悟时机器人应用层ROS接口

版本	编辑时间	编辑人	编辑内容
v0.0.1-beta	2025-1-2	HuiMin	初稿

简介

略

开发与运行环境

1. Ubuntu 24.04 AMD64 ROS 2 **Jazzy**
2. Ubuntu 22.04 AMD64 ROS 2 **Humble**
3. Ubuntu 20.04 AMD64 ROS 2 **Foxy**
4. Ubuntu 20.04 **ARM64** ROS 2 **Foxy**

部署运行

获取对应版本的`ros-xxx-woosh-robot-agent_xxx_xxx.run`安装包.

```
chmod +x ros-xxx-woosh-robot-agent_xxx_xxx.run
./ros-xxx-woosh-robot-agent_xxx_xxx.run
```

运行`agent`:

```
ros2 run woosh_robot_agent agent --ros-args -r __ns:=/woosh_robot -p
ip:="172.20.8.74"
```

命名空间`ns`建议设定为`/woosh_robot`, 文档及提供的`Demo`均是以此为准.

`ip`为机器人底盘的`IP`, 默认为`169.254.128.2`.

接口说明

目前提供三种形式的接口，分别是`service`、`topic`和`action`

机器人信息相关

获取机器人所有信息

- 接口类型：`service`
- 服务名称：`robot/RobotInfo`
- 消息类型：`woosh_robot_msgs/srv/RobotInfo`

ros cli command:

```
ros2 service call /woosh_robot/robot/RobotInfo woosh_robot_msgs/srv/RobotInfo
```

参数说明详见`ros2 interface show woosh_robot_msgs/msg/RobotInfo --all-comments`

获取常规信息

- 接口类型：`service`
- 服务名称：`robot/General`
- 消息类型：`woosh_robot_msgs/srv/General`

ros cli command：

```
ros2 service call /woosh_robot/robot/General woosh_robot_msgs/srv/General
```

参数说明详见 `ros2 interface show woosh_robot_msgs/msg/General --all-comments`

```
# 机器人常规信息

# 机器人类型
woosh_robot_msgs/Type type
    # 未定义的
    int32 K_TYPE_UNDEFINED=0
    # 通用底盘
    int32 K_BASE_ROBOT_200=1
    # 栈板平台举升
    int32 K_PALLET_LIFT_ROBOT_500=11
    # 移动料车举升
    int32 K_SHELF_LIFT_ROBOT_500=21
    # 牵引机器人
    int32 K_TRACTOR_ROBOT_500=31
    # 辊筒机器人
    int32 K_ROLLER_ROBOT_500=41
    # 复合机器人统称
    int32 K_COMPLEX_ROBOT=50
    # 复合机械臂
    int32 K_ARM_ROBOT_14=61

    int32 value
# 机器人尺寸+自重+载重
woosh_robot_msgs/GeneralModelData model_data
    # 长
    uint32 length
    # 宽
    uint32 width
    # 高
    uint32 height
    # 自重
    uint32 weight
    # 载重
    uint32 load
# 模型名称
string urdf_name
# 显示名称
string display_model
# 机器人编号
uint32 serial_number
# 机器人服务号
string service_id
```

```
# 驱动方式, 0: 两轮差速, 1: 四舵轮
uint32 driver_method
woosh_robot_msgs/GeneralVersion version
    # 机器人系统版本号
    string system
    # 应用模块版本号
    string rc
```

获取配置信息

- 接口类型：`service` | `topic`
- 服务名称：`robot/Setting`
- 话题名称：`robot/Setting`
- 消息类型：`woosh_robot_msgs/msg/Setting`

ros cli command:

```
ros2 service call /woosh_robot/robot/Setting woosh_robot_msgs/srv/Setting
ros2 topic echo /woosh_robot/robot/Setting
```

参数说明详见 `ros2 interface show woosh_robot_msgs/msg/Setting --all-comments`

```
# 机器人基本设置信息

# 机器人标识
woosh_robot_msgs/Identity identity
    # 机器人昵称
    string name
# 连接服务器信息
woosh_robot_msgs/Server server
    # 服务器IP
    string ip
    # 服务器端口
    uint32 port
# 电量配置信息
woosh_robot_msgs/Power power
    # 警告电量值
    uint32 alarm
    # 低电量值
    uint32 low
    # 空闲电量值
    uint32 idle
    # 满电量值
    uint32 full
# 声音设置信息
woosh_robot_msgs/Sound sound
    # 静音
    bool mute
    # 音量
    uint32 volume
woosh_robot_msgs/SettingAllow allow
    # 是否开启低电量时自主回充
    bool auto_charge
```

```
    # 是否开启空闲时自主泊车
    bool auto_park
    # 是否开启货物检测
    bool goods_check
    # 是否开启机械检测
    bool mechanism_check
```

获取机器人状态

- 接口类型：`service` | `topic`
- 服务名称：`robot/RobotState`
- 话题名称：`robot/RobotState`
- 消息类型：`woosh_robot_msgs/msg/RobotState`

ros cli command：

```
ros2 service call /woosh_robot/robot/RobotState woosh_robot_msgs/srv/RobotState
ros2 topic echo /woosh_robot/robot/RobotState
```

参数说明详见 `ros2 interface show woosh_robot_msgs/msg/RobotState --all-comments`

```
# 机器人状态
woosh_robot_msgs/State state
    # 未定义的
    int32 K_STATE_UNDEFINED=0
    # 未初始化
    int32 K_UNINIT=1
    # 空闲
    int32 K_IDLE=2
    # 泊车中
    int32 K_PARKING=3
    # 任务中
    int32 K_TASK=4
    # 警告
    int32 K_WARNING=5
    # 异常
    int32 K_FAULT=6
    # 跟随中
    int32 K_FOLLOWING=7
    # 充电中
    int32 K_CHARGING=8
    # 构图中
    int32 K_MAPPING=9

    int32 value
```

获取模式信息

- 接口类型：`service` | `topic`
- 服务名称：`robot/Mode`
- 话题名称：`robot/Mode`

- 消息类型：`woosh_robot_msgs/msg/Mode`

ros cli command：

```
ros2 service call /woosh_robot/robot/Mode woosh_robot_msgs/srv/Mode
ros2 topic echo /woosh_robot/robot/Mode
```

参数说明详见 `ros2 interface show woosh_robot_msgs/msg/Mode --all-comments`

```
# 机器人控制模式信息

# 控制模式
woosh_robot_msgs/ControlMode ctrl
    # 未定义的
    int32 K_CONTROL_MODE_UNDEFINED=0
    # 自动
    int32 K_AUTO=1
    # 手动
    int32 K_MANUAL=2
    # 维护
    int32 K_MAINTAIN=3
    int32 value
# 工作模式，控制模式为自动时有效
woosh_robot_msgs/WorkMode work
    # 未定义的
    int32 K_WORK_MODE_UNDEFINED=0
    # 部署模式
    int32 K_DEPLOY_MODE=1
    # 任务模式
    int32 K_TASK_MODE=2
    # 调度模式
    int32 K_SCHEDULE_MODE=3

    int32 value
```

获取位姿速度

- 接口类型：`service` | `topic`
- 服务名称：`robot/PoseSpeed`
- 话题名称：`robot/PoseSpeed`
- 消息类型：`woosh_robot_msgs/msg/PoseSpeed`

ros cli command：

```
ros2 service call /woosh_robot/robot/PoseSpeed woosh_robot_msgs/srv/PoseSpeed
ros2 topic echo /woosh_robot/robot/PoseSpeed
```

参数说明详见 `ros2 interface show woosh_robot_msgs/msg/PoseSpeed --all-comments`

```
# 机器人位姿速度

# 速度
woosh_common_msgs/Twist twist
```

```
    # 线速度
    float32 linear
    # 角速度
    float32 angular
# 位姿
woosh_common_msgs/Pose2D pose
    # x
    float32 x
    # y
    float32 y
    # 朝向
    float32 theta
# 地图ID
uint32 map_id
# 累计里程，单位m
uint32 mileage
```

