01 00 00 00 00 03 E8 00 00 00 00 00 00
```

响应

- 编号: 11000 (0x2AF8)
- 名称: robot_status_info_res
- 描述: 查询机器人信息的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
id	string	机器人 ID	否
version	string	RoboKit 版本号	否
model	string	机器人模型名	否
dsp_version	string	固件版本	否
map_version	string	地图版本	否
model_version	string	模型版本	否
netprotocol_version	string	网络协议版本	否
width	number	机器人宽度	是
head	number	机器人头部长度	是
tail	number	机器人尾部长度	是
wheelbase	number	机器人轴距	是
wheel_radius	number	机器人轮半径	是
encoder_line	number	机器人编码器线数	是
reduction_ratio	number	机器人减速比	是
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

假设响应内容为:


```
{"id":"S001","version":"v1.1.0","model":"S1","dsp_version":"v1.2.2","map_version":"v1.0.0","model_version":"v1.1.0","
netprotocol_version":"v1.2.0"}
```

加上头部共 163 (0xA3) 个字节 (Byte), 因此响应的内容为:

```
5A 01 00 01 00 00 00 2E 2A F8 00 00 00 00 00 00
7B 22 69 64 22 3A 22 53 30 30 31 22 2C 22 76 65 72 73 69 6F
6E 22 3A 22 76 31 2E 31 2E 30 22 2C 22 6D 6F 64 65 6C 22 3A
22 53 31 22 2C 22 64 73 70 5F 76 65 72 73 69 6F 6E 22 3A 22
76 31 2E 32 2E 32 22 2C 22 6D 61 70 5F 76 65 72 73 69 6F 6E
22 3A 22 76 31 2E 30 2E 30 22 2C 22 6D 6F 64 65 6C 5F 76 65
72 73 69 6F 6E 22 3A 22 76 31 2E 31 2E 30 22 2C 22 6E 65 74
70 72 6F 74 6F 63 6F 6C 5F 76 65 72 73 69 6F 6E 22 3A 22 76
31 2E 32 2E 30 22 7D
```

查询机器人运行信息

请求

- 编号: 1002 (0x03EA)
- 名称: robot_status_run_req
- 描述: 查询机器人的运行状态信息(如运行时间、里程等)
- JSON 数据区: 无

请求示例

略

响应

- 编号: 11002 (0x2AFA)
- 名称: robot_status_run_res
- 描述: 查询机器人的运行状态信息的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
odo	number	累计行驶里程, 单位 m	否
time	number	本次运行时间, 单位 ms	否
total_time	number	累计运行时间, 单位 ms	否
controller_temp	number	控制器温度, 单位 ℃	是
controller_humi	number	控制器湿度, 单位 %	是
controller_voltage	number	控制器电压, 单位 V	是
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

查询机器人运行模式

请求

- 编号: 1003 (0x03EB)
- 名称: robot_status_mode_req
- 描述: 查询机器人运行模式
- JSON 数据区: 无

请求示例

略

响应

- 编号: 11003 (0x2AFB)
- 名称: robot_status_mode_res
- 描述: 查询机器人运行模式的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
mode	number	0 表示手动模式, 1 表示自动模式	否
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

查询机器人位置

请求

- 编号: 1004 (0x03EC)
- 名称: robot_status_loc_req
- 描述: 查询机器人位置
- JSON 数据区: 无

请求示例

略

响应

- 编号: 11004 (0x2AFC)
- 名称: robot_status_loc_res
- 描述: 查询机器人位置的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
x	number	机器人的 x 坐标, 单位 m	否
y	number	机器人的 y 坐标, 单位 m	否
angle	number	机器人的 angle 坐标, 单位 rad	否
confidence	number	机器人的定位置信度, 范围 [0, 1]	否
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

查询机器人速度

请求

- 编号: 1005 (0x03ED)
- 名称: robot_status_speed_req
- 描述: 查询机器人速度
- JSON 数据区: 无

请求示例

略

响应

- 编号: 11005 (0x2AFD)
- 名称: robot_status_speed_res
- 描述: 查询机器人速度的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
vx	number	机器人在机器人坐标系的 x 方向速度, 单位 m/s	否
vy	number	机器人在机器人坐标系的 y 方向速度, 单位 m/s	否
w	number	机器人在机器人坐标系的角速度(即顺时针转为负, 逆时针转为正), 单位 rad/s	否
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

查询机器人的被阻挡状态

请求

- 编号: 1006 (0x03EE)
- 名称: robot_status_block_req
- 描述: 查询机器人的被阻挡状态
- JSON 数据区: 无

请求示例

略

响应

- 编号: 11006 (0x2AFE)
- 名称: robot_status_block_res
- 描述: 查询机器人的被阻挡状态的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
blocked	boolean	机器人是否被阻挡	否
block_reason	number	被阻挡的原因, 0 = Ultrasonic (超声检测到被阻挡), 1 = Laser (激光检测到被阻挡), 2 = Fallingdown (防跌落传感器检测到被阻挡), 3 = Collision (碰撞传感器检测到被阻挡), 4 = Infrared (红外传感器检测到被阻挡)	是
block_x	number	阻挡位置的 x 坐标, 单位 m	是
block_y	number	阻挡位置的 y 坐标, 单位 m	是
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

查询机器人电池状态

请求

- 编号: 1007 (0x03EF)
- 名称: robot_status_battery_req
- 描述: 查询机器人电池状态
- JSON 数据区: 无

请求示例

略

响应

- 编号: 11007 (0x2AFF)
- 名称: robot_status_battery_res
- 描述: 查询机器人电池状态的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
battery_level	number	机器人电池电量, 范围 [0, 1]	是
battery_temp	number	机器人电池温度, 单位 ℃	是
charging	boolean	电池是否正在充电	是
voltage	number	电压, 单位 V	是
current	number	电流, 单位 A	是
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

查询机器人抱闸状态

请求

- 编号: 1008 (0x03F0)
- 名称: robot_status_brake_req
- 描述: 查询机器人抱闸状态
- JSON 数据区: 无

请求示例

略

响应

- 编号: 11008 (0x2B00)
- 名称: robot_status_brake_res
- 描述: 查询机器人抱闸状态的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
brake	boolean	机器人是否抱闸, 如果该字段不存在, 说明机器人没有抱闸功能	是
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

查询机器人激光雷达数据

请求

- 编号: 1009 (0x03F1)
- 名称: robot_status_laser_req
- 描述: 查询机器人激光雷达数据
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
step	number	激光数据采样的步长, 0 或 1 为不进行采样, 即原始数据; 如果缺省, 则使用默认值, 默认值见配置的参数, 一般为 5	是

请求示例

假设数据区为 `{"step":3}`, 长度为 10 (0x0A) 个字节。

拼接包头后请求数据如下:

```
5A 01 00 01 00 00 00 0A 03 F1 00 00 00 00 00 00
7D 22 73 74 65 70 22 3A 33 7D
```

响应

- 编号: 11009 (0x2B01)
- 名称: robot_status_laser_res
- 描述: 查询机器人激光雷达数据的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
laser_beams	array[array[number]], array 中每个元素也是 array, 其包含两个 number, 分别为 x, y 代表激光点在世界坐标系中的坐标	激光点数据, 数据示例见下文	否
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

数据示例:

如下为五个激光点的数据:

```
{
    "laser_beams": [[1.0, 1.0], [1.2, 0.2], [2.12, -1.23], [2.31, 2.132], [-12.21, -5.22]]
}
```

为方便阅读，这里 JSON 数据采用了展开的形式, 实际使用中为减少数据流 JSON 数据区是紧凑的, 不会有多余的空格与换行, 后文中同理。

响应示例

略

查询机器人自由导航路径数据

请求

- 编号: 1010 (0x03F2)
- 名称: robot_status_path_req
- 描述: 查询机器人自由导航时的路径数据
- JSON 数据区: 无

注意：机器人的路径数据只有在自由导航的时候才有意义，固定路径导航时机器人是沿着绘制的贝塞尔曲线运动的，通过该 ***API*** 查询到的路径数据没有意义，如果需要查询固定路径导航以及巡检时的路径，请参阅 ***1020*** 任务状态

请求示例

略

响应

- 编号: 11010 (0x2B02)
- 名称: robot_status_path_res
- 描述: 查询机器人自由导航时的路径数据的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
path	array[array[number]], array 中每个元素也是 array, 其包含两个 number, 分别为 x, y 代表路径点在世界坐标系中的坐标	按顺序的路径点数据, 依次连接就是机器人规划的路径, 数据示例见下文	否
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

数据示例：

如下为五个激光点的数据

```
{
    "path": [[1.0, 1.2], [2.0, 1.3], [3.2, 2.1], [3.4, 1.4], [3.9, 1.5]]
}
```

如上的路径由 5 个点组成，5 个点分别为 `(1.0, 1.2)`，`(2.0, 1.3)`，`(3.2, 2.1)`，`(3.4, 1.4)`，`(3.9, 1.5)`，其坐标值为在世界坐标系中的坐标，将这 5 个点按顺序连接即为机器人当前规划的路径。

响应示例

略

查询机器人当前所在区域

请求

- 编号: 1011 (0x03F3)
- 名称: robot_status_area_req
- 描述: 查询机器人当前所在的区域信息
- JSON 数据区: 无

请求示例

略

响应

- 编号: 11011 (0x2B03)
- 名称: robot_status_area_res
- 描述: 查询机器人当前所在的区域信息的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
area_ids	array[string]	机器人所在区域 id 的数组(由于地图上的区域是可以重叠的, 所以机器人可能同时在多个区域)，数组可能为空	否
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

数据示例:

如上说明机器人当前处于 id 号为 "1" 和 "2" 的区域中

```
{
    "area_ids": ["1","2"]
}
```

响应示例

略

查询机器人急停状态

请求

- 编号: 1012 (0x03F4)
- 名称: robot_status_emergency_req
- 描述: 查询机器人急停按钮的状态
- JSON 数据区: 无

请求示例

略

响应

- 编号: 11012 (0x2B04)
- 名称: robot_status_emergency_res
- 描述: 查询机器人急停按钮的状态的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
emergency	boolean	true 表示急停按钮处于激活状态(按下), false 表示急停按钮处于非激活状态(拔起)	是
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

数据示例:

