

无际轮式人形机器人开发接口文档

1. 文档概述

本文档提供了自研机器人 gRPC 控制接口的完整规范说明。该 API 采用 Protocol Buffers v3 (<https://grpc.io/>) 接口定义语言描述，定义了机器人各模块（手臂、腰部、头部、升降系统、底盘、相机）的控制接口和数据格式。通过 gRPC 框架，客户端可以方便地与机器人硬件进行通信，实现实时控制、状态监控和数据采集等功能。

1.1 接口特性

- 支持单向 RPC 调用和双向流式通信
- 使用标准 gRPC 错误处理机制
- 包含完善的模块使能/禁用控制
- 提供详细的时序和状态信息

2 基础数据类型

2.1 通用状态响应

```
message StatusResponse {  
  bool success = 1;    // 操作是否成功  
  string message = 2;  // 详细结果描述或错误信息  
}
```

该消息类型作为所有操作的通用响应格式，success 字段表示操作执行结果，message 字段提供可读的详细信息。

2.2 模块状态枚举

```
enum ModuleState {  
    MODULE_STATE_UNSPECIFIED = 0; // 未指定状态  
    MODULE_ENABLED = 1;           // 模块已使能（正常工作）  
    MODULE_DISABLED = 2;          // 模块已禁用（停止工作）  
}
```

2.3 模块使能控制

```
message ModuleEnableRequest {  
    bool enable = 1; // true: 使能模块, false: 禁用模块  
}  
  
message ModuleEnableResponse {  
    StatusResponse status = 1; // 操作结果  
    ModuleState current_state = 2; // 当前模块状态  
}
```

所有服务模块均使用相同的使能控制接口，保证一致性。

2.4 位姿表示（坐标系定义）

```
message Pose {  
    double x = 1; // X 轴坐标（米），机器人本体坐标系，X 轴向前  
    double y = 2; // Y 轴坐标（米），机器人本体坐标系，Y 轴向左  
    double z = 3; // Z 轴坐标（米），机器人本体坐标系，Z 轴向上  
    double qx = 4; // 四元数 x 分量（姿态）  
    double qy = 5; // 四元数 y 分量  
    double qz = 6; // 四元数 z 分量  
    double qw = 7; // 四元数 w 分量（实部）  
}
```

位姿表示采用机器人本体坐标系，坐标原点为机器人本体中心，使用四元数表示姿态，避免万向节锁问题。

3 服务接口详情

3.1 手臂控制服务（ArmService）

手臂服务提供机器人手臂的关节控制、位姿控制和状态监控功能，支持流式传输和单次调用两种模式。

3.1.1 关节状态控制

方法名称	请求类型	响应类型	调用类型	说明
StreamGetJointStates	GetJointStatesRequest	stream JointStateResponse	服务器流	实时推送关节状态（角度、电流、功率）
StreamSetJointStates	stream SetJointStatesRequest	google.protobuf.Empty	双向流	实时控制关节角度（遥操作场景）
SetJointStates	SetJointStatesRequest	SetJointStatesResponse	一元调用	设置目标关节角度（常规控制）

GetJointStatesRequest：指定要查询的手臂。

```
message GetJointStatesRequest {
  ArmType arm_type = 1; // 手臂标识（左臂/右臂）
}
```

JointStateResponse：返回关节状态数据。

```
message JointStateResponse {
  ArmType arm_type = 1;           // 手臂标识
  repeated JointState joints = 2; // 6 个关节的状态信息
  google.protobuf.Timestamp timestamp = 3; // 数据时间戳
}
```

JointState：单个关节的详细状态。

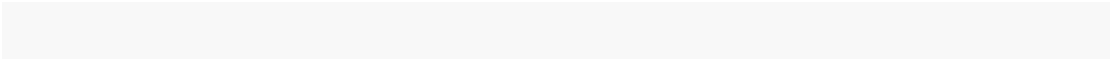
```
message JointState {
  int32 joint_id = 1; // 关节 ID (1 - 6)
  double angle_deg = 2; // 当前角度 (度)
  double current_a = 3; // 电流 (安培)
  double power_w = 4; // 功率 (瓦特)
}
```