获取电池信息

- 接口类型：`service` | `topic`
- 服务名称：`robot/Battery`
- 话题名称：`robot/Battery`
- 消息类型：`woosh_robot_msgs/msg/Battery`

ros cli command：

```
ros2 service call /woosh_robot/robot/Battery woosh_robot_msgs/srv/Battery
ros2 topic echo /woosh_robot/robot/Battery
```

参数说明详见 `ros2 interface show woosh_robot_msgs/msg/Battery --all-comments`

```
# 机器人电池信息

# 充电状态
woosh_robot_msgs/BatteryChargeState charge_state
    # 未定义的
    int32 K_CHARGE_STATE_UNDEFINED=0
    # 0: 没在充电
    int32 K_NOT=1
    # 1: 手动充电中
    int32 K_MANUAL=2
    # 2: 自动充电中
    int32 K_AUTO=3

    int32 value
# 电量百分比，0-100数值，100表示全满，0表示没有电量
uint32 power
# 电池健康(充满容量/设计容量)
uint32 health
# 循环次数
uint32 charge_cycle
# 电池寿命
uint32 battery_cycle
```

```
# 电池温度(最高温度)
uint32 temp_max
```

获取网络信息

- 接口类型：`service` | `topic`
- 服务名称：`robot/Network`
- 话题名称：`robot/Network`
- 消息类型：`woosh_robot_msgs/msg/Network`

ros cli command:

```
ros2 service call /woosh_robot/robot/Network woosh_robot_msgs/srv/Network
ros2 topic echo /woosh_robot/robot/Network
```

参数说明详见 `ros2 interface show woosh_robot_msgs/msg/Network --all-comments`

```
# 机器人网络信息

# 网络连接状态
bool is_connected
# 机器人IP
string robot_ip
# 调度IP
string sch_ip
# 机器人WiFi信息
woosh_robot_msgs/NetworkWiFi wifi
    # 当前连接WiFi名称
    string name
    # 网络连接状态码
    uint64 code
    # WiFi列表json格式
    uint8[] list_json
    # WiFi信号强度
    uint32 strength
    # WiFi模式
    woosh_robot_msgs/NetworkWiFiMode mode
        # 未定义的
        int32 K_WI_FI_MODE_UNDEFINED=0
        # ap模式
        int32 K_AP=1
        # 正在切换到ap模式
        int32 K_TO_AP=2
        # 客户端模式
        int32 K_CLIENT=3
        # 正在切换到客户端模式
        int32 K_TO_CLIENT=4

        int32 value
```

获取场景信息

- 接口类型：`service` | `topic`
- 服务名称：`robot/Scene`
- 话题名称：`robot/Scene`
- 消息类型：`woosh_robot_msgs/msg/Scene`

ros cli command：

```
ros2 service call /woosh_robot/robot/Scene woosh_robot_msgs/srv/Scene
ros2 topic echo /woosh_robot/robot/Scene
```

参数说明详见 `ros2 interface show woosh_robot_msgs/msg/Scene --all-comments`

```
# 机器人场景信息

# 场景名
string scene_name
# 当前地图ID
uint32 map_id
# 地图名
string map_name
# 地图数据版本号
int64 version
```

获取任务进度

- 接口类型：`service` | `topic`
- 服务名称：`robot/TaskProc`
- 话题名称：`robot/TaskProc`
- 消息类型：`woosh_robot_msgs/msg/TaskProc`

ros cli command：

```
ros2 service call /woosh_robot/robot/TaskProc woosh_robot_msgs/srv/TaskProc
ros2 topic echo /woosh_robot/robot/TaskProc
```

参数说明详见 `ros2 interface show woosh_robot_msgs/msg/TaskProc --all-comments`

```
# 机器人任务执行信息

# 机器人任务ID
int64 robot_task_id
# 任务类型
woosh_task_msgs/Type type
    # 未定义的
    int32 K_TYPE_UNDEFINED=0
    # 拣选
    int32 K_PICK=1
    # 泊车
    int32 K_PARKING=2
```

```
    # 充电
    int32 K_CHARGE=3
    # 搬运
    int32 K_CARRY=4

    int32 value
# 任务状态
woosh_task_msgs/State state
    # 未定义的
    int32 K_STATE_UNDEFINED=0
    # 初始化
    int32 K_INIT=1
    # 准备的
    int32 K_READY=2
    # 执行中
    int32 K_EXECUTING=3
    # 暂停的
    int32 K_PAUSED=4
    # 动作等待
    int32 K_ACTION_WAIT=5
    # 任务等待
    int32 K_TASK_WAIT=6
    # 完成的
    int32 K_COMPLETED=7
    # 取消的
    int32 K_CANCELED=8
    # 失败的
    int32 K_FAILED=9

    int32 value
# 动作信息
woosh_robot_msgs/TaskProcAction action
    # 动作类型
    woosh_action_msgs/Type type
        # 未定义的
        int32 K_TYPE_UNDEFINED=0
        # 导航
        int32 K_NAV=1
        # 单步控制
        int32 K_STEP_CTRL=2
        # 二次定位进入
        int32 K_SECONDPOS_ENTER=3
        # 二次定位退出
        int32 K_SECONDPOS_QUIT=4
        # 搬运动作
        int32 K_CARRY=5
        # 等待
        int32 K_WAIT=6
        # 充电
        int32 K_CHARGE=7

        int32 value
    # 动作状态
    woosh_action_msgs/State state
        # 未定义的
        int32 K_STATE_UNDEFINED=0
```

```
        # 执行中
        int32 K_ROS_EXECUTING=1
        # 警告
        int32 K_ROS_WARNING=2
        # 取消
        int32 K_ROS_CANCEL=3
        # 完成
        int32 K_ROS_SUCCESS=4
        # 失败
        int32 K_ROS_FAILURE=5
        # 暂停
        int32 K_SUSPEND=10
        # 管制
        int32 K_TRAFFI_CTRL=11

        int32 value
    # 动作等待ID
    int32 wait_id
# 目的地
string dest
# 消息
string msg
# 最后更新时间(s)
int32 time
```

获取最近50条历史任务

- 接口类型：`service`
- 服务名称：`robot/TaskHistory`
- 消息类型：`woosh_robot_msgs/msg/TaskHistory`

ros cli command：

```
ros2 service call /woosh_robot/robot/TaskHistory woosh_robot_msgs/srv/TaskHistory
```

参数说明详见 `ros2 interface show woosh_robot_msgs/msg/TaskHistory --all-comments`

获取设备状态

- 接口类型：`service` | `topic`
- 服务名称：`robot/DeviceState`
- 话题名称：`robot/DeviceState`
- 消息类型：`woosh_robot_msgs/msg/DeviceState`

ros cli command：

```
ros2 service call /woosh_robot/robot/DeviceState woosh_robot_msgs/srv/DeviceState
ros2 topic echo /woosh_robot/robot/DeviceState
```

参数说明详见：

```
ros2 interface show woosh_robot_msgs/msg/DeviceState --all-comments
ros2 interface show woosh_robot_msgs/msg/DeviceStateHardwareBit --all-comments
ros2 interface show woosh_robot_msgs/msg/DeviceStateSoftwareBit --all-comments
```

```
# 机器人设备状态

# DeviceState.HardwareBit, 每个bit表示一个状态
uint32 hardware
# DeviceState.SoftwareBit, 每个bit表示一个状态
uint32 software
```

```
# 机器人硬件设备位信息

# 未定义
int32 K_HARDWARE_BIT_UNDEFINED=0
# 按钮1(暂停/继续/下一步)
int32 K_BTN1=1
# 按钮2(复位)
int32 K_BTN2=2
# 按钮3
int32 K_BTN3=4
# 按钮4
int32 K_BTN4=8
# 按钮5
int32 K_BTN5=16
# 按钮6
int32 K_BTN6=32
# 按钮7
int32 K_BTN7=64
# 按钮8
int32 K_BTN8=128
# 伺服释放按钮
int32 K_SERVO_BTN=256
# 举升按钮
int32 K_LIFT_BTN=512
# 急停触发
int32 K_EMG_BTN=1024

int32 value
```

```
# 机器人软件设备位信息

# 未定义
int32 K_SOFTWARE_BIT_UNDEFINED=0
# 定位状态
int32 K_LOCATION=1
# 调度连接
int32 K_SCHEDULE=2
# 货物状态
int32 K_GOODS_STATE=4
# 占用状态
int32 K_OCCUPANCY=8
# 屏蔽呼叫
```

```
int32 K_MUTE_CALL=16
# 程序静音
int32 K_PROGRAM_MUTE=32

int32 value
```