```
{
    "emergency": false
}
```

响应示例

略

查询机器人IO数据

请求

- 编号: 1013 (0x03F5)
- 名称: robot_status_io_req
- 描述: 查询机器人IO数据
- JSON 数据区: 无

请求示例

略

响应

- 编号: 11013 (0x2B05)
- 名称: robot_status_io_res
- 描述: 查询机器人IO数据的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
DI	array[boolean]	DI 数据, boolean 表示高低电平, 从 0 开始, 默认为 true (高电平)	是
DO	array[boolean]	DO 数据, boolean 表示高低电平, 从 0 开始, 默认为 true (高电平)	是
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

查询机器人任务状态

请求

- 编号: 1020 (0x03FC)
- 名称: robot_status_task_req
- 描述: 查询机器人当前的任务状态, 任务站点, 任务相关路径(已经经过的站点, 尚未经过的站点)等
- JSON 数据区: 无

请求示例

略

响应

- 编号: 11020 (0x2B0C)
- 名称: robot_status_task_req
- 描述: 查询机器人当前的任务状态的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
task_status	number	0 = NONE, 1 = WAITING, 2 = RUNNING, 3 = SUSPENDED, 4 = COMPLETED, 5 = FAILED, 6 = CANCELED	否
task_type	number	任务类型, 0 = 没有任务, 1 = 自由导航到任意点, 2 = 自由导航到站点, 3 = 固定路径导航到站点, 4 = 巡检 ,100 = 其他	否
target_id	string	当前任务要去的站点, 仅当 task_type 为 2 或 3 时该字段有效, task_status 为 RUNNING 时说明正在去这个站点, task_status 为 COMPLETED 时说明已经到达这个站点, task_status 为 FAILED 时说明去这个站点失败, task_status 为 SUSPENDED 说明去这个站点的任务暂停	是
target_point	array[number]	当前任务要去的坐标点, 为一个包含三个元素的数组, 分别为在世界坐标系中的 x, y, r 坐标, 仅当 task_type 为 1 时该字段有效, task_status 为 RUNNING 时说明正在去这个坐标点, task_status 为 COMPLETED 时说明已经到达这个坐标点, task_status 为 FAILED 时说明去这个坐标点失败, task_status 为 SUSPENDED 说明去这个坐标点的任务暂停	是
finished_path	array[string]	当前任务路径上已经经过的站点, 为站点的数组, 仅当 task_type 为 3 或 4 时该字段有效点。如果是固定路径导航任务, 这里会列出所有已经经过的中间点。如果是巡检任务, 则会分解为多个固定路径导航任务, 并显示当前固定路径导航任务中已经经过的中间点。	是
unfinished_path	array[string]	当前任务路径上尚未经过的站点, 为站点的数组, 仅当 task_type 为 3 或 4 时该字段有效。如果是固定路径导航任务, 这里会列出所有尚未经过的中间点。如果是巡检任务, 则会分解为多个固定路径导航任务, 并显示当前固定路径导航任务中尚未经过的中间点。	是
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

数据示例:

如下数据表示机器人正在使用固定路径导航的方式去 id 为 "1" 的站点, 已经经过的站点为 1, 2, 3, 尚未经过的站点为 4, 5, 6

```
{
    "task_status": 2,
    "task_type": 3,
    "target_id": "1",
    "finished_path": ["1", "2", "3"],
    "unfinished_path": ["4", "5", "6"]
}
```

如下数据表示机器人正在使用自由导航的方式去世界坐标为 (1.2, 2.3) 的点

```
{
    "task_status": 2,
    "task_type": 1,
    "target_point": [1.2, 2.3],
}
```

响应示例

略

查询机器人重定位状态

请求

- 编号: 1021 (0x03FD)
- 名称: robot_status_reloc_req
- 描述: 查询机器人当前的重定位状态
- JSON 数据区: 无

请求示例

略

响应

- 编号: 11021 (0x2B0D)
- 名称: robot_status_reloc_res
- 描述: 查询机器人当前的重定位状态的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
reloc_status	number	0 = FAILED(重定位失败), 1 = SUCCESS(重定位正确), 2 = RELOCING(正在重定位), 3=COMPLETED(重定位完成)	否
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

注意**:** 重定位状态 ***(1021)*** 用于指示当前机器人定位状态是否正确，重定位状态如果为 ***COMPLETED,*** 说明重定位已经完成，但是用户没有确认，此时需要通过 ***(2003)*** 来进行确认或重新进行定位。用户确认重定位正确后，重定位状态会变为 ***SUCCESS,*** 此时才可以切换为自动模式 ***(4000)***

响应示例

略

查询机器人地图载入状态

请求

- 编号: 1022 (0x03FE)
- 名称: robot_status_loadmap_req
- 描述: 查询机器人当前地图载入状态
- JSON 数据区: 无

请求示例

略

响应

- 编号: 11022 (0x2B0E)
- 名称: robot_status_loadmap_res
- 描述: 查询机器人当前地图载入状态的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
loadmap_status	number	0 = FAILED(载入地图失败), 1 = SUCCESS(载入地图成功), 2 = LOADING(正在载入地图)	否
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

注意**:** 地图载入状态 ***(1022)*** 用于指示当前地图是否载入成功，当机器人刚启动或者发生切换地图操作时，地图载入状态会指示为 ***LOADING***，此时无法进行重定位操作，必须等地图载入状态变为 ***SUCCESS*** 后才能进行重定位

响应示例

略

查询机器人扫图状态

请求

- 编号: 1025 (0x0401)
- 名称: robot_status_slam_req
- 描述: 查询机器人当前的扫图状态
- JSON 数据区: 无

请求示例

略

响应

- 编号: 11025 (0x2B11)
- 名称: robot_status_slam_res
- 描述: 查询机器人当前的扫图状态的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
slam_status	number	0 = 没有扫图, 1 = 正在扫图, 2 = 正在实时扫图	否
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

查询机器人告警状态

请求

- 编号: 1050 (0x041A)
- 名称: robot_status_alarm_req
- 描述: 查询机器人当前的告警状态
- JSON 数据区: 无

请求示例

略

响应

- 编号: 11050 (0x2B2A)
- 名称: robot_status_alarm_res
- 描述: 查询机器人当前的告警状态的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
fatals	array[object]	告警码 Fatal的数组, 所有出现的 Fatal 告警都会出现在数据中, object 格式见下文	是
errors	array[object]	告警码 Error的数组, 所有出现的 Error 告警都会出现在数据中, object 格式见下文	是
warnings	array[object]	告警码 Warning的数组, 所有出现的 Warning 告警都会出现在数据中, object 格式见下文	是
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

object 格式如下:

```
{"50000":1497698400} // key 为报警码, 对应的值为自 Thu Jan 01 08:00:00 1970 至报警码产生时经过的秒数
```

响应示例