3.1.2 位姿控制接口

方法名称	请求类型	响应类型	调用类型	说明
SetArmPose	SetArmPoseRequest	SetArmPoseResponse	一元调用	设置手臂在本体坐标系下的目标位姿
StreamGetArmPose	GetArmPoseRequest	stream ArmPoseResponse	服务器流	实时获取手臂位姿信息

SetArmPoseRequest：手臂位姿控制请求。

```
message SetArmPoseRequest {
  ArmType arm_type = 1; // 手臂标识
  Pose target_pose = 2; // 目标位姿 (本体坐标系)
  double speed_ratio = 3; // 运动速度比例 (0 - 1, 默认 0.5)
}
```



SetArmPoseResponse：位姿控制响应。

```
message SetArmPoseResponse {
  StatusResponse status = 1; // 操作状态
  bool is_reachable = 2;     // 目标位姿是否完全可达
  Pose actual_pose = 3;      // 实际可达位姿
  string ik_error = 4;       // 逆运动学求解错误信息
}
```

3.2 腰部俯仰控制服务（WaistService）

腰部服务控制机器人的俯仰动作，提供角度设置和状态查询功能。

3.2.1 接口方法

方法名称	请求类型	响应类型	说明
SetPitchAngle	SetWaistPitchRequest	SetWaistPitchResponse	设置腰部俯仰角度
GetCurrentPitch	google.protobuf.Empty	GetWaistPitchResponse	查询当前俯仰角度

SetWaistPitchRequest：设置俯仰角度请求。

```
message SetWaistPitchRequest {
  double pitch_angle_deg = 1; // 俯仰角度（度），正值前俯
}
```

GetWaistPitchResponse：当前俯仰角度响应。

```
message GetWaistPitchResponse {
  double current_pitch_deg = 1; // 当前俯仰角度
  google.protobuf.Timestamp timestamp = 2; // 时间戳
}
```

3.3 头部控制服务（HeadService）

头部服务控制机器人的头部运动，包括俯仰（pitch）和偏航（yaw）两个自由度。

3.3.1 接口方法

方法名称	请求类型	响应类型	说明
SetHeadPitch	SetHeadPitchRequest	SetHeadPitchResponse	设置头部俯仰角度
GetHeadPitch	google.protobuf.Empty	GetHeadPitchResponse	查询头部俯仰角度
SetHeadYaw	SetHeadYawRequest	SetHeadYawResponse	设置头部偏航角度
GetHeadYaw	google.protobuf.Empty	GetHeadYawResponse	查询头部偏航角度

角度控制参数：

- 俯仰角度：pitch_angle_deg – 头部上下摆动角度
- 偏航角度：yaw_angle_deg – 头部左右转动角度，正数左转

3.4 升降服务控制（LiftService）

升降服务控制机器人的高度调节，采用比例值控制方式。

3.4.1 接口方法

方法名称	请求类型	响应类型	说明
SetLiftHeight	SetLiftHeightRequest	SetLiftHeightResponse	设置升降高度比例
GetCurrentLiftHeight	google.protobuf.Empty	GetLiftHeightResponse	查询当前高度比例

SetLiftHeightRequest：高度设置请求。

```
message SetLiftHeightRequest {  
  double scale = 1; // 高度比例值[0,1], 0 为最低, 1 为最高  
}
```

3.5 底盘服务控制（BaseService）

底盘服务提供机器人的移动控制功能，支持实时控制和导航控制两种模式。

3.5.1 实时运动控制

方法名称	请求类型	响应类型	说明
RealTimeTranslate	RealTimeTranslateRequest	ControlResponse	实时平移控制 (持续速度)
RealTimeRotate	RealTimeRotateRequest	ControlResponse	实时旋转控制 (持续速度)

RealTimeTranslateRequest：实时平移请求。

```
message RealTimeTranslateRequest {
    double speed_mps = 1;    // 平移速度（米/秒，>0 有效）
    double direction_deg = 2; // 方向角（度）：正前方为 0，向左为正
}
```