获取硬件状态

- 接口类型：`service` | `topic`
- 服务名称：`robot/HardwareState`
- 话题名称：`robot/HardwareState`
- 消息类型：`woosh_robot_msgs/msg/HardwareState`

ros cli command：

```
ros2 service call /woosh_robot/robot/HardwareState
woosh_robot_msgs/srv/HardwareState
ros2 topic echo /woosh_robot/robot/HardwareState
```

参数说明详见：

```
ros2 interface show woosh_robot_msgs/msg/HardwareState --all-comments
ros2 interface show woosh_robot_msgs/msg/HardwareStateState --all-comments
```

获取运行状态

- 接口类型：`service` | `topic`
- 服务名称：`robot/OperationState`
- 话题名称：`robot/OperationState`
- 消息类型：`woosh_robot_msgs/msg/OperationState`

ros cli command：

```
ros2 service call /woosh_robot/robot/OperationState
woosh_robot_msgs/srv/OperationState
ros2 topic echo /woosh_robot/robot/OperationState
```

参数说明详见：

```
ros2 interface show woosh_robot_msgs/msg/OperationState --all-comments
ros2 interface show woosh_robot_msgs/msg/OperationStateNavBit --all-comments
ros2 interface show woosh_robot_msgs/msg/OperationStateRobotBit --all-comments
```

```
# 机器人运行状态

# OperationState.NavBit 每个bit表示一个状态
uint32 nav
# OperationState.RobotBit 每个bit表示一个状态
uint32 robot
```

```
# 机器人导航相关位信息

# 未定义
int32 K_NAV_BIT_UNDEFINED=0
# 狭窄通道
int32 K_NARROW=1
# 引导到达
int32 K_GUIDE=2
# 乘梯中
int32 K_INA_LIFT=4
# 阻碍
int32 K_IMPEDE=8
# 二维码
int32 K_QR_CODE=16
# 分段到达
int32 K_STAGE=32

int32 value
```

```
# 机器人位信息

# 未定义
int32 K_ROBOT_BIT_UNDEFINED=0
# 可接任务
int32 K_TASKABLE=1

int32 value
```

获取模型信息

- 接口类型：`service` | `topic`
- 服务名称：`robot/Model`
- 话题名称：`robot/Model`
- 消息类型：`woosh_robot_msgs/msg/Model`

ros cli command：

```
ros2 service call /woosh_robot/robot/Model woosh_robot_msgs/srv/Model
ros2 topic echo /woosh_robot/robot/Model
```

参数说明详见 `ros2 interface show woosh_robot_msgs/msg/Model --all-comments`

```
# 机器人模型

woosh_common_msgs/Point[] model
    # x
    float32 x
    # y
    float32 y
    # z
    float32 z
# 模型类型
woosh_robot_msgs/FootPrint type
```

```
    # 原始
    int32 K_ORIGINAL=0
    # 膨胀(背负货物)
    int32 K_EXPAND=1
    # 备用
    int32 K_SPARE=2
    # 对接
    int32 K_DOCK=3

    int32 value
```

获取异常码

- 接口类型：`service` | `topic`
- 服务名称：`robot/AbnormalCodes`
- 话题名称：`robot/AbnormalCodes`
- 消息类型：`woosh_robot_msgs/msg/AbnormalCodes`

ros cli command：

```
ros2 service call /woosh_robot/robot/AbnormalCodes
woosh_robot_msgs/srv/AbnormalCodes
ros2 topic echo /woosh_robot/robot/AbnormalCodes
```

参数说明详见 `ros2 interface show woosh_robot_msgs/msg/AbnormalCodes --all-comments`

订阅状态码

- 接口类型：`topic`
- 话题名称：`robot/StatusCode`
- 消息类型：`woosh_robot_msgs/msg/StatusCode`

ros cli command：

```
ros2 topic echo /woosh_robot/robot/StatusCode
```

参数说明详见 `ros2 interface show woosh_robot_msgs/msg/StatusCode --all-comments`

获取最近50条状态码

- 接口类型：`service`
- 服务名称：`robot/StatusCodes`
- 消息类型：`woosh_robot_msgs/msg/StatusCodes`

ros cli command：

```
ros2 service call /woosh_robot/robot/StatusCodes woosh_robot_msgs/srv/StatusCodes
```

参数说明详见 `ros2 interface show woosh_robot_msgs/msg/StatusCodes --all-comments`

获取导航路径

- 接口类型：`service` | `topic`
- 服务名称：`robot/NavPath`
- 话题名称：`robot/NavPath`
- 消息类型：`woosh_robot_msgs/msg/NavPath`

ros cli command：

```
ros2 service call /woosh_robot/robot/NavPath woosh_robot_msgs/srv/NavPath
ros2 topic echo /woosh_robot/robot/NavPath
```

参数说明详见 `ros2 interface show woosh_robot_msgs/msg/NavPath --all-comments`

```
# 机器人导航路径

# 导航路径
woosh_nav_msgs/Path path
    # 路径
    woosh_common_msgs/Pose2D[] poses
        # x
        float32 x
        # y
        float32 y
        # 朝向
        float32 theta
```

机器人请求相关

初始化机器人

- 接口类型：`service`
- 服务名称：`robot/InitRobot`
- 消息类型：`woosh_robot_msgs/srv/InitRobot`

ros cli command：

```
# 原地复位
ros2 service call /woosh_robot/robot/InitRobot woosh_robot_msgs/srv/InitRobot "
{arg:{is_record: true}}"
# 指定坐标复位
ros2 service call /woosh_robot/robot/InitRobot woosh_robot_msgs/srv/InitRobot "
{arg:{pose:{x: 1.23, y: 2.34, theta: 1.57}}}"
```

参数说明详见 `ros2 interface show woosh_robot_msgs/srv/InitRobot --all-comments`

```
# 初始化机器人

woosh_robot_msgs/InitRobot arg
    # 是否记录点复位
    bool is_record
```

```
    # 设置机器人为新的坐标
    woosh_common_msgs/Pose2D pose
        # x
        float32 x
        # y
        float32 y
        # 朝向
        float32 theta
---
# 请求成功与否
bool ok
# 请求状态消息
string msg
```

机器人位置校准

- 接口类型：`service`
- 服务名称：`robot/SetRobotPose`
- 消息类型：`woosh_robot_msgs/srv/SetRobotPose`

ros cli command：

```
ros2 service call /woosh_robot/robot/SetRobotPose
woosh_robot_msgs/srv/SetRobotPose "{arg:{pose:{x: 1.23, y: 2.34, theta: 1.57}}}"
```

参数说明详见 `ros2 interface show woosh_robot_msgs/srv/SetRobotPose --all-comments`

```
# 设置机器人位姿

woosh_robot_msgs/SetRobotPose arg
    woosh_common_msgs/Pose2D pose
        float32 x
        float32 y
        float32 theta
---
# 请求成功与否
bool ok
# 请求状态消息
string msg
```

设置机器人占用

- 接口类型：`service`
- 服务名称：`robot/SetOccupancy`
- 消息类型：`woosh_robot_msgs/srv/SetOccupancy`

ros cli command：

```
ros2 service call /woosh_robot/robot/SetOccupancy
woosh_robot_msgs/srv/SetOccupancy "{arg:{occupy: true}}"
```

参数说明详见 `ros2 interface show woosh_robot_msgs/srv/SetOccupancy --all-comments`

```
# 设置机器人占用

woosh_robot_msgs/SetOccupancy arg
    bool occupy
---
# 请求成功与否
bool ok
# 请求状态消息
string msg
```

设置屏蔽呼叫

- 接口类型：`service`
- 服务名称：`robot/SetMuteCall`
- 消息类型：`woosh_robot_msgs/srv/SetMuteCall`

ros cli command：

```
ros2 service call /woosh_robot/robot/SetMuteCall woosh_robot_msgs/srv/SetMuteCall
"{arg:{mute: true}}"
```