```
{
    "fatals": [{"50000":1497698400}],
    "errors": [{"52201":1497698402},{"52118":1497698404}],
    "warnings": [{"54003":1497698405}]
}
```

查询批量数据1

请求

- 编号: 1100 (0x044C)
- 名称: robot_status_all1_req
- 描述: 查询批量数据1
- JSON 数据区: 无

all1 是为了通过一个请求批量获取之前提到的大部分状态数据, 包括 `1002`, `1003`, `1004`, `1005`, `1006`, `1007`, `1008`, `1009`, `1010`, `1011`, `1012`, `1013`, `1020`, `1021`, `1022`, `1024`, `1025`, `1050`

请求示例

略

响应

- 编号: 11100 (0x2B5C)
- 名称: robot_status_all1_res
- 描述: 查询批量数据1的响应
- JSON 数据区: 见下表

JSON 数据区会包含 `11002`, `11003`, `11004`, `11005`, `11006`, `11007`, `11008`, `11009`, `11010`, `11011`, `11012`, `11013`, `11020`, `11021`, `11022`, `11024`, `11025`, `11050` 的所有字段

字段名	类型	描述	可缺省
odo	number	累计行驶里程, 单位 m	否
time	number	本次运行时间, 单位 ms	否
total_time	number	累计运行时间, 单位 ms	否
controller_temp	number	控制器温度, 单位 ℃	是
controller_humi	number	控制器湿度, 单位 %	是
controller_voltage	number	控制器电压, 单位 V	是
mode	number	0 表示手动模式, 1 表示自动模式	否
x	number	机器人的 x 坐标, 单位 m	否
y	number	机器人的 y 坐标, 单位 m	否
angle	number	机器人的 angle 坐标, 单位 rad	否
confidence	number	机器人的定位置信度, 范围 [0, 1]	否
vx	number	机器人在机器人坐标系的 x 方向速度, 单位 m/s	否
vy	number	机器人在机器人坐标系的 y 方向速度, 单位 m/s	否
w	number	机器人在机器人坐标系的角速度, 单位 rad/s	否
blocked	boolean	机器人是否被阻挡	否
block_reason	number	被阻挡的原因, 0 = Ultrasonic (超声检测到被阻挡), 1 = Laser (激光检测到被阻挡), 2 = Fallingdown (防跌落传感器检测到被阻挡), 3 = Collision (碰撞传感器检测到被阻挡)	是
block_x	number	阻挡位置的 x 坐标, 单位 m	是

block_y	number	阻挡位置的 y 坐标, 单位 m	是
battery_level	number	机器人电池电量, 范围 [0, 1]	是
battery_temp	number	机器人电池温度, 单位 ℃	是
charging	boolean	电池是否正在充电	是
voltage	number	电压, 单位 V	是
current	number	电流, 单位 A	是
brake	boolean	机器人是否抱闸, 如果该字段不存在, 说明机器人没有抱闸功能	是
laser_beams	array[array[number]], array 中每个元素也是 array, 其包含两个 number, 分别为 x, y 代表激光点在世界坐标系中的坐标	激光点数据, 激光采样步长为 NetProtocol 中 laserStep 的值	否
path	array[array[number]], array 中每个元素也是 array, 其包含两个 number, 分别为 x, y 代表路径点在世界坐标系中的坐标	按顺序的路径点数据, 依次连接就是机器人规划的路径	否
area_ids	array[string]	机器人所在区域 id 的数组(由于地图上的区域是可以重叠的, 所以机器人可能同时在多个区域)，数组可能为空	否
emergency	boolean	true 表示急停按钮处于激活状态(按下), false 表示急停按钮处于非激活状态(拔起)	是
DI	array[boolean]	DI 数据, boolean 表示高低电平, 从 0 开始, 默认为 true (高电平)	是
DO	array[boolean]	DO 数据, boolean 表示高低电平, 从 0 开始, 默认为 true (高电平)	是
task_status	number	0 = NONE, 1 = WAITING, 2 = RUNNING, 3 = SUSPENDED, 4 = COMPLETED, 5 = FAILED, 6 = CANCELED	否
task_type	number	任务类型, 0 = 没有任务, 1 = 自由导航到任意点, 2 = 自由导航到站点, 3 = 固定路径导航到站点, 4 = 其他	否
target_id	string	当前任务要去的站点, 仅当 task_type 为 2 或 3 时该字段有效, task_status 为 RUNNING 时说明正在去这个站点, task_status 为 COMPLETED 时说明已经到达这个站点, task_status 为 FAILED 时说明去这个站点失败, task_status 为 SUSPENDED 说明去这个站点的任务暂停	是
target_point	array[number]	当前任务要去的坐标点, 为一个包含三个元素的数组, 分别为在世界坐标系中的 x, y, r 坐标, 仅当 task_type 为 1 时该字段有效, task_status 为 RUNNING 时说明正在去这个坐标点, task_status 为 COMPLETED 时说明已经到达这个坐标点, task_status 为 FAILED 时说明去这个坐标点失败, task_status 为 SUSPENDED 说明去这个坐标点的任务暂停	是
finished_path	array[string]	当前任务路径上已经经过的站点, 为站点的数组, 仅当 task_type 为 3 或 4 时该字段有效点。如果是固定路径导航任务, 这里会列出所有已经经过的中间点。如果是巡检任务, 则会分解为多个固定路径导航任务, 并显示当前固定路径导航任务中已经经过的中间点。	是
unfinished_path	array[string]	当前任务路径上尚未经过的站点, 为站点的数组, 仅当 task_type 为 3 或 4 时该字段有效。如果是固定路径导航任务, 这里会列出所有尚未经过的中间点。如果是巡检任务, 则会分解为多个固定路径导航任务, 并显示当前固定路径导航任务中尚未经过的中间点。	是
reloc_status	number	0 = FAILED(重定位失败), 1 = SUCCESS(重定位正确), 2 = RELOCING(正在重定位), 3=COMPLETED(重定位完成)	否

loadmap_status	number	0 = FAILED(载入地图失败), 1 = SUCCESS(载入地图成功), 2 = LOADING(正在载入地图)	否
slam_status	number	0 = 没有扫图, 1 = 正在扫图, 2 = 正在实时扫图	否
fatals	array[object]	告警码 Fatal的数组, 所有出现的 Fatal 告警都会出现在数据中	是
errors	array[object]	告警码 Error的数组, 所有出现的 Error 告警都会出现在数据中	是
warnings	array[object]	告警码 Warning的数组, 所有出现的 Warning 告警都会出现在数据中	是
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

查询批量数据2

请求

- 编号: 1101 (0x044D)
- 名称: robot_status_all2_req
- 描述: 查询批量数据2
- JSON 数据区: 无

all2 是为了通过一个请求批量获取更新比较频繁或比较实时的数据(如位置, 速度等), 包括 `1003`, `1004`, `1005`, `1006`, `1008`, `1009`, `1010`, `1011`, `1012`, `1013`, `1020`, `1050`

请求示例

略

响应

- 编号: 11101 (0x2B5D)
- 名称: robot_status_all2_res
- 描述: 查询批量数据2的响应
- JSON 数据区: 见下表

JSON 数据区会包含 `11003`, `11004`, `11005`, `11006`, `11008`, `11009`, `11010`, `11011`, `11012`, `11013`, `11020`, `11050` 的所有字段

字段名	类型	描述	可缺省
mode	number	0 表示手动模式, 1 表示自动模式	否
x	number	机器人的 x 坐标, 单位 m	否
y	number	机器人的 y 坐标, 单位 m	否
angle	number	机器人的 angle 坐标, 单位 rad	否
confidence	number	机器人的定位置信度, 范围 [0,1]	否
vx	number	机器人在机器人坐标系的 x 方向速度, 单位 m/s	否
vy	number	机器人在机器人坐标系的 y 方向速度, 单位 m/s	否
w	number	机器人在机器人坐标系的角速度, 单位 rad/s	否
blocked	boolean	机器人是否被阻挡	否
block_reason	number	被阻挡的原因, 0 = Ultrasonic (超声检测到被阻挡), 1 = Laser (激光检测到被阻挡), 2 = Fallingdown (防跌落传感器检测到被阻挡), 3 = Collision (碰撞传感器检测到被阻挡)	是
block_x	number	阻挡位置的 x 坐标, 单位 m	是
block_y	number	阻挡位置的 y 坐标, 单位 m	是
brake	boolean	机器人是否抱闸, 如果该字段不存在, 说明机器人没有抱闸功能	是
laser_beams	array[array[number]], array 中每个元素也是 array, 其包含两个 number, 分别为 x, y 代表激光点在世界坐标系中的坐标	激光点数据, 激光采样步长为 NetProtocol 中 laserStep 的值	否
	array[array[number]],		

path	array 中每个元素也是 array, 其包含两个 number, 分别为 x, y 代表路径点在世界坐标系中的坐标	按顺序的路径点数据, 依次连接就是机器人规划的路径	否
area_ids	array[string]	机器人所在区域 id 的数组(由于地图上的区域是可以重叠的, 所以机器人可能同时在多个区域)，数组可能为空	否
emergency	boolean	true 表示急停按钮处于激活状态(按下), false 表示急停按钮处于非激活状态(拔起)	是
DI	array[boolean]	DI 数据, boolean 表示高低电平, 从 0 开始, 默认为 true (高电平)	是
DO	array[boolean]	DO 数据, boolean 表示高低电平, 从 0 开始, 默认为 true (高电平)	是
task_status	number	0 = NONE, 1 = WAITING, 2 = RUNNING, 3 = SUSPENDED, 4 = COMPLETED, 5 = FAILED, 6 = CANCELED	否
task_type	number	任务类型, 0 = 没有任务, 1 = 自由导航到任意点, 2 = 自由导航到站点, 3 = 固定路径导航到站点, 4 = 其他	否
target_id	string	当前任务要去的站点, 仅当 task_type 为 2 或 3 时该字段有效, task_status 为 RUNNING 时说明正在去这个站点, task_status 为 COMPLETED 时说明已经到达这个站点, task_status 为 FAILED 时说明去这个站点失败, task_status 为 SUSPENDED 说明去这个站点的任务暂停	是
target_point	array[number]	当前任务要去的坐标点, 为一个包含三个元素的数组, 分别为在世界坐标系中的 x, y, r 坐标, 仅当 task_type 为 1 时该字段有效, task_status 为 RUNNING 时说明正在去这个坐标点, task_status 为 COMPLETED 时说明已经到达这个坐标点, task_status 为 FAILED 时说明去这个坐标点失败, task_status 为 SUSPENDED 说明去这个坐标点的任务暂停	是
finished_path	array[string]	当前任务路径上已经经过的站点, 为站点的数组, 仅当 task_type 为 3 或 4 时该字段有效点。如果是固定路径导航任务, 这里会列出所有已经经过的中间点。如果是巡检任务, 则会分解为多个固定路径导航任务, 并显示当前固定路径导航任务中已经经过的中间点。	是
unfinished_path	array[string]	当前任务路径上尚未经过的站点, 为站点的数组, 仅当 task_type 为 3 或 4 时该字段有效。如果是固定路径导航任务, 这里会列出所有尚未经过的中间点。如果是巡检任务, 则会分解为多个固定路径导航任务, 并显示当前固定路径导航任务中尚未经过的中间点。	是
fatals	array[object]	告警码 Fatal的数组, 所有出现的 Fatal 告警都会出现在数据中	是
errors	array[object]	告警码 Error的数组, 所有出现的 Error 告警都会出现在数据中	是
warnings	array[object]	告警码 Warning的数组, 所有出现的 Warning 告警都会出现在数据中	是
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

查询批量数据3

请求

- 编号: 1102 (0x044E)
- 名称: robot_status_all3_req
- 描述: 查询批量数据3
- JSON 数据区: 无

all3 是为了通过一个请求批量获取更新不频繁的状态数据, 包括 `1002`, `1007`, `1021`, `1022`, `1025`

请求示例

略

响应

- 编号: 11102 (0x2B5E)
- 名称: robot_status_all3_res
- 描述: 查询批量数据3的响应
- jJSONson 数据区: 见下表

JSON 数据区会包含 `11002`, `11007`, `11021`, `11022`, `11025` 的所有字段

字段名	类型	描述	可缺省
odo	number	累计行驶里程, 单位 m	否
time	number	本次运行时间, 单位 ms	否
total_time	number	累计运行时间, 单位 ms	否
controller_temp	number	控制器温度, 单位 ℃	是
controller_humi	number	控制器湿度, 单位 %	是
controller_voltage	number	控制器电压, 单位 V	是
battery_level	number	机器人电池电量, 范围 [0, 1]	是
battery_temp	number	机器人电池温度, 单位 ℃	是
charging	boolean	电池是否正在充电	是
voltage	number	电压, 单位 V	是
current	number	电流, 单位 A	是
reloc_status	number	0 = FAILED(重定位失败), 1 = SUCCESS(重定位正确), 2 = RELOCING(正在重定位), 3=COMPLETED(重定位完成)	否
loadmap_status	number	0 = FAILED(载入地图失败), 1 = SUCCESS(载入地图成功), 2 = LOADING(正在载入地图)	否
slam_status	number	0 = 没有扫图, 1 = 正在扫图, 2 = 正在实时扫图	否
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

查询机器人初始化状态

请求

- 编号: 1111 (0x0457)
- 名称: robot_status_init_req
- 描述: 查询机器人初始化状态
- JSON 数据区: 无

请求示例

略

响应

- 编号: 11111 (0x2B67)
- 名称: robot_status_init_res
- 描述: 查询机器人初始化状态的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
init_status	number	0 = FAILED(初始化失败), 1 = SUCCESS(初始化成功), 2 = INITING(正在初始化)	否
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

注意***:*** 机器人启动后会有一定的初始化时间，通过 ***(1111)*** 可查询机器人的初始化状态，当初始化状态为 ***SUCCESS*** 时说明初始化已完成。初始化未完成时，控制 ***API*** 和任务 ***API*** 不可使用

响应示例

略

查询机器人载入的地图以及储存的地图

请求

- 编号: 1300 (0x0514)
- 名称: robot_status_map_req
- 描述: 查询机器人载入的地图以及储存的地图
- JSON 数据区: 无

请求示例

略

响应

- 编号: 11300 (0x2C24)
- 名称: robot_status_map_res
- 描述: 查询机器人载入的地图以及储存的地图的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
current_map	string	当前载入的地图	否
maps	array[string]	所有储存在机器人上的地图名组成的数组	否
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

数据示例:

```
{
    "current_map": "301",
    "maps": ["301", "302", "303"]
}
```

响应示例

略

查询机器人当前载入地图中的站点信息

该 API 用于获得地图中所有站点的坐标、角度以及类型信息

请求

- 编号: 1301 (0x0515)
- 名称: robot_status_station_req
- 描述: 查询机器人当前载入地图中的站点信息
- JSON 数据区: 无

请求示例

略

响应

- 编号: 11301 (0x2C25)
- 名称: robot_status_station_res
- 描述: 查询机器人当前载入地图中的站点信息的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
stations	array[object]	站点数组, 若地图中没有站点, 则为空数组, object 形式见下文	否
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

object 形式如下:

```
{
    "id": "LM1",         // 站点的ID
    "type": "LandMark", // 站点的类型 LandMark, RobotHome, ChargePoint, GyrocaliPoint, ReturnPoint
    "x": 1.23,           // 站点在世界坐标系中的 x 坐标(m)
    "y": 4.56,           // 站点在世界坐标系中的 y 坐标(m)
    "r": 1.57            // 站点在世界坐标系中的朝向角(rad)
}
```

数据示例:

```
{
    "stations": [
        {
            "id": "LM1",
            "type": "LandMark",
            "x": 1.23,
            "y": 4.56,
            "r": 1.57
        },
        {
            "id": "LM2",
            "type": "LandMark",
            "x": -1.23,
            "y": 4.56,
            "r": 3.14
        },
        {
            "id": "GP3",
            "type": "GyrocaliPoint",
            "x": -1.23,
            "y": -4.56,
            "r": 0
        }
    ],
}
```


响应示例

略

查询机器人参数

请求

- 编号: 1400 (0x0578)
- 名称: robot_status_params_req
- 描述: 查询机器人参数信息
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
plugin	string	参数所属的插件名, 如果缺省, 表示查询所有插件的所有参数	是
param	string	参数名, 如果 plugin 存在，但 param 缺省，代表查询该插件的所有参数。否侧查询该插件的指定参数	是

数据示例:

如下表示查询 MoveFactory 插件的 MaxAcc 参数

```
{
    "plugin": "MoveFactory",
    "param": "MaxAcc"
}
```

请求示例

略

响应

- 编号: 11400 (0x2C88)
- 名称: robot_status_params_res
- 描述: 查询机器人参数信息的响应
- JSON 数据区: 见下

A: 如果请求中 **plugin** 与 **param** 均不缺省, 表示查询特定插件的某个参数, 下面以 **MoveFactory** 插件为例说明数据区的格式

plugin = "MoveFactory", param = "MaxAcc", 查询 MoveFactory 插件的 MaxAcc 参数, 响应数据区如下:

```
{
    "MoveFactory": { // 查询成功的情况下才会有, 插件不存在, 参数不存在或发生其他错误时请看 ret_code
        "MaxAcc": {
            "value": 0.5    // value 字段表示其数值, 数值类型随参数的不同而不同, 各个参数的数值类型参阅附录三
        }
    },
    "ret_core": 0, // 可缺省
    "err_msg": ""  // 可缺省
}
```

B: 如果请求中 **plugin** 不缺省而 **param** 缺省, 表示查询特定插件的所有参数, 下面以 **NetProtocol** 插件为例说明数据区的格式

plugin = "NetProtocol", param 缺省或 param="", 查询 NetProtocol 插件的所有参数:

```
{
    "NetProtocol": { // 查询成功的情况下才会有, 插件不存在或发生其他错误时请看 ret_code
        // NetProtocol 有两个参数分别为 laserStep, timeOut
        "laserStep": {
            "value": 5
        },
        "timeOut": {
            "value": 0
        }
    }
    "ret_core": 0, // 可缺省
    "err_msg": ""  // 可缺省
}
```

C: 如果请求中 **plugin** 缺省, 表示查询所有参数

plugin 缺省或 plugin="", 查询所有参数:

```
{
    // 若不出错, 则会以如下形式列出所有的插件和其运行时包含的所有参数, 这里只是举例, 实际运行时会有很多插件及参数, 具体的插件及参数说明请参阅附录三
    "NetProtocol": {
        "laserStep": {
            "value": 5
        },
        "timeOut": {
            "value": 0
        }
    },
    "MoveFactory": {
        "MaxAcc": {
            "value": 0.5
        },
        "MaxSpeed": {
            "value": 0.3
        },
        "MaxRotAcc": {
            "value": 60
        },
        "MaxRotSpeed": {
            "value": 90
        }
        // ...
    }
    // ....
    "ret_core": 0, // 可缺省
    "err_msg": ""  // 可缺省
}
```

响应示例

略

机器人控制 API

机器人控制 API 手动模式下才有效。

注 *1:* 机器人启动后默认为手动模式

注 *2:* 确认定位确认 *(2003)* 必须在机器人初始化状态 *(1111)* 为 *SUCCESS*, 机器人载入地图状态 *(1022)* 为 *SUCCESS*, 机器人重定位状态 *(1021)* 为 *COMPLETED* 的情况下才能使用，确认成功后，重定位状态 *(1021)* 将会变为 *SUCCESS*, 此时才可以将模式切换为自动模式 *(4000)*

API 列表

请求

编号	名称	描述
2000	robot_control_stop_req	停止运动
2001	robot_control_gyrocal_req	标定陀螺仪
2002	robot_control_reloc_req	重定位
2003	robot_control_comfirmloc_req	确认定位正确
2010	robot_control_motion_req	开环运动
2020(DEPRECATED)	robot_control_slam_req	开始扫地图
2021(DEPRECATED)	robot_control_endslam_req	停止扫地图
2022	robot_control_loadmap_req	切换载入的地图
2023	robot_control_loadmapobj_req	重新加载地图中的元素
…	…	…

响应

机器人控制 API 的响应会立即返回给客户端，并不会等待机器人置信完毕。若 ret_code 不存在，说明指令将被执行，但并不表示指令已经执行完毕了，对于有些控制指令，如(2022) 切换地图，可以通过 (1022) 查询载入地图的状态。其他大部分指令都是瞬时的，用户不必关心机器人什么时候执行完毕，只需要知道机器人将执行指令即可。

编号	名称	描述
12000	robot_control_stop_res	对 2000 请求的响应
12001	robot_control_gyrocal_res	对 2001 请求的响应
12002	robot_control_reloc_res	对 2002 请求的响应
12003	robot_control_comfirmloc_res	对 2003 请求的响应
12010	robot_control_motion_res	对 2010 请求的响应
12020(DEPRECATED)	robot_control_slam_res	对 2020 请求的响应
12021(DEPRECATED)	robot_control_endslam_res	对 2021 请求的响应
12022	robot_control_loadmap_res	对 2022 请求的响应
12023	robot_control_loadmapobj_res	对 2023 请求的响应
…	…	…

停止运动

请求

- 编号: 2000 (0x07D0)
- 名称: robot_control_stop_req
- 描述: 停止运动
- JSON 数据区: 无

请求示例

略

响应

- 编号: 12000 (0x2EE0)
- 名称: robot_control_stop_res
- 描述: 停止运动的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

标定陀螺仪

请求

- 编号: 2001 (0x07D1)
- 名称: robot_control_gyrocal_req
- 描述: 标定陀螺仪
- JSON 数据区: 无

请求示例

略

响应

- 编号: 12001 (0x2EE1)
- 名称: robot_control_gyrocal_res
- 描述: 标定陀螺仪的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

重定位

请求

- 编号: 2002 (0x07D2)
- 名称: robot_control_reloc_req
- 描述: 重定位
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
x	number	世界坐标系中的 x 坐标, 单位 m	否
y	number	世界坐标系中的 y 坐标, 单位 m	否
angle	number	世界坐标系中的角度(若缺省，表示在 2π 弧度范围内进行定位, 可能比指定角度的重定位慢), 单位 rad	是

请求示例

略

响应

- 编号: 12002 (0x2EE2)
- 名称: robot_control_reloc_res
- 描述: 重定位的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

确认定位正确

请求

- 编号: 2003 (0x07D3)
- 名称: robot_control_comfirmloc_req
- 描述: 确认定位正确
- JSON 数据区: 无

请求示例

略

响应

- 编号: 12003 (0x2EE3)
- 名称: robot_control_comfirmloc_res
- 描述: 确认定位正确的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

开环运动

请求

- 编号: 2010 (0x07DA)
- 名称: robot_control_motion_req
- 描述: 开环运动
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
vx	number	机器人在机器人坐标系中的 x 轴方向速度, 若缺省则认为是 0, 单位 m/s	是
vy	number	机器人在机器人坐标系中的 y 轴方向速度, 若缺省则认为是 0, 单位 m/s	是
w	number	机器人在机器人坐标系中的角速度(即顺时针转为负, 逆时针转为正), 若缺省则认为是 0, 单位 rad/s	是

请求示例

略

响应

- 编号: 12010 (0x2EEA)
- 名称: robot_control_motion_res
- 描述: 开环运动的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

开始扫地图

该 API 已标记为 DEPRECATED, 将在未来的某个版本中移除

请求

- 编号: 2020 (0x07E4)
- 名称: robot_control_slam_req
- 描述: 开始扫地图
- JSON 数据区: 无

请求示例

略

响应

- 编号: 12020 (0x2EF4)
- 名称: robot_control_slam_res
- 描述: 开始扫地图的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

停止扫地图

该 API 已标记为 DEPRECATED, 将在未来的某个版本中移除

请求

- 编号: 2021 (0x07E5)
- 名称: robot_control_endslam_req
- 描述: 停止扫地图
- JSON 数据区: 无

请求示例

略

响应

- 编号: 12021 (0x2EF5)
- 名称: robot_control_endslam_res
- 描述: 停止扫地图的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

切换载入的地图

请求

- 编号: 2022 (0x07E6)
- 名称: robot_control_loadmap_req
- 描述: 切换载入的地图
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
map_name	string	要切换的地图名(不能包含中文等非法字符, 只能使用 0-9, a-z, A-Z, -, _)	否

请求示例

略

响应

- 编号: 12022 (0x2EF6)
- 名称: robot_control_loadmap_res
- 描述: 切换载入的地图的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

重新加载地图中的元素

请求

- 编号: 2023 (0x07E7)
- 名称: robot_control_loadmapobj_req
- 描述: 重新加载当前地图中的元素
- JSON 数据区: 无

请求示例

略

响应

- 编号: 12023 (0x2EF7)
- 名称: robot_control_loadmapobj_res
- 描述: 重新加载当前地图中的元素的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

机器人任务 API

机器人任务 API 自动模式下才有效。

API 列表

请求

编号	名称	描述
3001	robot_task_pause_req	暂停当前任务
3002	robot_task_resume_req	继续当前任务
3003	robot_task_cancel_req	取消当前任务
3050	robot_task_gopoint_req	自由导航(根据地图上的坐标值或站点自由规划路径导航)
3051	robot_task_gotarget_req	固定路径导航(根据地图上站点及固定路径导航)
3052	robot_task_patrol_req	巡检(设定路线进行固定路径导航)
3055	robot_task_translate_req	平动，以固定速度直线运动固定距离
3056	robot_task_turn_req	转动，以固定角速度旋转固定角度
…	…	…

注***: 3055*** 和 ***3056*** 不能同时进行

响应

机器人任务 API 的会立即响应给客户端，不会等待机器人执行任务。若 ret_code 不存在或为 0，说明任务将被执行，但并不表示任务已经执行完毕了，用户需要通过机器人状态 API (1020) 查询任务的状态。机器人某一时刻只能有一个任务，新的任务指令将会打断之前的任务。

编号	名称	描述
13001	robot_task_pause_res	对 3001 请求的响应
13002	robot_task_resume_res	对 3002 请求的响应
13003	robot_task_cancel_res	对 3003 请求的响应
13050	robot_task_gopoint_res	对 3050 请求的响应
13051	robot_task_gotarget_res	对 3051 请求的响应
13052	robot_task_patrol_res	对 3052 请求的响应
13055	robot_task_translate_req	对 3055 请求的响应
13056	robot_task_turn_req	对 3056 请求的响应
…	…	…

暂停当前任务

请求

- 编号: 3001 (0x0BB9)
- 名称: robot_task_pause_req
- 描述: 暂停当前任务
- JSON 数据区: 无

请求示例

略

响应

- 编号: 13001 (0x32C9)
- 名称: robot_task_pause_res
- 描述: 暂停当前任务的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

继续当前任务

请求

- 编号: 3002 (0x0BBA)
- 名称: robot_task_resume_req
- 描述: 继续当前任务
- JSON 数据区: 无

请求示例

略

响应

- 编号: 13002 (0x32CA)
- 名称: robot_task_resume_res
- 描述: 继续当前任务的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

取消当前任务

请求

- 编号: 3003 (0x0BBB)
- 名称: robot_task_cancel_req
- 描述: 取消当前任务
- JSON 数据区: 无

请求示例

略

响应

- 编号: 13003 (0x32CB)
- 名称: robot_task_cancel_res
- 描述: 取消当前任务的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

自由导航

自由导航为给定目标站点或这地图坐标点, 由机器人自动规划路径到达, 此时的路径是不会沿着地图上的贝塞尔曲线。

请求

- 编号: 3050 (0x0BEA)
- 名称: robot_task_gopoint_req
- 描述: 自由导航
- JSON 数据区: 见下表

注意 ***1: id*** 字段若不缺省, 则认为是导航到站点, 此时会忽略 ***x, y*** 字段。若 ***id*** 字段缺省, 则会认为是导航到坐标点, 此时需要保证 ***x, y*** 均不缺省, 否则会返回错误。

注意 ***2: angle*** 为到点朝向, 如果缺省则对于导航到站点会使用站点上设置的值, 对于导航到坐标点会认为是 ***0***。

注意 ***3: max_speed, max_wspeed, max_acc, max_wacc*** 字段不建议使用, 最好通过地图上的路径和区域属性或机器人参数来控制这些值, 并且这些字段可能在将来被删除

字段名	类型	描述	可缺省
id	string	自由导航目标站点的 id	是
x	number	自由导航目标点(世界坐标系)的 x 坐标值, 单位 m	是
y	number	自由导航目标点(世界坐标系)的 y 坐标值, 单位 m	是
angle	number	自由导航目标点(世界坐标系)的角度值, 单位 rad	是
max_speed	number	最大速度, 单位 m/s	是
max_wspeed	number	最大角速度, 单位 rad/s	是
max_acc	number	最大加速度, 单位 m/s^2	是
max_wacc	number	最大角加速度, 单位 rad/s^2	是

请求示例

略

响应

- 编号: 13050 (0x32FA)
- 名称: robot_task_gopoint_res
- 描述: 自由导航的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

固定路径导航

固定路径导航为给定目标站点, 由机器人自动规划沿着贝塞尔曲线的路径运行, 期间会经过其它中间站点但不会停留。

请求

- 编号: 3051 (0x0BEB)
- 名称: robot_task_gotarget_req
- 描述: 固定路径导航
- JSON 数据区: 见下表

注意 ***1: angle*** 为到点朝向**,** 如果缺省则会使用站点上设置的值。

注意 ***2: max_speed, max_wspeed, max_acc, max_wacc*** 字段不建议使用**,** 最好通过地图上的路径和区域属性或机器人参数来控制这些值**,** 并且这些字段可能在将来被删除

字段名	类型	描述	可缺省
id	string	自由导航目标站点的 id	否
angle	number	自由导航目标点(世界坐标系)的角度值, 单位 rad	是
max_speed	number	最大速度, 单位 m/s	是
max_wspeed	number	最大角速度, 单位 rad/s	是
max_acc	number	最大加速度, 单位 m/s^2	是
max_wacc	number	最大角加速度, 单位 rad/s^2	是

请求示例

略

响应

- 编号: 13051 (0x32FB)
- 名称: robot_task_gotarget_res
- 描述: 固定路径导航的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

巡检

巡检为给定一组目标站点，由机器人按顺序依次经过目标站点，整个过程沿着地图上的贝塞尔曲线。

请求

- 编号: 3052 (0x0BEC)
- 名称: robot_task_patrol_req
- 描述: 巡检
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
route	string	巡检的的路线名	否
loop	boolean	是否循环巡检，即到达路线上最后一个站点后是否回到第一个站点重新巡检，若缺省，则为否	是

请求示例

略

响应

- 编号: 13052 (0x32FC)
- 名称: robot_task_patrol_res
- 描述: 巡检的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

平动

请求

- 编号: 3055 (0x0BEF)
- 名称: robot_task_translate_req
- 描述: 平动，以固定速度直线运动固定距离
- JSON 数据区: 见下表

注意***: 3055(***平动***)*** 和 ***3056(***转动***)*** 不能同时进行

字段名	类型	描述	可缺省
dist	number	直线运动距离, 绝对值, 单位 m	否
vx	number	直线运动的速度, 正为向前，负为向后, 单位 m/s	否
mode	number	0 = 里程模式(根据里程进行运动), 1 = 自定位模式(根据机器人自定位进行运动), 若缺省则默认为里程模式	是

里程模式不需要定位精准, 但是对于长距离的运动可能会产生较大误差, 误差随距离的增大而增大。

自定位模式需要当前环境定位稳定, 误差与当前环境和定位精度相关。(自定位模式目前不可用)

请求示例

略

响应

- 编号: 13055 (0x32FF)
- 名称: robot_task_translate_res
- 描述: 平动，以固定速度直线运动固定距离的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

转动

请求

- 编号: 3056 (0x0BF0)
- 名称: robot_task_turn_req
- 描述: 转动, 以固定角速度旋转固定角度
- JSON 数据区: 见下表

注意***: 3055(***平动***)*** 和 ***3056(***转动***)*** 不能同时进行

字段名	类型	描述	可缺省
angle	number	转动的角度(机器人坐标系), 绝对值, 单位 rad, 可以大于 2π	否
vw	number	转动的角速度(机器人坐标系), 正为逆时针转, 负为顺时针转 单位 rad/s	否
mode	number	0 = 里程模式(根据里程进行运动), 1 = 自定位模式(根据机器人自定位进行运动), 若缺省则默认为里程模式	是

里程模式不需要定位精准, 但是对于角度很大的转动动可能会产生较大误差, 误差随旋转角度的增大而增大。

自定位模式需要当前环境定位稳定, 误差与当前环境和定位精度相关。(自定位模式目前不可用)

请求示例

略

响应

- 编号: 13056 (0x3300)
- 名称: robot_task_turn_res
- 描述: 转动, 以固定角速度旋转固定角度的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

机器人配置 API

机器人配置 API 用于配置机器人的参数, 切换运行模式以及地图等。

API 列表

请求

编号	名称	描述
4000	robot_config_mode_req	切换运行模式(手动, 自动)
4100	robot_config_setparams_req	配置参数
4101	robot_config_saveparams_req	配置并保存参数
4102	robot_config_reloadparams_req	重载参数
4300	robot_config_clearfatal_req	清除机器人的 Fatal 错误码
…	…	…

响应

编号	名称	描述
14000	robot_config_mode_res	对 4000 请求的响应
14100	robot_config_setparams_res	对 4100 请求的响应
14101	robot_config_saveparams_res	对 4101 请求的响应
14102	robot_config_reloadparams_res	对 4102 请求的响应
14300	robot_config_clearfatal_res	对 4300 请求的响应
…	…	…

切换运行模式

请求

- 编号: 4000 (0x0FA0)
- 名称: robot_config_mode_req
- 描述: 切换运行模式
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
mode	number	0 = 手动模式, 1 = 自动模式	否

请求示例

略

响应

- 编号: 14000 (0x36B0)
- 名称: robot_config_mode_res
- 描述: 切换运行模式的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

配置参数

请求

- 编号: 4100 (0x1004)
- 名称: robot_config_setparams_req
- 描述: 配置参数
- JSON 数据区: 见下

通过一个 JSON 对象可以一次性配置一个或多个参数, 格式如下:

```
{
    // 这里配置了两个插件的 3 个参数
    "MoveFactory": {
        "MaxAcc": 1.0,  // 这里的数值类型根据不同的参数而改变, 具体的参数说明请参阅[附录三]
        "MaxSpeed": 0.8
        // ...
    },
    "NetProtocol": {
        "timeOut": 10
        // ...
    }
    // ... 可以配置更多的参数
}
```

```
{
    // 这里只配置了MoveFactory插件的 MaxAcc 参数
    "MoveFactory": {
        "MaxAcc": 1.0
    }
}
```

请求示例

略

响应

- 编号: 14100 (0x3714)
- 名称: robot_config_setparams_res
- 描述: 配置参数的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

配置并保存参数

请求

- 编号: 4101 (0x1005)
- 名称: robot_config_saveparams_req
- 描述: 配置并保存参数
- JSON 数据区: 见下

配置并保存参数的格式和配置参数的格式相同

通过一个 JSON 对象可以一次性配置并保存一个或多个参数, 格式如下:

```
{
    // 这里配置并保存了两个插件的 3 个参数
    "MoveFactory": {
        "MaxAcc": 1.0,  // 这里的数值类型根据不同的参数而改变, 具体的参数说明请参阅[附录三]
        "MaxSpeed": 0.8
        // ...
    },
    "NetProtocol": {
        "timeOut": 10
        // ...
    }
    // ... 可以配置并保存更多的参数
}
```

```
{
    // 这里只配置并保存了MoveFactory插件的 MaxAcc 参数
    "MoveFactory": {
        "MaxAcc": 1.0
    }
}
```

请求示例

略

响应

- 编号: 14101 (0x3715)
- 名称: robot_config_saveparams_res
- 描述: 配置并保存参数的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

重载参数

请求

- 编号: 4102 (0x1006)
- 名称: robot_config_reloadparams_req
- 描述: 重载参数, 恢复参数的出厂默认值
- JSON 数据区: 见下

通过一个 JSON 数组对象可以一次性重载一个或多个参数, 格式如下:

```
// 注意这是一个 JSON Array, 而不是 JSON Object
[
{
    // 重载 MoveFactory 的 MaxAcc 和 MaxSpeed 参数
    "plugin": "MoveFactory",    // 如果 plugin 缺省或 plugin="" 则这个元素对象无效
    "params": ["MaxAcc", "MaxSpeed"] // 如果 params 缺省或为空数组, 代表重载该插件的所有参数, 否则重载该插件的指定参数
}, {
    // 重载 NetProtocol 的所有参数
    "plugin": "NetProtocol",
    "params": []
}
// ... 可以重载更多参数
]
```

```
// 一个空的数组代表重载所有插件的所有参数
[]
```

请求示例

略

响应

- 编号: 14102 (0x3716)
- 名称: robot_config_reloadparams_res
- 描述: 重载参数的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

清除机器人的 **Fatal** 错误码

请求

- 编号: 4300 (0x10CC)
- 名称: robot_config_clearfatal_req
- 描述: 清除机器人所有的 Fatal 错误码
- JSON 数据区: 无

请求示例

略

响应

- 编号: 14300 (0x37DC)
- 名称: robot_config_clearfatal_res
- 描述: 清除机器人所有的 Fatal 错误码的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

机器人核心 API

机器人核心 API 用于对机器人进行重启关机或重置固件等操作，请谨慎使用。

API 列表

请求

编号	名称	描述
5000	robot_core_shutdown_req	关闭机器人, 机器人将断电并失去控制
5001	robot_core_end_req	停止机器人程序, 机器人将停止工作, 但不会断电
5002	robot_core_start_req	启动机器人程序
5003	robot_core_reboot_req	重启机器人, 重启期间连接将断开
5004	robot_core_restart_req	重启机器人程序, 但不会重启机器人
5005	robot_core_resetdsp_req	重置机器人固件
…	…	…

响应

编号	名称	描述
15000	robot_core_shutdown_res	对5000请求的响应
15001	robot_core_end_res	对5001请求的响应
15002	robot_core_start_res	对5002请求的响应
15003	robot_core_reboot_res	对5003请求的响应
15004	robot_core_restart_res	对5004请求的响应
15005	robot_core_resetdsp_res	对5005请求的响应
…	…	…

关闭机器人

请求

- 编号: 5000 (0x1388)
- 名称: robot_core_shutdown_req
- 描述: 关闭机器人, 机器人将断电并失去控制
- JSON 数据区: 无

请求示例

略

响应

- 编号: 15000 (0x3A98)
- 名称: robot_core_shutdown_res
- 描述: 关闭机器人的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

停止机器人程序

请求

- 编号: 5001 (0x1389)
- 名称: robot_core_end_req
- 描述: 停止机器人程序, 机器人将停止工作, 但不会断电
- JSON 数据区: 无

请求示例

略

响应

- 编号: 15001 (0x3A99)
- 名称: robot_core_end_res
- 描述: 停止机器人程序的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

启动机器人程序

请求

- 编号: 5002 (0x138A)
- 名称: robot_core_start_req
- 描述: 启动机器人程序
- JSON 数据区: 无

请求示例

略

响应

- 编号: 15002 (0x3A9A)
- 名称: robot_core_start_res
- 描述: 启动机器人程序的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

重启机器人

请求

- 编号: 5003 (0x138B)
- 名称: robot_core_reboot_req
- 描述: 重启机器人, 重启期间连接将断开
- JSON 数据区: 无

请求示例

略

响应

- 编号: 15003 (0x3A9B)
- 名称: robot_core_reboot_req
- 描述: 重启机器人的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

重启机器人程序

请求

- 编号: 5004 (0x138C)
- 名称: robot_core_restart_req
- 描述: 重启机器人程序
- JSON 数据区: 无

请求示例

略

响应

- 编号: 15004 (0x3A9C)
- 名称: robot_core_restart_res
- 描述: 重启机器人程序的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

重置机器人固件

请求

- 编号: 5005 (0x138D)
- 名称: robot_core_resetdsp_req
- 描述: 重置机器人固件
- JSON 数据区: 无

机器人固件为驱动机器人运动相关硬件的程序, 重置固件会使机器人暂时停止运动, 并切换为手动模式且不响应任何控制和任务指令。

请求示例

略

响应

- 编号: 15005 (0x3A9D)
- 名称: robot_core_resetdsp_res
- 描述: 重置机器人固件的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

其他 API

其他 API 包含各种杂项，如喇叭控制等。

API 列表

请求

编号	名称	描述
6000	robot_other_speaker_req	喇叭控制
6001	robot_other_setdo_req	设置 DO
…	…	…

响应

编号	名称	描述
16000	robot_other_speaker_res	对 6000 请求的响应
16001	robot_other_setdo_res	对 6001 请求的响应
…	…	…

喇叭控制

请求

- 编号: 6000 (0x1770)
- 名称: robot_other_speaker_req
- 描述: 喇叭控制, 播放指定的音频文件
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
name	string	需要播放的音频文件名	否

请求示例

略

响应

- 编号: 16000 (0x3E80)
- 名称: robot_other_speaker_res
- 描述: 喇叭控制的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

设置 **DO**

该 API 用于设置机器人控制器上的 DO (Digital Output) 状态

请求

- 编号: 6001 (0x1771)
- 名称: robot_other_setdo_req
- 描述: 设置 DO
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
id	number	DI 的 id 号, 从 0 开始	否
status	boolean	true 为高电平, false 为低电平	否

请求示例

略

响应

- 编号: 16001 (0x3E81)
- 名称: robot_other_setdo_res
- 描述: 设置 DO 的响应
- JSON 数据区: 见下表

字段名	类型	描述	可缺省
ret_code	number	API 错误码	是
err_msg	string	错误信息	是

响应示例

略

对错误请求的响应

对错误请求的响应是当请求的报文头部格式解析正确，但是头部内容或者数据区内容解析错误时产生的响应。 编号会填入响应报文头部的报文类型(编号)字段中。

编号	名称	描述
60000 (0xEA60)	robot_error_wtype_res	错误的报文类型，若用户将某类型的报文发错了端口将得到这个响应
60001 (0xEA61)	robot_error_utype_res	未知的报文类型, 报文类型号未在上文中定义
60002 (0xEA62)	robot_error_data_res	错误的数据区，当数据区无法反序列化为 JSON 对象时将得到这个响应
60003 (0xEA63)	robot_error_version_res	协议版本错误时得到的响应
60004 (0xEA64)	robot_error_hugedata_res	数据区过大，服务器会主动断连接，限制 10M
…	…	…

注意：对错误请求的响应数据区均为空，不会有 ***ret_code*** 和 ***err_msg***

响应示例

假设请求的协议版本错误, 包头中的编号为 0x0001, 那么会得到 60003 (0xEA63) 的响应, 如下:

```
5A 01 00 01 00 00 00 00 EA 63 00 00 00 00 00 00
```

机器人告警码

机器人告警码用于指示机器人当前的错误告警信息，在机器人状态 API 中 1050 号可以查询到这些告警码，若 1050 号请求的 11050 号响应报文中各类告警码为空数组，说明没有错误告警。 告警码共分为 3 类, 分别为 Fatal, Error, Warning, 告警级别依次降低。

各个告警码的具体见下表:

Fatal 告警码

告警码	名称	描述
50000	internal error	内部错误
50001	resource exhaustion	内部资源耗尽
50002	lua error	内部 lua 调用出错
50100	map parse error	地图解析出错
50101	map load error	无法加载地图
50102	map is too large	地图过大
50103	map is empty	地图数据为空
50104	map meta error	地图元数据错误
50105	map resolution is illegal	地图分辨率非法
50106	map format invaild	地图格式非法
50200	move skills path error	错误的 moveskills 路径
50201	move skills init error	moveskills 初始化错误
50202	move skills register error	moveskills 注册出错
…	…	…

Error 告警码

告警码	名称	描述
52100	laser error	失去与激光设备的通讯
52101	laser data invalid	所有激光数据无效
52110	imu error	陀螺仪错误
52111	motor driver connection error	驱动器连接故障
52112	ultera sonic error	超声波雷达故障
52113	RFID reader error	RFID读取故障
52114	magnetic 0 tracker error	磁轨导航传感器故障
52115	magnetic 1 tracker error	磁轨导航传感器故障
52116	controller net down	控制器网络断开
52117	controller link down	控制器连接断开
52118	subsystem error	子系统连接故障
52119	low task frequency	单片机任务频率过低
52130	motor GVDD over voltage	电机GVDD过压
52131	motor FET over current	电机桥臂过流
52132	motor over tempreture	电机驱动器过热
52133	motor VDD under voltege	电机电源欠压
52135	motor error see log	电机其他需要看log的错误
52200	can not find bezier curve	固定路径导航路径规划失败
52201	can not find target id	找不到指定的地图中的站点
52202	path plan failed	自由导航路径规划失败
52203	unsupported skill	不支持的move skill
52204	charge failed and not set prepoint	充电失败但未设置前置点
52210	can not find patrol route	找不到巡检线路
52300	too low confidence of localization	定位的置信度太低
52301	...	...
…	…	…

Warning 告警码

告警码	名称	描述
54000	joystick error	手柄连接出错
54001	battery error	电池通讯出错
54002	temperature sensor error	温度传感器出错
54003	motor overs speed	电机超速
54004	motor emergency stop	电机被拍急停
…	…	…

API 错误码

API 错误码为响应数据区 JSON 对象包含的错误码（`ret_code`），用于指示对请求执行不成功或请求出错等错误。若 `ret_code` 为 0 或缺省，说明没有错误，若不为0，说明发生了错误，错误码见下表:

错误码只会出现在 `JSON` 数据中，编号不会填入响应报文头部的报文类型字段中。

错误码	名称	描述
40000	req_unavailable	请求不可用
40001	param_missing	必要的请求参数缺失
40002	param_type_error	请求参数类型错误
40003	param_illegal	请求参数不合法
40004	mode_error	运行模式错误
40005	illegal_map_name	非法的地图名
40006	programming_dsp	正在烧写固件
40007	program_dsp_error	烧写固件错误
40010	shutdown_error	关机指令出现错误
40011	reboot_error	重启指令出现错误
40050	map_parse_error	地图解析出错
40051	map_not_exists	地图不存在
40052	load_map_error	加载地图错误
40053	load_mapobj_error	重载地图错误
40054	empty_map	空地图
40070	model_parse_error	模型文件解析错误
40071	calibration_parse_error	标定数据解析错误
40100	req_timeout	请求执行超时
40101	req_forbidden	请求被禁止
40102	robot_busy	机器人繁忙
40199	robot_internal_error	内部错误
41000	init_status_error	初始化状态错误
41001	loadmap_status_error	地图载入状态错误
41002	reloc_status_error	重定位状态错误
…	…	…

机器人参数配置指南

Roboshop 中机器人的参数配置说明如下, 下文中未提到的参数请勿随意修改

LaserObstacleDetection

1. MaxDetectionDist

含义: 检测该半径范围内的障碍物, 以机器人轮轴中心为圆心

类型: double

单位: m

典型值: 4

配置建议: 视情况而定, 不能过小, 一般不需要修改

2. MaxReservedCycle

含义: 检测到障碍物后将障碍物原地保留的时间

类型: uint

单位: 帧数, 30 帧大约为 1 秒钟

典型值: 23

配置建议: 视情况而定, 一般不需要修改

MCLoc

1. StartFromHome

含义: 启动机器人时是否认为机器人在 RobotHome 点, 并在该点进行第一次重定位

类型: bool

单位: 无

典型值: false

配置建议: 视情况而定

MoveFactory

1. ManualBlock

含义: 手动模式操控机器人时, 遇到障碍物是否会停止

类型: bool

单位: 无

典型值: true

配置建议: 建议开启; 但某些情况下需要手动操控经过狭窄处, 请关闭此项, 否则机器人可能会无法运动

2. MaxSpeed

含义: 机器人运行时的最大速度默认值, 该值会被地图中的曲线或区域中设定的 maxspeed 所覆盖

类型: double

单位: m/s

典型值: 0.5

配置建议: 视机器人性能而定, 不建议大于 1.0

3. MaxAcc

含义: 机器人运行时的最大加速度默认值, 该值会被地图中的曲线或区域中设定的 maxacc 所覆盖

类型: double

单位: m/s^2

典型值: 0.8

配置建议: 视机器人性能而定, 不建议大于 1.0

4. MaxRot

含义: 机器人运行时的最大角速度默认值, 该值会被地图中的曲线或区域中设定的 maxrot 所覆盖

类型: double

单位: degree/s

典型值: 60

配置建议: 视机器人性能而定, 不建议大于 120

5. MaxRotAcc

含义: 机器人运行时的最大角加速度默认值, 该值会被地图中的曲线或区域中设定的 maxrotacc 所覆盖

类型: double

单位: degree/s^2

典型值: 45

配置建议: 视机器人性能而定, 不建议大于 80

6. SlowdownDist

含义: 机器人到点之前提前减速的距离

类型: double

单位: m

典型值: 0.3

配置建议: 情况而定, 提前减速有助于提高到点精度。机器人运行速度越快需要提前减速的距离就越长

NetProtocol

1. laserStep

含义: 通过网络协议获取激光束的密度(步长), 即每隔多少束激光取一束激光, 该值可被网络协议 1009 中的 step 覆盖

类型: uint

单位: 无

典型值: 0, 5

配置建议: 0 表示取所有激光, 5 表示每 5 束激光取一束激光。增大该值有助于减小网络传输的数据量, 但是会使客户端中的激光显示更不准确

2. timeOut

含义: 失联后返回 ReturnPoint 的超时时间

类型: uint

单位: 秒

典型值: 无

配置建议: 若没有任何客户端与机器人连接(这里的客户端不仅仅指 RoboShop, 任何通过网络协议与机器人连接的都是客户端), 且该值不为 0, 那么机器人会立即静止, 该值时间内如果没有任何客户端重新建立连接, 则机器人会自动返回 ReturnPoint(必须在地图中存在并且可达); 若该值为 0, 没有任何客户端连接时, 机器人不做处理, 继续保持之前的状态

XBox360Joystick

1. safeMode

含义: 是否启用手柄操控安全模式

类型: bool

单位: 无

典型值: true

配置建议: 建议启用, 启用后机器人会被限速(见下文三个参数), 并且松开按钮后就会减速停止, 更安全; 若不启用, 松开按钮后机器人会保持松开前的速度直到停止键被按下

2. backwardSpeed

含义: 手柄操控安全模式时的最大后退速度的绝对值

类型: double

单位: m/s

典型值: 0.5

配置建议: 按住手柄后退按钮, 机器人会加速到该值, 然后匀速运行, 松开按钮则减速后停止; 不建议设置过大

3. forwardSpeed

含义: 手柄操控安全模式时的最大前进速度的绝对值

类型: double

单位: m/s

典型值: 0.5

配置建议: 按住手柄前进按钮, 机器人会加速到该值, 然后匀速运行, 松开按钮则减速后停止; 不建议设置过大

4. turnSpeed

含义: 手柄操控安全模式时的最大角速度的绝对值

类型: double

单位: degree/s

典型值: 60

配置建议: 按住手柄左(右)旋转按钮, 机器人会加速旋转到该值, 然后匀速转动, 松开按钮则减速后停止; 不建议设置过大

地图格式

注 ***1:*** 地图中的坐标点的单位均为米，最多会保留三位小数，即精确到 ***0.001*** 米

注 ***2:*** 地图的坐标系即为世界坐标系

地图的 **protobuf** 格式

Protobuf3 请参阅 Google

```
syntax = "proto3";
package rbk.protocol;
import "google/protobuf/wrappers.proto";

message Message_MapProperty {
    string key = 1;  // 属性名
     string type = 2; // 属性值的类型, 这里 type 一定是 string, bool, int32, uint32, int64, uint64, float, double, bytes
 中的一个
    bytes value = 3; // 属性值 (for backward compatibility)
    oneof oneof_value { // 属性值
        string string_value = 4;
        bool bool_value = 5;
        int32 int32_value = 6;
        uint32 uint32_value = 7;
        int64 int64_value = 8;
        uint64 uint64_value = 9;
        float float_value = 10;
        double double_value = 11;
        bytes bytes_value = 12;
    }
}

// 普通点
message Message_MapPos {
    double x = 1;
    double y = 2;
}

// 普通线
message Message_MapLine {
    Message_MapPos start_pos = 1; // 线的起始点
    Message_MapPos end_pos = 2;   // 线的终止点
}

// 地图元数据
message Message_MapHeader {
    string map_type = 1;    //type of map: 2D-map or 3D-map
    string map_name = 2;    //name of map: the map file name
    Message_MapPos min_pos = 3; // 地图中 x,y 最小值, 该点不一定存在, 只是指示包含增长地图的矩形的左下点
    Message_MapPos max_pos = 4; // 地图中 x,y 最大值, 该点不一定存在, 只是指示包含增长地图的矩形的右上点
    double resolution = 5; // 分辨率 (m)
    string version = 8; // 地图的格式的版本号
}

// 特征, 用户不必关注该条目
message Message_MapAttribute {
    string description = 1; // description of the this class
    uint32 color_pen = 2;
    uint32 color_brush = 3;
    uint32 color_font = 4;
}

// 高级点
message Message_AdvancedPoint {
    string class_name = 1;      // 高级点的类型, 目前只有 LandMark(普通站点), ChargePoint(充电点), ReturnPoint(返航点), Gyr
oCaliPoint(陀螺仪标定点), RobotHome(出生点)
    string instance_name = 2; // 唯一标识名
```

```
    Message_MapPos pos = 3;   // 点坐标
    double dir = 4;           // 方向
    repeated Message_MapProperty property = 5; // 高级点的属性
    bool ignore_dir = 6;  // 是否忽略朝向
    bytes desc = 8;
    Message_MapAttribute attribute = 10; // 特征
}

// 高级线
message Message_AdvancedLine {
    string class_name = 1;    // 高级线的类型, 目前只有 ForbiddenLine, NormalLine, VirtualLine
    string instance_name = 2; // 唯一标识名
    Message_MapLine line = 3; // 线坐标
    repeated Message_MapProperty property = 4; // 高级线的属性
    bytes desc = 8;
    Message_MapAttribute attribute = 10; // 特征
}

// 高级曲线(用于连接两个高级点), 目前只有 BezierPath(三阶贝塞尔曲线) 一种
message Message_AdvancedCurve {
    string class_name = 1;   // 高级曲线的类型, 目前只有 BezierPath(贝塞尔曲线) 一种
    string instance_name = 2;  // 唯一标识名
    Message_AdvancedPoint start_pos = 3; // 曲线起始点(是某个高级点,一定是地图中出现过的某个高级点)
    Message_AdvancedPoint end_pos = 4;   // 曲线终止点(是某个高级点,一定是地图中出现过的某个高级点)
    Message_MapPos control_pos1 = 5;     // 曲线控制点1
    Message_MapPos control_pos2 = 6;     // 曲线控制点2
    repeated Message_MapProperty property = 7; // 高级曲线的属性
    bytes desc = 8;
    repeated Message_Device devices = 12;   // 设备模型相关信息
    Message_MapAttribute attribute = 15; // 特征
}

// 高级区域
message Message_AdvancedArea {
    string class_name = 1;    // 高级曲线的类型,
    string instance_name = 2; // 唯一标识名
    repeated Message_MapPos pos_group = 3; // 区域边界线
    double dir = 4; // 方向
    repeated Message_MapProperty property = 5; // 高级区域的属性
    bytes desc = 8;
    repeated Message_Device devices = 10;  // 设备模型相关信息
    Message_MapAttribute attribute = 15; // 特征
}

// 激光设备, 用户不必关注
message Message_LaserDevice {
    uint32 id = 1;
    repeated Message_MapPos laser_margin_pos = 2;
}

// 设备模型信息, 用户不必关注
message Message_Device {
    string model_name = 1;
    repeated Message_LaserDevice laser_devices = 5;
    repeated double ultrasonic_dist = 6; // dist = -1 means not used
    repeated double fallingdown_dist = 7; // dist = -1 means not used
}

// 巡检站点
message Message_PatrolRouteStation {
    string id = 1; // 站点 ID
}

// 巡检线路
message Message_PatrolRoute {
    string name = 1; // 线路名称
    repeated Message_PatrolRouteStation station_list = 2; // 线路上的站点列表
    google.protobuf.DoubleValue max_speed = 4;   // 巡检线路上的最大速度, 若缺省则使用地图中的设置
    google.protobuf.DoubleValue max_acc = 5;     // 巡检线路上的最大加速度, 若缺省则使用地图中的设置
    google.protobuf.DoubleValue max_rot = 6;     // 巡检线路上的最大角速度, 若缺省则使用地图中的设置
    google.protobuf.DoubleValue max_rot_acc = 7; // 巡检线路上的最大角加速度, 若缺省则使用地图中的设置
    bytes desc = 8;
```