RealTimeRotateRequest：实时旋转请求。

```
message RealTimeRotateRequest {
    double speed_degps = 1;    // 旋转速度（度/秒，>0 有效）
    RotationDirection direction = 2; // 旋转方向（LEFT/RIGHT）
}
```

3.5.2 常规运动控制

方法名称	请求类型	响应类型	说明
TranslateWithDistance	TranslateWithDistanceRequest	ControlResponse	带距离的平移控制
RotateWithAngle	RotateWithAngleRequest	ControlResponse	带角度的旋转控制

TranslateWithDistanceRequest：常规平移请求。

```
message TranslateWithDistanceRequest {
    double speed_mps = 1;    // 平移速度（米/秒）
    double direction_deg = 2; // 方向角
    double distance_m = 3;   // 移动距离（米，>0 有效）
}
```

3.5.3 导航与状态查询

方法名称	请求类型	响应类型	说明
------	------	------	----

NavigateTo	NavigateRequest	NavigateResponse	导航到目标位置
Stop	google.protobuf.Empty	StatusResponse	急停（立即停止所有运动）
GetBaseStatus	google.protobuf.Empty	BaseStatusResponse	查询底盘状态

BaseStatusResponse：底盘状态信息。

```
message BaseStatusResponse {
  string navi_status = 1; // 导航状态（IDLE/NAVIGATING/FAILED）
  float x = 2;           // 当前 X 坐标（米，全局坐标系）
  float y = 3;           // 当前 Y 坐标（米，全局坐标系）
  float yaw = 4;         // 当前朝向角（度）
  int32 battery = 5;     // 电池电量（百分比，0 - 100）
  google.protobuf.Timestamp update_time = 6; // 状态更新时间
}
```

3.6 相机服务（CameraService）

相机服务提供 2D 图像和 RGBD 数据获取功能，支持单帧获取和流式传输两种模式。

3.6.1 图像数据获取

方法名称	请求类型	响应类型	调用类型	说明
StreamGetImage2D	CameraRequest	stream Image2D	服务器流	流式获取 2D 图像（JPEG）

GetRGBDImage	CameraRequest	RGBDImageResponse	一元调用	获取单帧 RGBD 图像
StreamGetRGBDImage	CameraRequest	stream RGBDImage	服务器流	流式获取 RGBD 图像

CameraRequest：通用相机请求。

```
message CameraRequest {
  CameraType camera_type = 1; // 相机类型（头部/胸部/左手/右手）
}
```

Image2D：2D 图像数据。

```
message Image2D {
  bytes jpeg_data = 1; // JPEG 二进制数据
  google.protobuf.Timestamp timestamp = 2; // 采集时间戳
}
```

3.6.2 相机参数管理

方法名称	请求类型	响应类型	说明
GetCameraIntrinsics	CameraRequest	CameraIntrinsicsResponse	获取相机内参
ControlDepthStream	DepthStreamControlRequest	ControlDepthStreamResponse	控制深度数据流

CameraIntrinsics：相机内参。

```
message CameraIntrinsics {
  double fx = 1; // 焦距 x
  double fy = 2; // 焦距 y
}
```

```
double cx = 3;          // 主点 x
double cy = 4;          // 主点 y
repeated double distortion = 5; // 畸变系数
int32 base_width = 6;    // 图像宽度（像素）
int32 base_height = 7;   // 图像高度（像素）
}
```

4 完整 Proto 源码

4.1 基础接口（common.proto）

protobuf

```
syntax = "proto3";

package robot.api.v1;

import "google/protobuf/timestamp.proto";
import "google/protobuf/empty.proto";

// 通用状态响应
message StatusResponse {
    bool success = 1;
    string message = 2;
}

enum ModuleState {
    MODULE_STATE_UNSPECIFIED = 0;
    MODULE_ENABLED = 1; // 已使能（正常工作）
    MODULE_DISABLED = 2; // 已禁用（停止工作）
}

// 使能控制请求（通用结构，所有模块的开关请求都用这个）
message ModuleEnableRequest {
    bool enable = 1; // true: 使能, false: 禁用
}

// 使能控制响应（通用结构，返回当前状态）
message ModuleEnableResponse {
    StatusResponse status = 1; // 操作结果（成功/失败原因）
    ModuleState current_state = 2; // 当前模块状态（使能/禁用）
}
```