参数说明详见 `ros2 interface show woosh_robot_msgs/srv/SetMuteCall --all-comments`

```
# 设置屏蔽呼叫

woosh_robot_msgs/SetMuteCall arg
    bool mute
---
# 请求成功与否
bool ok
# 请求状态消息
string msg
```

设置程序静音

- 接口类型：`service`
- 服务名称：`robot/SetProgramMute`
- 消息类型：`woosh_robot_msgs/srv/SetProgramMute`

ros cli command：

```
ros2 service call /woosh_robot/robot/SetProgramMute
woosh_robot_msgs/srv/SetProgramMute "{arg:{mute: true}}"
```

参数说明详见 `ros2 interface show woosh_robot_msgs/srv/SetProgramMute --all-comments`

```
# 设置程序静音

woosh_robot_msgs/SetProgramMute arg
    bool mute
---
# 请求成功与否
bool ok
# 请求状态消息
string msg
```

切换控制模式

- 接口类型：`service`
- 服务名称：`robot/SwitchControlMode`
- 消息类型：`woosh_robot_msgs/srv/SwitchControlMode`

ros cli command：

```
ros2 service call /woosh_robot/robot/SwitchControlMode
woosh_robot_msgs/srv/SwitchControlMode "{arg:{mode:{value: 1}}}"
```

参数说明详见 `ros2 interface show woosh_robot_msgs/srv/SwitchControlMode --all-comments`

```
# 切换控制模式

woosh_robot_msgs/SwitchControlMode arg
    # 机器人控制模式
    woosh_robot_msgs/ControlMode mode
        # 未定义的
        int32 K_CONTROL_MODE_UNDEFINED=0
        # 自动
        int32 K_AUTO=1
        # 手动
        int32 K_MANUAL=2
        # 维护
        int32 K_MAINTAIN=3

        int32 value
---
# 机器人模式
woosh_robot_msgs/Mode ret
    # 控制模式
    woosh_robot_msgs/ControlMode ctrl
        # 未定义的
        int32 K_CONTROL_MODE_UNDEFINED=0
        # 自动
        int32 K_AUTO=1
        # 手动
        int32 K_MANUAL=2
        # 维护
        int32 K_MAINTAIN=3
```

```
        int32 value
    # 工作模式，控制模式为自动时有效
    woosh_robot_msgs/WorkMode work
        # 未定义的
        int32 K_WORK_MODE_UNDEFINED=0
        # 部署模式
        int32 K_DEPLOY_MODE=1
        # 任务模式
        int32 K_TASK_MODE=2
        # 调度模式
        int32 K_SCHEDULE_MODE=3

        int32 value
# 请求成功与否
bool ok
# 请求状态消息
string msg
```

切换工作模式

- 接口类型：`service`
- 服务名称：`robot/SwitchWorkMode`
- 消息类型：`woosh_robot_msgs/srv/SwitchWorkMode`

ros cli command：

```
ros2 service call /woosh_robot/robot/SwitchWorkMode
woosh_robot_msgs/srv/SwitchWorkMode "{arg:{mode:{value: 2}}}"
```

参数说明详见 `ros2 interface show woosh_robot_msgs/srv/SwitchWorkMode --all-comments`

```
# 切换工作模式

woosh_robot_msgs/SwitchWorkMode arg
    # 工作模式
    woosh_robot_msgs/WorkMode mode
        # 未定义的
        int32 K_WORK_MODE_UNDEFINED=0
        # 部署模式
        int32 K_DEPLOY_MODE=1
        # 任务模式
        int32 K_TASK_MODE=2
        # 调度模式
        int32 K_SCHEDULE_MODE=3

        int32 value
---
# 机器人模式
woosh_robot_msgs/Mode ret
    # 控制模式
    woosh_robot_msgs/ControlMode ctrl
        # 未定义的
        int32 K_CONTROL_MODE_UNDEFINED=0
        # 自动
```

```
        int32 K_AUTO=1
        # 手动
        int32 K_MANUAL=2
        # 维护
        int32 K_MAINTAIN=3

        int32 value
    # 工作模式，控制模式为自动时有效
    woosh_robot_msgs/WorkMode work
        # 未定义的
        int32 K_WORK_MODE_UNDEFINED=0
        # 部署模式
        int32 K_DEPLOY_MODE=1
        # 任务模式
        int32 K_TASK_MODE=2
        # 调度模式
        int32 K_SCHEDULE_MODE=3

        int32 value
# 请求成功与否
bool ok
# 请求状态消息
string msg
```

切换模型类型

- 接口类型：`service`
- 服务名称：`robot/SwitchFootPrint`
- 消息类型：`woosh_robot_msgs/srv/SwitchFootPrint`

ros cli command：

```
ros2 service call /woosh_robot/robot/SwitchFootPrint
woosh_robot_msgs/srv/SwitchFootPrint "{arg:{type:{value: 1}}}"
```

参数说明详见 `ros2 interface show woosh_robot_msgs/srv/SwitchFootPrint --all-comments`

```
# 切换模型类型

woosh_robot_msgs/SwitchFootPrint arg
    # 模型类型
    woosh_robot_msgs/FootPrint type
        # 原始
        int32 K_ORIGINAL=0
        # 膨胀(背负货物)
        int32 K_EXPAND=1
        # 备用
        int32 K_SPARE=2
        # 对接
        int32 K_DOCK=3

        int32 value
---
# 模型类型
```

```
woosh_robot_msgs/SwitchFootPrint ret
    # 模型类型
    woosh_robot_msgs/FootPrint type
        # 原始
        int32 K_ORIGINAL=0
        # 膨胀(背负货物)
        int32 K_EXPAND=1
        # 备用
        int32 K_SPARE=2
        # 对接
        int32 K_DOCK=3

        int32 value
# 请求成功与否
bool ok
# 请求状态消息
string msg
```

切换地图

- 接口类型：`service`
- 服务名称：`robot/SwitchMap`
- 消息类型：`woosh_robot_msgs/srv/SwitchMap`

ros cli command:

```
ros2 service call /woosh_robot/robot/SwitchMap woosh_robot_msgs/srv/SwitchMap "
{arg:{scene_name: "scenex"}}"
```

参数说明详见 `ros2 interface show woosh_robot_msgs/srv/SwitchMap --all-comments`

```
# 切换地图

woosh_robot_msgs/SwitchMap arg
    # 场景名
    string scene_name
    # 地图名
    string map_name
    # 为空仅切换，否则一并更新
    woosh_common_msgs/FileData[] file_datas
        # 文件名
        string name
        # 文件数据
        uint8[] data
---
# 请求成功与否
bool ok
# 请求状态消息
string msg
```

雷达点云数据

- 接口类型：`service`
- 服务名称：`robot/ScannerData`
- 消息类型：`woosh_robot_msgs/srv/ScannerData`

ros cli command：

```
ros2 service call /woosh_robot/robot/ScannerData woosh_robot_msgs/srv/ScannerData
"{arg:{}}"
```

参数说明详见 `ros2 interface show woosh_robot_msgs/srv/ScannerData --all-comments`

```
# 雷达数据请求

---
# 雷达点云数据
woosh_robot_msgs/ScannerData ret
    # scan的开始角度 [弧度]
    float32 angle_min
    # scan的结束角度 [弧度]
    float32 angle_max
    # 测量的角度间的距离 [弧度]
    float32 angle_increment
    # 测量间的时间 [秒]
    float32 time_increment
    # 扫描间的时间 [秒]
    float32 scan_time
    # 最小的测量距离 [米]
    float32 range_min
    # 最大的测量距离 [米]
    float32 range_max
    # 测量的距离数据 [米] (注意: 值 < range_min 或 > range_max 应当被丢弃)
    float32[] ranges
    # 位姿
    woosh_common_msgs/Pose2D pose
        # x
        float32 x
        # y
        float32 y
        # 朝向
        float32 theta
# 请求成功与否
bool ok
# 请求状态消息
string msg
```