```
}

// 地图
message Message_Map {
    string map_directory = 1;
    Message_MapHeader header = 2; // 元数据
    repeated Message_MapPos normal_pos_list = 3; // 普通点数组
    repeated Message_MapLine normal_line_list = 4; // 普通线数组
    repeated Message_AdvancedPoint advanced_point_list = 6; // 高级点数组
    repeated Message_AdvancedLine  advanced_line_list = 7;  // 高级线数组
    repeated Message_AdvancedCurve advanced_curve_list = 8; // 高级曲线数组
    repeated Message_AdvancedArea  advanced_area_list = 9;  // 高级区域数组
    repeated Message_PatrolRoute patrol_route_list = 10;    // 巡检线路数组
}
```

地图的 JSON 格式

地图的 JSON 格式是从 protobuf 格式转换而来, JSON 和 protobuf 格式可以相互转换, 转换按照 protobuf3 的规则。

首先是对地图中会出现的一些类型的说明：

注: 注释中 *[]* 表示这个 ***key*** 一定存在，*()* 小括号表示可缺省

MapProperty 属性:

属性都是跟软件GUI或者算法相关的，用户不必理会

```
{
    "key": "spin",   // [string]
    "type": "int32",   // [string]
    "value": "MQ==",  // [string] 这里的值需要将类型转换为 string, 再进行 base64 编码, 如 int32 = 1, 转换为 string 是 "1", 进行 base64 编码后为 "MQ=="
    "int32Value": 1  // 该字段会根据 type 变化, 可能为 stringValue, boolValue, int32Value, uint32Value, int64Value, uint64Value, floatValue, doubleValue, bytesValue
}
```

MapPos 普通点:

```
{
    "x": 10.111,  // [double] 世界坐标系中 x 坐标
    "y": 2.222,   // [double] 世界坐标系中 y 坐标
}
```

MapLine 普通线段:

```
{
    "startPos": {      // [MapPos], 是该线段的起始点，类型也就是一个 MapPos
        "x": 1.123,
        "y": 2.123
    },
    "endPos": {        // [MapPos], 是该线段的终止点，类型也就是一个 MapPos
        "x": 3.321,
        "y": 5.123
    }
}
```

MapHeader 地图头部:

```
{
    "mapType": "2D-map",      // [string] 2D-map or 3D-map，目前只会是 2D-map
    "mapName": "seer.map",    // [string] 地图的名字
     "minPos": {              // [MapPos] 地图中可以囊括整个地图的左上点（不一定实际存在这个点），类型也就是一个 MapPos
         "x": -43.812,
         "y": -70.232
     },
     "maxPos": {              // [MapPos] 地图中可以囊括整个地图的右下点（不一定实际存在这个点），类型也就是一个 MapPos
         "x": 166.321,
         "y": 86.600
    },
    "resolution": 0.02,        // [double] 分辨率(m)
    "version": "1.0.0"
}
```

AdvancedPoint 高级点:

```
{
    "className": "LandMark",           // [string] 高级点类型, 目前只有 LandMark(普通站点), ChargePoint(充电点), ReturnPoi
nt(返航点), GyroCaliPoint(陀螺仪标定点), RobotHome(出生点)
    "instanceName": "0",               // [string] 唯一标识名
    "pos": {"x": 18.420, "y": 9.270},  // [MapPos]
    "dir": 0.894,                      // (double) 方向
    "property": [{                     // (MapProperty[]) 属性数组
         "key": "spin",
         "type": "int32",
         "value": "MQ==",
         "int32Value": 1
    }]
}
```

AdvancedLine 高级线:

```
{
    "className": "ForbiddenLine",      // [string] 高级线类型, 目前只有 ForbiddenLine, NormalLine, VirtualLine
    "instanceName": "2",               // [string] 唯一标识名
    "line": {                          // [MapLine]
        "startPos": {"x": 50.970,"y": 28.460},
        "endPos": {"x": 74.690, "y": 28.460}
    },
    "property": [{                     // (MapProperty[]) 属性数组
        "key": "spin",
        "type": "int32",
        "value": "MQ==",
        "int32Value": 1
    }]
}
```

AdvancedCurve 高级曲线:

```
{
    "className": "BezierPath",          // [string] 高级曲线类型
    "instanceName": "3",                // (string) 唯一标识名
    "startPos": {                       // [AdvancedPoint] 贝塞尔曲线起始点,一定是地图中出现过的某个高级点
        "className": "LandMark",
        "instanceName": "10",
        "pos": {"x": 22.415, "y": 38.104}
    },
    "endPos": {                         // [AdvancedPoint] 贝塞尔曲线终止点曲线起始点,一定是地图中出现过的某个高级点
        "className": "LandMark",
        "instanceName": "2",
        "pos": {"x": 22.297, "y": 14.284}
    },
    "controlPos1": {"x": 22.009, "y": 28.664},  // [MapPos] 贝塞尔曲线控制点1
    "controlPos2": {"x": 22.369, "y": 20.501},  // [MapPos] 贝塞尔曲线控制点2
    "property": [],                             // (MapProperty[]) 属性数组
    "devices": []                               // (Message_Device[]) 设备模型相关
}
```

AdvancedArea 高级区域:

```
{
    "className":"AdvancedArea",   // [string] 高级曲线类型, 目前都为 AdvanceArea
    "instanceName":"1", // [string] 唯一标识名
    "posGroup":[        // [MapPos[]] 区域顶点，首尾相连
        {"x":1.889,"y":2.250},
        {"x":2.380,"y":2.250},
        {"x":2.380,"y":1.751},
        {"x":1.889,"y":1.751}],
    "property":[], // (MapProperty[]) 属性数组
    "devices": []  // (Message_Device[]) 设备模型相关
}
```

巡检站点:

```
{
    "id": "LM1"
}
```

巡检线路:

```
{
    "name": "route1",
    "stationList": [
        {
            "id": "LM1"
        },
        {
            "id": "LM2"
        }
    ]
}
```

Map 地图:

实际的地图对象，以上只是地图中会用到的对象的说明

```
{
    "mapDirectory": "",  // (string) 地图路径, 目前不用
    "header": {          // [MapHeader] 地图头部
        "mapType": "2D-map",
        "mapName": "seer.smap",
```

```
        "minPos": {"x": -43.800, "y": -70.200},
        "maxPos": {"x": 166.000, "y": 86.600},
        "resolution": 0.02,
        "version": "1.0.0"
    },
    "normalPosList": [   // (MapPos[]) 普通点数组
       {"x": -43.800, "y": -20.800},
       {"x": -43.600, "y": -20.600},
       {"x": -43.400, "y": -20.600}
    ],
    "normalLineList": [  // (MapLine[]) 普通线数组
       {
           "startPos": {"x": 59.280, "y": -24.420},
           "endPos": {"x": 63.270, "y": -22.280}
       },
       {
           "startPos": {"x": 116.540, "y": -52.120},
           "endPos": {"x": 145.040, "y": -51.880}
       }
    ],
    "advancedPointList":[             // (AdvancedPoint[]) 高级点数组
        {"className": "LandMark", "instanceName": "0", "pos": {"x": 18.420, "y": 9.270}},
        {"className": "RobotHome", "instanceName": "1", "pos":{"x": 21.930, "y": 29.520}},
        {"className": "LandMark", "instanceName": "2", "pos":{"x":22.415,"y":38.103},
         "property": [{"key": "spin", "type": "int32", "value": "MQ==", "int32Value": 1},
                      {"key": "tshaped", "type": "int32", "value": "MQ==", "int32Value": 1}]},
        {"className": "LandMark", "instanceName": "3", "pos":{"x":22.297,"y":14.284},
         "property": [{"key": "spin", "type": "int32", "value": "MQ==", "int32Value": 1},
                      {"key": "tshaped", "type": "int32", "value": "MQ==", "int32Value": 1}]},
       {"className": "LandMark", "instanceName": "4", "pos":{"x":36.825,"y":-8.824},
        "property": [{"key": "spin", "type": "int32", "value": "MQ==", "int32Value": 1},
                     {"key": "tshaped", "type": "int32", "value": "MQ==", "int32Value": 1}]}
   ],
    "advancedLineList": [             // (AdvancedLine[]) 高级线数组
        {"className": "ForbiddenLine", "instanceName":"",
         "line": {"startPos": {"x": 50.970,"y": 28.460}, "endPos": {"x": 74.690, "y": 28.460}}},
       {"className": "ForbiddenLine", "instanceName":"",
        "line": {"startPos": {"x": -18.460,"y": 14.010}, "endPos": {"x": 3.280, "y": 14.010}}},
       {"className": "VirtualLine", "instanceName":"",
        "line": {"startPos": {"x": 45.600,"y": -2.580}, "endPos": {"x": 111.700, "y": -2.580}}},
       {"className": "VirtualLine", "instanceName":"",
        "line": {"startPos":{"x":4597,"y":-1440}, "endPos": {"x": 10725, "y": -1440}}}
   ],
    "advancedCurveList": [            // (AdvancedCurve[]) 高级曲线数组
       {"className": "BezierPath", // 连接上面高级点数组中 instanceName 为 0 和 1 的曲线
        "startPos": {"className": "LandMark", "instanceName": "0", "pos": {"x": 18.420, "y": 9.270}},
        "endPos": {"className": "LandMark", "instanceName": "1", "pos": {"x": 21.930, "y": 29.520}},
        "controlPos1": {"x": 22.009, "y": 28.664},
        "controlPos2": {"x": 22.369, "y": 20.501},
        "property": [],
        "devices": []},
       {"className": "BezierPath", // 连接上面高级点数组中 instanceName 为 1 和 2 的曲线
        "startPos": {"className": "LandMark", "instanceName": "1", "pos": {"x": 21.930, "y": 29.520}},
       "endPos": {"className": "LandMark", "instanceName": "2", "pos": {"x": 22.415, "y": 38.103}},
        "controlPos1": {"x": -32.702, "y": 6.313},
        "controlPos2": {"x": -32.702, "y": 6.313},
        "property": [],
        "devices": []}
    ],
    "advancedAreaList": [             // (AdvancedArea[]) 高级区域数组
        {"className":"AdvancedArea",
        "instanceName":"1",
        "posGroup":[
            {"x":1.889,"y":2.250},
            {"x":2.380,"y":2.250},
            {"x":2.380,"y":1.751},
            {"x":1.889,"y":1.751}],
        "property":[
            {"key":"ultrasonic","type":"bool","value":"ZmFsc2U=", "boolValue":false},  // 超声是否打开，不存在此字段时默认
打开
            {"key":"fallingdown","type":"bool","value":"ZmFsc2U=", "boolValue":false}] // 防跌落是否打开，不存在此字段时默
认打开
```

```
        }
    ],
    "patrolRouteList": [
        {"name": "route1",
         "stationList": [
            {
                "id": "LM1"
            },
            {
                "id": "LM2"
            }]
        }
    ]
}
```

注：以上地图只是示例，并不是一张可以使用的地图。

注：高级点 ***(AdvancedPoint)*** 的点即为导航点，用户只需关心各种高级点以及连接它们的 ***BezierPath (AdvancedCurve)*** 即可。