4.2 手臂控制（arm.proto）

protobuf

```
syntax = "proto3";

package robot.api.v1;

import "common.proto";
import "google/protobuf/empty.proto";
import "google/protobuf/timestamp.proto";

// 手臂标识
enum ArmType {
    ARM_UNSPECIFIED = 0;
    ARM_LEFT = 1;
    ARM_RIGHT = 2;
}

// 坐标系位姿表示（机器人本体坐标系，四元数姿态）
// 坐标原点：机器人本体中心，X轴向前，Y轴向左，Z轴向上
message Pose {
    double x = 1; // X轴坐标（单位：米）
    double y = 2; // Y轴坐标（单位：米）
    double z = 3; // Z轴坐标（单位：米）
    double qx = 4; // 四元数 x 分量（姿态）
    double qy = 5; // 四元数 y 分量
    double qz = 6; // 四元数 z 分量
    double qw = 7; // 四元数 w 分量（实部）
}

service ArmService {
    // 流式获取关节状态（实时数据推送）
    rpc StreamGetJointStates(GetJointStatesRequest) returns (stream JointStateResponse);
    // 流式设置关节状态（实时数据发送，主要用于遥操作）
    rpc StreamSetJointStates(stream SetJointStatesRequest) returns (google.protobuf.Empty);
    // 设置目标关节角度
    rpc SetJointStates(SetJointStatesRequest) returns (SetJointStatesResponse);

    // 新增：设置手臂在本体坐标系下的目标位姿（四元数控制）
    rpc SetArmPose(SetArmPoseRequest) returns (SetArmPoseResponse);
    // 新增：流式获取手臂在本体坐标系下的实时位姿（含四元数）
    rpc StreamGetArmPose(GetArmPoseRequest) returns (stream ArmPoseResponse);

    rpc SetEnabled(ModuleEnableRequest) returns (ModuleEnableResponse);
    rpc GetEnabled(google.protobuf.Empty) returns (ModuleEnableResponse);
}

message GetJointStatesRequest {
    ArmType arm_type = 1;
}

message JointStateResponse {
    ArmType arm_type = 1;
```

```

repeated JointState joints = 2; // 6 个关节状态
google.protobuf.Timestamp timestamp = 3;
}

message JointState {
    int32 joint_id = 1; // 关节 ID (1-6)
    double angle_deg = 2; // 当前角度 (度)
    double current_a = 3; // 电流 (安培)
    double power_w = 4; // 功率 (瓦特)
}

message JointRequest {
    int32 joint_id = 1; // 关节 ID (1-6, 唯一标识)
    double target_angle_deg = 2; // 该关节的目标角度
    double speed_ratio = 3; // 该关节的速度比例 (0-1, 可选, 默认用全局值)
}

message SetJointStatesRequest {
    ArmType arm_type = 1;
    repeated JointRequest joint_request = 2; // 多个目标关节请求
}

message SetJointStatesResponse {
    StatusResponse status = 1;
}

// 新增: 获取手臂位姿请求
message GetArmPoseRequest {
    ArmType arm_type = 1; // 指定手臂
}

// 新增: 手臂位姿响应 (实时数据)
message ArmPoseResponse {
    ArmType arm_type = 1; // 手臂标识
    Pose pose = 2; // 本体坐标系下位姿 (含四元数)
    google.protobuf.Timestamp timestamp = 3; // 时间戳
    bool is_reachable = 4; // 该位姿是否在可达范围内
}

// 新增: 设置手臂位姿请求 (四元数)
message SetArmPoseRequest {
    ArmType arm_type = 1; // 手臂标识
    Pose target_pose = 2; // 目标位姿 (本体坐标系, 四元数)
    double speed_ratio = 3; // 运动速度比例 (0-1, 可选, 默认 0.5)
}

// 新增: 设置手臂位姿响应
message SetArmPoseResponse {
    StatusResponse status = 1;
    bool is_reachable = 2; // 目标位姿是否可达
    Pose actual_pose = 3; // 实际到达的位姿 (若已完成运动)
    string ik_error = 4; // 逆运动学求解错误信息 (若失败)
}