执行预定义任务

- 接口类型：`service`
- 服务名称：`robot/ExecPreTask`
- 消息类型：`woosh_robot_msgs/srv/ExecPreTask`

ros cli command:

```
ros2 service call /woosh_robot/robot/ExecPreTask woosh_robot_msgs/srv/ExecPreTask
"{arg:{task_set_id: 666}}"
```

参数说明详见 `ros2 interface show woosh_robot_msgs/srv/ExecPreTask --all-comments`

```
# 执行预定义任务

woosh_robot_msgs/ExecPreTask arg
    # 预定义任务集ID
    int32 task_set_id
---
# 请求成功与否
bool ok
# 请求状态消息
string msg
```

执行任务请求

- 接口类型：`service`
- 服务名称：`robot/ExecTask`
- 消息类型：`woosh_robot_msgs/srv/ExecTask`

ros cli command:

```
ros2 service call /woosh_robot/robot/ExecTask woosh_robot_msgs/srv/ExecTask "
{arg:{type:{value: 1}, mark_no: A23}}"
```

参数说明详见 `ros2 interface show woosh_robot_msgs/srv/ExecTask --all-comments`

动作指令请求

- 接口类型：`service`
- 服务名称：`robot/ActionOrder`
- 消息类型：`woosh_robot_msgs/srv/ActionOrder`

ros cli command:

```
ros2 service call /woosh_robot/robot/ActionOrder woosh_robot_msgs/srv/ActionOrder
"{arg:{order:{value: 2}}}"
```

参数说明详见 `ros2 interface show woosh_robot_msgs/srv/ActionOrder --all-comments`

```
# 动作指令请求

woosh_robot_msgs/ActionOrder arg
    # 动作指令
    woosh_action_msgs/Order order
        # 未定义的
        int32 K_ORDER_UNDEFINED=0
        # 开始(弃用)
```

```
        int32 K_START=1
        # 暂停
        int32 K_PAUSE=2
        # 继续
        int32 K_CONTINUE=3
        # 取消
        int32 K_CANCEL=4
        # 恢复(单机任务有效)
        int32 K_RECOVER=5
        # 等待打断
        int32 K_WAIT_BREAK=6
        # 交通管制
        int32 K_TM_CTRL=7
        # 解除管制
        int32 K_RELEASE_CTRL=8

        int32 value
---
# 请求成功与否
bool ok
# 请求状态消息
string msg
```

改变导航路径

- 接口类型：`service`
- 服务名称：`robot/ChangeNavPath`
- 消息类型：`woosh_robot_msgs/srv/ChangeNavPath`

ros cli command：

```
ros2 service call /woosh_robot/robot/ChangeNavPath
woosh_robot_msgs/srv/ChangeNavPath "{arg:{paths:{plan_path:[{target:{x: 1.23, y:
2.34, theta: 1.57}, path:[{x: 0.0, y: 0.0, theta: 0.0}, {x: 1.23, y: 2.34, theta:
1.57}]}]}}"
```

参数说明详见 `ros2 interface show woosh_robot_msgs/srv/ChangeNavPath --all-comments`

```
# 改变导航路径请求

woosh_robot_msgs/ChangeNavPath arg
    # 导航路径集合
    woosh_robot_msgs/PlanPath paths
        # 全局规划路径
        woosh_nav_msgs/PlanPath[] plan_path
            # 导航路径，不能为空，只有一个值则表示由单机自主规划路径
            woosh_nav_msgs/Path path
                woosh_common_msgs/Pose2D[] poses
                    # x
                    float32 x
                    # y
                    float32 y
                    # 朝向
                    float32 theta
```

```
            # 路径所在地图ID
            uint32 map_id
            # 路径去往虫洞ID, 虫洞ID为0则表示该路径不经过虫洞
            uint32 wormhole_id
            # 虫洞到达的地图ID
            uint32 dest_map_id
            # 分段目标点
            woosh_common_msgs/Pose2D target
                # x
                float32 x
                # y
                float32 y
                # 朝向
                float32 theta
            # 路径优化
            woosh_nav_msgs/PlanPathOptimal optimal
                # 未定义的
                int32 K_OPTIMAL_UNDEFINED=0
                # 优化
                int32 K_OPTIMAL=1
                # 目标点优化
                int32 K_DEST_OPTIMAL=2
                # 严格的(禁用优化)
                int32 K_STRICT=9

                int32 value
---
# 请求成功与否
bool ok
# 请求状态消息
string msg
```

改变导航模式

- 接口类型：`service`
- 服务名称：`robot/ChangeNavMode`
- 消息类型：`woosh_robot_msgs/srv/ChangeNavMode`

ros cli command:

```
ros2 service call /woosh_robot/robot/ChangeNavMode
woosh_robot_msgs/srv/ChangeNavMode "{arg:{nav_mode:{type:{value: 2}, mode:{value:
1}}}}"
```

参数说明详见 `ros2 interface show woosh_robot_msgs/srv/ChangeNavMode --all-comments`

```
# 改变导航模式请求

woosh_robot_msgs/ChangeNavMode arg
    # 导航模式设置
    woosh_nav_msgs/ModeSetting nav_mode
        # 导航到达类型
        woosh_nav_msgs/ArrType type
            # 未定义的
```

```
            int32 K_ARR_TYPE_UNDEFINED=0
            # 模糊到达
            int32 K_VAGUE=1
            # 精准到达
            int32 K_ACCURATE=2

            int32 value
        # 导航模式
        woosh_nav_msgs/Mode mode
            # 未定义的
            int32 K_MODE_UNDEFINED=0
            # 导航避障
            int32 K_AVOID=1
            # 等待...
            int32 K_NAV_WAIT=2
            # 等待.超时.重新规划
            int32 K_TIMEOUT=3
            # 等待.超时.导航失败
            int32 K_OVERTIME=4
            # 狭窄通道
            int32 K_NARROW=10
            # 磁条导航
            int32 K_MAGNETIC=11
            # 二维码导航
            int32 K_QRCODE=12

            int32 value
        # nav_mode为kTimeout时有效, 该参数指定超时时间(秒)
        uint32 wait_timeout
        # 导航的最大速度, 为0时取默认速度
        float32 max_speed
        # 是否允许通行
        bool permitted_passage
        # 通道车量
        int32 capacity
    # 域入口点
    woosh_common_msgs/Pose2D in_point
        # x
        float32 x
        # y
        float32 y
        # 朝向
        float32 theta
    # 域出口点
    woosh_common_msgs/Pose2D out_point
        # x
        float32 x
        # y
        float32 y
        # 朝向
        float32 theta
---
# 请求成功与否
bool ok
# 请求状态消息
string msg
```

语音播报

- 接口类型：`service`
- 服务名称：`robot/Speak`
- 消息类型：`woosh_robot_msgs/srv/Speak`

ros cli command：

```
ros2 service call /woosh_robot/robot/Speak woosh_robot_msgs/srv/Speak "{arg:
{text: "Hello\ World"}}"
```

参数说明详见 `ros2 interface show woosh_robot_msgs/srv/Speak --all-comments`

```
# 语音播报请求

woosh_robot_msgs/Speak arg
    # 语音合成的内容，为空则停止播报
    string text
---
# 请求成功与否
bool ok
# 请求状态消息
string msg
```

速度控制(遥控)

- 接口类型：`service`
- 服务名称：`robot/Twist`
- 消息类型：`woosh_robot_msgs/srv/Twist`

ros cli command：

```
ros2 service call /woosh_robot/robot/Twist woosh_robot_msgs/srv/Twist "{arg:
{linear: 0.2, angular: 0.785}}"
```

说明：此接口需要持续请求，停止请求后小车将平滑减速至0速度，需要小车立马停止需要主动发送0速度，即: `{arg:{linear: 0.0, angular: 0.0}}`

参数说明详见 `ros2 interface show woosh_robot_msgs/srv/Twist --all-comments`

```
# 速度控制请求

woosh_robot_msgs/Twist arg
    # 线速度，单位为m/s，正值往前
    float32 linear
    # 角速度，单位为弧度/s，正值逆时针旋转
    float32 angular
    # 线速度y，单位为m/s，正值往前
    float32 linear_y
---
# 请求成功与否
```

```
bool ok
# 请求状态消息
string msg
```

跟随请求

- 接口类型：`service`
- 服务名称：`robot/Follow`
- 消息类型：`woosh_robot_msgs/srv/Follow`

ros cli command：

```
ros2 service call /woosh_robot/robot/Follow woosh_robot_msgs/srv/Follow "{arg:
{type: true}}"
```

参数说明详见 `ros2 interface show woosh_robot_msgs/srv/Follow --all-comments`

```
# 跟随请求

woosh_robot_msgs/Follow arg
    # 1:开启自动跟随, 0: 关闭自动跟随
    bool type
---
# 请求成功与否
bool ok
# 请求状态消息
string msg
```

机器人设置相关

设置机器人标识

- 接口类型：`service`
- 服务名称：`setting/SetIdentity`
- 消息类型：`woosh_robot_msgs/srv/SetIdentity`

ros cli command：

```
ros2 service call /woosh_robot/setting/Identity woosh_robot_msgs/srv/SetIdentity
"{arg:{name: "woow"}}"
```

参数说明详见 `ros2 interface show woosh_robot_msgs/srv/SetIdentity --all-comments`

```
# 设置标识

woosh_robot_msgs/Identity arg
    # 机器人昵称
    string name
---
# 机器人标识
woosh_robot_msgs/Identity ret
    # 机器人昵称
```

```
    string name
# 请求成功与否
bool ok
# 请求状态消息
string msg
```

设置服务器连接

- 接口类型：`service`
- 服务名称：`setting/Server`
- 消息类型：`woosh_robot_msgs/srv/SetServer`

ros cli command:

```
ros2 service call /woosh_robot/setting/Server woosh_robot_msgs/srv/SetServer "
{arg:{name: "woow"}}"
```

参数说明详见 `ros2 interface show woosh_robot_msgs/srv/SetServer --all-comments`

```
# 设置连接服务器地址

woosh_robot_msgs/Server arg
    # 服务器IP
    string ip
    # 服务器端口
    uint32 port
---
# 连接服务器地址
woosh_robot_msgs/Server ret
    # 服务器IP
    string ip
    # 服务器端口
    uint32 port
# 请求成功与否
bool ok
# 请求状态消息
string msg
```

开关自主充电

- 接口类型：`service`
- 服务名称：`setting/AutoCharge`
- 消息类型：`woosh_robot_msgs/srv/SetAutoCharge`

ros cli command:

```
ros2 service call /woosh_robot/setting/AutoCharge
woosh_robot_msgs/srv/SetAutoCharge "{arg:{allow: true}}"
```

参数说明详见 `ros2 interface show woosh_robot_msgs/srv/SetAutoCharge --all-comments`

```
# 开关自主回充
```

```
woosh_robot_msgs/AutoCharge arg
    # 是否允许
    bool allow
---
# 自主回充
woosh_robot_msgs/AutoCharge ret
    # 是否允许
    bool allow
# 请求成功与否
bool ok
# 请求状态消息
string msg
```

开关自主泊车

- 接口类型：`service`
- 服务名称：`setting/AutoPark`
- 消息类型：`woosh_robot_msgs/srv/SetAutoPark`

ros cli command：

```
ros2 service call /woosh_robot/setting/AutoPark woosh_robot_msgs/srv/SetAutoPark
"{arg:{allow: true}}"
```

参数说明详见 `ros2 interface show woosh_robot_msgs/srv/SetAutoPark --all-comments`

```
# 开关自主泊车

woosh_robot_msgs/AutoPark arg
    # 是否允许
    bool allow
---
# 自主泊车
woosh_robot_msgs/AutoPark ret
    # 是否允许
    bool allow
# 请求成功与否
bool ok
# 请求状态消息
string msg
```

开关货物检测

- 接口类型：`service`
- 服务名称：`setting/GoodsCheck`
- 消息类型：`woosh_robot_msgs/srv/SetGoodsCheck`

ros cli command：

```
ros2 service call /woosh_robot/setting/GoodsCheck
woosh_robot_msgs/srv/SetGoodsCheck "{arg:{allow: true}}"
```

参数说明详见 `ros2 interface show woosh_robot_msgs/srv/SetGoodsCheck --all-comments`

```
# 开关货物检测

woosh_robot_msgs/GoodsCheck arg
    # 是否允许
    bool allow
---
# 货物检测
woosh_robot_msgs/GoodsCheck ret
    # 是否允许
    bool allow
# 请求成功与否
bool ok
# 请求状态消息
string msg
```

充电电量配置

- 接口类型：`service`
- 服务名称：`setting/Power`
- 消息类型：`woosh_robot_msgs/srv/SetPower`

ros cli command：

```
ros2 service call /woosh_robot/setting/Power woosh_robot_msgs/srv/SetPower "{arg:
{alarm: 5, low: 20, idle: 80, full: 100}}"
```

参数说明详见 `ros2 interface show woosh_robot_msgs/srv/SetPower --all-comments`

```
# 电量配置

woosh_robot_msgs/Power arg
    # 警告电量值
    uint32 alarm
    # 低电量值
    uint32 low
    # 空闲电量值
    uint32 idle
    # 满电量值
    uint32 full
---
# 电量配置
woosh_robot_msgs/Power ret
    # 警告电量值
    uint32 alarm
    # 低电量值
    uint32 low
    # 空闲电量值
    uint32 idle
    # 满电量值
    uint32 full
# 请求成功与否
```

```
bool ok
# 请求状态消息
string msg
```

系统声音设置

- 接口类型：`service`
- 服务名称：`setting/Sound`
- 消息类型：`woosh_robot_msgs/srv/SetSound`

ros cli command：

```
ros2 service call /woosh_robot/setting/Sound woosh_robot_msgs/srv/SetSound "{arg:
{mute: false, volume: 50}}"
```

参数说明详见 `ros2 interface show woosh_robot_msgs/srv/SetSound --all-comments`

```
# 声音设置

woosh_robot_msgs/Sound arg
    # 静音
    bool mute
    # 音量
    uint32 volume
---
# 声音设置
woosh_robot_msgs/Sound ret
    # 机器人声音设置

    # 静音
    bool mute
    # 音量
    uint32 volume
# 请求成功与否
bool ok
# 请求状态消息
string msg
```

地图相关

获取场景数据

- 接口类型：`service`
- 服务名称：`map/SceneDataEasy`
- 消息类型：`woosh_map_msgs/srv/SceneDataEasy`

ros cli command：

```
ros2 service call /woosh_robot/map/SceneDataEasy woosh_map_msgs/srv/SceneDataEasy
"{}"
```

参数说明详见 `ros2 interface show woosh_map_msgs/srv/SceneDataEasy --all-comments`

```
# 获取场景数据(Easy)

---
woosh_map_msgs/SceneDataEasy info
    # 场景数据简单版

    # 场景名
    string name
    # 地图信息数据
    woosh_map_msgs/SceneDataEasyMap[] maps
        # 地图ID
        uint32 id
        # 地图名
        string name
        # 楼层名
        string floor
        # 地图版本
        int64 version
        # 储位集
        woosh_map_msgs/Storages storages
            # 储位集合
            woosh_map_msgs/StoragesBase[] bases
                # 标识
                woosh_map_msgs/Identity identity
                    # ID(唯一)
                    uint32 id
                    # 编号(唯一)
                    string no
                    # 描述
                    string desc
                # 位姿
                woosh_map_msgs/Pose pose
                    # 对接点坐标
                    woosh_common_msgs/Pose2D dock
                        # x
                        float32 x
                        # y
                        float32 y
                        # 朝向
                        float32 theta
                    # 实际坐标
                    woosh_common_msgs/Pose2D real
                        # x
                        float32 x
                        # y
                        float32 y
                        # 朝向
                        float32 theta
                # 自定义字段
                uint8[] custom
```

任务相关

获取预定义任务列表

- 接口类型：`service`
- 服务名称：`task/RepeatTasks`
- 消息类型：`woosh_task_msgs/srv/RepeatTasks`

ros cli command：

```
ros2 service call /woosh_robot/task/RepeatTasks woosh_task_msgs/srv/RepeatTasks "
{}"
```