```

4.3 腰部俯仰控制 (waist.proto)

protobuf

```
syntax = "proto3";

package robot.api.v1;

import "common.proto";

service WaistService {
  rpc SetPitchAngle(SetWaistPitchRequest) returns (SetWaistPitchResponse);
  rpc GetCurrentPitch(google.protobuf.Empty) returns (GetWaistPitchResponse);

  rpc SetEnabled(ModuleEnableRequest) returns (ModuleEnableResponse);
  rpc GetEnabled(google.protobuf.Empty) returns (ModuleEnableResponse);
}

message SetWaistPitchRequest {
  double pitch_angle_deg = 1; // 俯仰角度（度），正值前俯
}

message SetWaistPitchResponse {
  StatusResponse status = 1;
}

message GetWaistPitchResponse {
  double current_pitch_deg = 1;
  google.protobuf.Timestamp timestamp = 2;
}
```

4.4 头部控制 (head.proto)

protobuf

```
syntax = "proto3";

package robot.api.v1;

import "common.proto";

service HeadService {
  // 俯仰控制
  rpc SetHeadPitch(SetHeadPitchRequest) returns (SetHeadPitchResponse);
  rpc GetHeadPitch(google.protobuf.Empty) returns (GetHeadPitchResponse);

  // 旋转控制
  rpc SetHeadYaw(SetHeadYawRequest) returns (SetHeadYawResponse);
  rpc GetHeadYaw(google.protobuf.Empty) returns (GetHeadYawResponse);
}
```

```

    rpc SetEnabled(ModuleEnableRequest) returns (ModuleEnableResponse);
    rpc GetEnabled(google.protobuf.Empty) returns (ModuleEnableResponse);
}

message SetHeadPitchRequest {
    double pitch_angle_deg = 1; // 俯仰角度
}

message SetHeadPitchResponse {
    StatusResponse status = 1;
}

message GetHeadPitchResponse {
    double current_pitch_deg = 1;
    google.protobuf.Timestamp timestamp = 2;
}

message SetHeadYawRequest {
    double yaw_angle_deg = 1; // 偏航角度，正数左转
}

message SetHeadYawResponse {
    StatusResponse status = 1;
}

message GetHeadYawResponse {
    double current_yaw_deg = 1;
    google.protobuf.Timestamp timestamp = 2;
}

```

4.5 升降服务控制 (lift.proto)

protobuf

```

syntax = "proto3";

package robot.api.v1;

import "common.proto";

service LiftService {
    rpc SetLiftHeight(SetLiftHeightRequest) returns (SetLiftHeightResponse);
    rpc GetCurrentLiftHeight(google.protobuf.Empty) returns
        (GetLiftHeightResponse);

    rpc SetEnabled(ModuleEnableRequest) returns (ModuleEnableResponse);
    rpc GetEnabled(google.protobuf.Empty) returns (ModuleEnableResponse);
}

message SetLiftHeightRequest {
    double scale = 1; // 比例值[0,1]
}

message SetLiftHeightResponse {

```

```

    StatusResponse status = 1;
}

message GetLiftHeightResponse {
    double current_scale = 1;
    google.protobuf.Timestamp timestamp = 2;
}

```

4.6 底盘服务控制 (base.proto)