参数说明详见 `ros2 interface show woosh_task_msgs/srv/RepeatTasks --all-comments`

获取呼叫任务列表

- 接口类型：`service`
- 服务名称：`task/CallTasks`
- 消息类型：`woosh_task_msgs/srv/CallTasks`

ros cli command：

```
ros2 service call /woosh_robot/task/CallTasks woosh_task_msgs/srv/CallTasks "{}"
```

参数说明详见 `ros2 interface show woosh_task_msgs/srv/CallTasks --all-comments`

Action

任务执行

- 接口类型：`action`
- 服务名称：`robot/ExecTask`
- 消息类型：`woosh_robot_msgs/action/ExecTask`

ros cli command：

```
ros2 action send_goal /woosh_robot/robot/ExecTask
woosh_robot_msgs/action/ExecTask "{arg:{type:{value: 1}, mark_no: A2}}" --
feedback
```

参数说明详见 `ros2 interface show woosh_robot_msgs/action/ExecTask --all-comments`

```
# 任务执行

woosh_robot_msgs/ExecTask arg
    # 任务ID
    int64 task_id
    # 任务类型
    woosh_task_msgs/Type type
        # 任务类型
```

```
        # 未定义的
        int32 K_TYPE_UNDEFINED=0
        # 拣选
        int32 K_PICK=1
        # 泊车
        int32 K_PARKING=2
        # 充电
        int32 K_CHARGE=3
        # 搬运
        int32 K_CARRY=4

        int32 value
    # 动作方向
    woosh_task_msgs/Direction direction
        # 方向

        # 未定义的
        int32 K_DIRECTION_UNDEFINED=0
        # 上料
        int32 K_FEEDING=1
        # 下料
        int32 K_CUTTING=2

        int32 value
    # 类型组合
    uint32 task_type_no
    # 目标点编号(三选一)
    string mark_no
    # 导航路径集合(三选一)
    woosh_robot_msgs/PlanPath plan_path
        # 机器人全局规划路径

        # 全局规划路径
        woosh_nav_msgs/PlanPath[] plan_path
            # 规划的路径

            uint8 PATH_FIELD_SET=1
            uint8 TARGET_FIELD_SET=16

            # 导航路径，不能为空，只有一个值则表示由单机自主规划路径
            woosh_nav_msgs/Path path
                # 路径(...)
    # 位姿(三选一)
    woosh_common_msgs/Pose2D pose
        # x
        float32 x
        # y
        float32 y
        # 朝向
        float32 theta
    # 自定义字段，因项目而异
    uint8[] custom
---
woosh_robot_msgs/TaskProc ret
    # 机器人任务ID
```

```
int64 robot_task_id
# 任务类型
woosh_task_msgs/Type type
    # 未定义的
    int32 K_TYPE_UNDEFINED=0
    # 拣选
    int32 K_PICK=1
    # 泊车
    int32 K_PARKING=2
    # 充电
    int32 K_CHARGE=3
    # 搬运
    int32 K_CARRY=4

    int32 value
# 任务状态
woosh_task_msgs/State state
    # 未定义的
    int32 K_STATE_UNDEFINED=0
    # 初始化
    int32 K_INIT=1
    # 准备的
    int32 K_READY=2
    # 执行中
    int32 K_EXECUTING=3
    # 暂停的
    int32 K_PAUSED=4
    # 动作等待
    int32 K_ACTION_WAIT=5
    # 任务等待
    int32 K_TASK_WAIT=6
    # 完成的
    int32 K_COMPLETED=7
    # 取消的
    int32 K_CANCELED=8
    # 失败的
    int32 K_FAILED=9

    int32 value
# 动作信息
woosh_robot_msgs/TaskProcAction action
    # 动作类型
    woosh_action_msgs/Type type
        # 未定义的
        int32 K_TYPE_UNDEFINED=0
        # 导航
        int32 K_NAV=1
        # 单步控制
        int32 K_STEP_CTRL=2
        # 二次定位进入
        int32 K_SECONDPOS_ENTER=3
        # 二次定位退出
        int32 K_SECONDPOS_QUIT=4
        # 搬运动作
        int32 K_CARRY=5
        # 等待
```

```
            int32 K_WAIT=6
            # 充电
            int32 K_CHARGE=7

            int32 value
        # 动作状态
        woosh_action_msgs/State state
            # 未定义的
            int32 K_STATE_UNDEFINED=0
            # 执行中
            int32 K_ROS_EXECUTING=1
            # 警告
            int32 K_ROS_WARNING=2
            # 取消
            int32 K_ROS_CANCEL=3
            # 完成
            int32 K_ROS_SUCCESS=4
            # 失败
            int32 K_ROS_FAILURE=5
            # 暂停
            int32 K_SUSPEND=10
            # 管制
            int32 K_TRAFFI_CTRL=11

            int32 value
        # 动作等待ID
        int32 wait_id
    # 目的地
    string dest
    # 消息
    string msg
    # 最后更新时间(s)
    int32 time
---
woosh_robot_msgs/TaskProc fb
    # 机器人任务ID
    int64 robot_task_id
    # 任务类型
    woosh_task_msgs/Type type
        # 未定义的
        int32 K_TYPE_UNDEFINED=0
        # 拣选
        int32 K_PICK=1
        # 泊车
        int32 K_PARKING=2
        # 充电
        int32 K_CHARGE=3
        # 搬运
        int32 K_CARRY=4

        int32 value
    # 任务状态
    woosh_task_msgs/State state
        # 未定义的
        int32 K_STATE_UNDEFINED=0
        # 初始化
```

```
        int32 K_INIT=1
        # 准备的
        int32 K_READY=2
        # 执行中
        int32 K_EXECUTING=3
        # 暂停的
        int32 K_PAUSED=4
        # 动作等待
        int32 K_ACTION_WAIT=5
        # 任务等待
        int32 K_TASK_WAIT=6
        # 完成的
        int32 K_COMPLETED=7
        # 取消的
        int32 K_CANCELED=8
        # 失败的
        int32 K_FAILED=9

        int32 value
    # 动作信息
    woosh_robot_msgs/TaskProcAction action
        # 动作类型
        woosh_action_msgs/Type type
            # 未定义的
            int32 K_TYPE_UNDEFINED=0
            # 导航
            int32 K_NAV=1
            # 单步控制
            int32 K_STEP_CTRL=2
            # 二次定位进入
            int32 K_SECONDPOS_ENTER=3
            # 二次定位退出
            int32 K_SECONDPOS_QUIT=4
            # 搬运动作
            int32 K_CARRY=5
            # 等待
            int32 K_WAIT=6
            # 充电
            int32 K_CHARGE=7

            int32 value
        # 动作状态
        woosh_action_msgs/State state
            # 未定义的
            int32 K_STATE_UNDEFINED=0
            # 执行中
            int32 K_ROS_EXECUTING=1
            # 警告
            int32 K_ROS_WARNING=2
            # 取消
            int32 K_ROS_CANCEL=3
            # 完成
            int32 K_ROS_SUCCESS=4
            # 失败
            int32 K_ROS_FAILURE=5
            # 暂停
```

```
            int32 K_SUSPEND=10
            # 管制
            int32 K_TRAFFI_CTRL=11

            int32 value
        # 动作等待ID
        int32 wait_id
    # 目的地
    string dest
    # 消息
    string msg
    # 最后更新时间(s)
    int32 time
```

步进控制

- 接口类型：`action`
- 服务名称：`ros/StepControl`
- 消息类型：`woosh_ros_msgs/action/StepControl`

ros cli command：

```
# 步进直行
ros2 action send_goal /woosh_robot/ros/StepControl
woosh_ros_msgs/action/StepControl "{arg:{action:{value: 1}, steps:[{mode:{value:
1}, speed: 0.5, value: 2}]}}" --feedback
# 步进旋转
ros2 action send_goal /woosh_robot/ros/StepControl
woosh_ros_msgs/action/StepControl "{arg:{action:{value: 1}, steps:[{mode:{value:
2}, speed: 0.78, value: 3.14}]}}" --feedback
```

参数说明详见 `ros2 interface show woosh_ros_msgs/action/StepControl --all-comments`

```
# 步进控制

woosh_ros_msgs/StepControl arg
    # 步进控制集
    woosh_ros_msgs/StepControlStep[] steps
        # 控制模式
        woosh_ros_msgs/StepControlStepMode mode
            # 未定义的
            int32 K_NONE=0
            # 直走
            int32 K_STRAIGHT=1
            # 旋转
            int32 K_ROTATE=2
            # 横移
            int32 K_LATERAL=3
            # 斜移
            int32 K_DIAGONALIZE=4

            int32 value
        # 旋转弧度/行驶距离，正前负后，正逆负顺，正左负右
        float32 value
```

```
        # 角速度(弧度/s)/线速度((米/s))
        float32 speed
        # 斜向运动角度, 正左负右
        float32 angle
    # 0:开启避障, 1:关闭避障
    int32 avoid
    # 控制动作
    woosh_ros_msgs/ControlAction action
        # 取消
        int32 K_CANCEL=0
        # 执行
        int32 K_EXECUTE=1
        # 暂停
        int32 K_PAUSE=2
        # 继续
        int32 K_RESUME=3

        int32 value
---
woosh_ros_msgs/Feedback ret
    # ros feedback

    # action name, e.g. woosh.ros.action.StepControl
    string action
    # 状态
    woosh_ros_msgs/State state
        # 未定义的
        int32 K_ROS_NONE=0
        # 取消
        int32 K_ROS_CANCEL=-2
        # 失败
        int32 K_ROS_FAILURE=-1
        # 完成
        int32 K_ROS_SUCCESS=1
        # 执行中
        int32 K_ROS_EXECUTING=2
        # 暂停
        int32 K_ROS_PAUSE=3
        # 暂停失败
        int32 K_ROS_PAUSE_FAILED=4
        # 执行失败
        int32 K_ROS_EXECUTE_FAILED=5
        # 异常消息
        int32 K_ROS_ERR_MSG=10
        # WiFi请求状态码
        int32 K_ROS_WI_FI_CODE=100
        # WiFi信息json
        int32 K_ROS_WI_FI_JSON=101

        int32 value
    # 状态码
    uint64 code
    # 消息
    string msg
---
woosh_ros_msgs/Feedback fb
```

```
    # action name, e.g. woosh.ros.action.StepControl
    string action
    # 状态
    woosh_ros_msgs/State state
        # 未定义的
        int32 K_ROS_NONE=0
        # 取消
        int32 K_ROS_CANCEL=-2
        # 失败
        int32 K_ROS_FAILURE=-1
        # 完成
        int32 K_ROS_SUCCESS=1
        # 执行中
        int32 K_ROS_EXECUTING=2
        # 暂停
        int32 K_ROS_PAUSE=3
        # 暂停失败
        int32 K_ROS_PAUSE_FAILED=4
        # 执行失败
        int32 K_ROS_EXECUTE_FAILED=5
        # 异常消息
        int32 K_ROS_ERR_MSG=10
        # WiFi请求状态码
        int32 K_ROS_WI_FI_CODE=100
        # WiFi信息json
        int32 K_ROS_WI_FI_JSON=101

        int32 value
    # 状态码
    uint64 code
    # 消息
    string msg
```

举升机构控制

- 接口类型：`action`
- 服务名称：`ros/LiftControl`
- 消息类型：`woosh_ros_msgs/action/LiftControl`

ros cli command：

```
# 举升上升
ros2 action send_goal /woosh_robot/ros/LiftControl
woosh_ros_msgs/action/LiftControl "{arg:{action:{value: 1}, execute_mode:{value:
1}}}" --feedback
# 举升下降
ros2 action send_goal /woosh_robot/ros/LiftControl
woosh_ros_msgs/action/LiftControl "{arg:{action:{value: 1}, execute_mode:{value:
2}}}" --feedback
```

参数说明详见 `ros2 interface show woosh_ros_msgs/action/LiftControl --all-comments`

```
# 举升机构控制
```

```
woosh_ros_msgs/LiftControl arg
    # 执行模式
    woosh_ros_msgs/LiftControlExecuteMode execute_mode
        int32 K_NONE_EXECUTE_MODE=0
        # 上升
        int32 K_UP=1
        # 下降
        int32 K_DOWN=2

        int32 value
    # 控制动作
    woosh_ros_msgs/ControlAction action
        # 取消
        int32 K_CANCEL=0
        # 执行
        int32 K_EXECUTE=1
        # 暂停
        int32 K_PAUSE=2
        # 继续
        int32 K_RESUME=3

        int32 value
---
woosh_ros_msgs/Feedback ret
---
woosh_ros_msgs/Feedback fb
```

升降机构控制

- 接口类型：`action`
- 服务名称：`ros/LiftControl3`
- 消息类型：`woosh_ros_msgs/action/LiftControl3`

ros cli command：

```
# 绝对位置
ros2 action send_goal /woosh_robot/ros/LiftControl3
woosh_ros_msgs/action/LiftControl3 "{arg:{action:{value: 1}, execute_mode:{value:
1}, speed: 0.2, height: 0.5}}" --feedback
# 相对位置
ros2 action send_goal /woosh_robot/ros/LiftControl3
woosh_ros_msgs/action/LiftControl3 "{arg:{action:{value: 1}, execute_mode:{value:
2}, speed: 0.2, height: 0.2}}" --feedback
```

参数说明详见 `ros2 interface show woosh_ros_msgs/action/LiftControl3 --all-comments`

```
# 升降机构控制3

woosh_ros_msgs/LiftControl3 arg
    # 执行模式
    woosh_ros_msgs/LiftControl3ExecuteMode execute_mode
        # 查询状态
        int32 K_QUERY=0
        # 绝对位置
```

```
        int32 K_ABSOLUTE=1
        # 相对位置
        int32 K_RELATIVE=2
        # 位置校准
        int32 K_CALIBRATION=3
        # 测试模式
        int32 K_TSET_MODE=4

        int32 value
    # 速度(米/s)
    float32 speed
    # 高度(米)，正值往上，负值往下
    float32 height
    uint32 flags
    # 控制动作
    woosh_ros_msgs/ControlAction action
        # 取消
        int32 K_CANCEL=0
        # 执行
        int32 K_EXECUTE=1
        # 暂停
        int32 K_PAUSE=2
        # 继续
        int32 K_RESUME=3

        int32 value
---
woosh_ros_msgs/Feedback ret
---
woosh_ros_msgs/Feedback fb
```

基础导航

- 接口类型：`action`
- 服务名称：`ros/MoveBase`
- 消息类型：`woosh_ros_msgs/action/MoveBase`

ros cli command：

```
ros2 action send_goal /woosh_robot/ros/MoveBase woosh_ros_msgs/action/MoveBase "
{arg:{poses:[{x: 0.57, y: 2.54, theta: 1.57}], target_pose:{x: 0.57, y: 2.54,
theta: 1.57}}}" --feedback
```

参数说明详见 `ros2 interface show woosh_ros_msgs/action/MoveBase --all-comments`

```
# 基础导航

woosh_ros_msgs/MoveBase arg
    # 导航路径
    woosh_common_msgs/Pose2D[] poses
        # x
        float32 x
        # y
        float32 y
```

```
        # 朝向
        float32 theta
    # 目标点
    woosh_common_msgs/Pose2D target_pose
        # x
        float32 x
        # y
        float32 y
        # 朝向
        float32 theta
    # 执行模式
    woosh_ros_msgs/MoveBaseExecutionMode execution_mode
        # 自由执行
        int32 K_FREE=0
        # 逐点执行
        int32 K_ONE_BY_ONE=1

        int32 value
    # 控制动作
    woosh_ros_msgs/ControlAction action
        # 取消
        int32 K_CANCEL=0
        # 执行
        int32 K_EXECUTE=1
        # 暂停
        int32 K_PAUSE=2
        # 继续
        int32 K_RESUME=3

        int32 value
---
woosh_ros_msgs/Feedback ret
---
woosh_ros_msgs/Feedback fb
```