protobuf

```

syntax = "proto3";

package robot.api.v1;

import "common.proto";

// 底盘控制服务（拆分实时/常规控制，新增导航和状态查询）
service BaseService {
    // 实时平移控制（持续速度，无距离限制，用于遥控操作等场景）
    rpc RealTimeTranslate(RealTimeTranslateRequest) returns (ControlResponse);

    // 实时旋转控制（持续速度，无角度限制，用于遥控操作等场景）
    rpc RealTimeRotate(RealTimeRotateRequest) returns (ControlResponse);

    // 常规平移控制（指定速度、方向和距离，到达后停止）
    rpc TranslateWithDistance(TranslateWithDistanceRequest) returns
    (ControlResponse);

    // 常规旋转控制（指定速度和目标角度，到达后停止）
    rpc RotateWithAngle(RotateWithAngleRequest) returns (ControlResponse);

    // 导航到目标位置（全局坐标系下的点，自动规划路径）
    rpc NavigateTo(NavigateRequest) returns (NavigateResponse);

    // 急停（立即停止所有运动）
    rpc Stop(google.protobuf.Empty) returns (StatusResponse);

    // 查询底盘状态（导航状态、位置、电量等）
    rpc GetBaseStatus(google.protobuf.Empty) returns (BaseStatusResponse);

    rpc SetEnabled(ModuleEnableRequest) returns (ModuleEnableResponse);
    rpc GetEnabled(google.protobuf.Empty) returns (ModuleEnableResponse);
}

// 实时平移请求（仅速度和方向，持续运动）
message RealTimeTranslateRequest {
    double speed_mps = 1;    // 平移速度（米/秒，>0 有效）
    double direction_deg = 2; // 方向角（度）：正前方为 0，向左为正（0~180），向右
    为负（-180~0）
}

```

```

// 实时旋转请求（仅速度，持续旋转）
message RealTimeRotateRequest {
    double speed_degps = 1;    // 旋转速度（度/秒，>0 有效）
    RotationDirection direction = 2; // 旋转方向
}

// 常规平移请求（带距离，到达后停止）
message TranslateWithDistanceRequest {
    double speed_mps = 1;      // 平移速度（米/秒，>0 有效）
    double direction_deg = 2;  // 方向角（同实时平移）
    double distance_m = 3;     // 移动距离（米，>0 有效）
}

// 常规旋转请求（带角度，到达后停止）
message RotateWithAngleRequest {
    double speed_degps = 1;    // 旋转速度（度/秒，>0 有效）
    RotationDirection direction = 2; // 旋转方向
    double angle_deg = 3;      // 旋转角度（度，>0 有效）
}

// 导航请求（全局坐标系目标点）
message NavigateRequest {
    string navi_target_name = 1; // 导航目的地
}

// 通用控制响应（实时/常规控制共用）
message ControlResponse {
    StatusResponse status = 1; // 控制指令执行状态（成功/参数错误等）
}

// 导航响应
message NavigateResponse {
    StatusResponse status = 1;
}

// 底盘状态响应（包含导航状态、位置、电量等）
message BaseStatusResponse {
    string navi_status = 1;    // 导航状态（如"IDLE"、"NAVIGATING"、"FAILED"）
    float x = 2;               // 当前 X 坐标（米，全局坐标系）
    float y = 3;               // 当前 Y 坐标（米，全局坐标系）
    float yaw = 4;             // 当前朝向角（度）
    int32 battery = 5;         // 电池电量（百分比，0-100）
    google.protobuf.Timestamp update_time = 6; // 状态更新时间
}

// 旋转方向枚举（语义更清晰，避免正负值混淆）
enum RotationDirection {
    ROTATION_DIRECTION_UNSPECIFIED = 0;
    LEFT = 1; // 向左旋转（逆时针）
    RIGHT = 2; // 向右旋转（顺时针）
}

```

4.7 相机服务（camera.proto）

protobuf

```
syntax = "proto3";

package robot.api.v1;

import "common.proto";

// 相机位置类型
enum CameraType {
    CAMERA_UNSPECIFIED = 0;
    CAMERA_HEAD = 1;
    CAMERA_CHEST = 2;
    CAMERA_LEFT_HAND = 3;
    CAMERA_RIGHT_HAND = 4;
}

// 2D 图像数据（仅含动态变化的核心数据）
message Image2D {
    bytes jpeg_data = 1;           // JPEG 二进制数据（动态变化）
    google.protobuf.Timestamp timestamp = 2; // 每帧采集时间戳（动态变化）
}

// RGBD 数据（2D+深度，尺寸通过内参获取）
message RGBDImage {
    Image2D image_2d = 1;          // 2D 图像（含时间戳）
    bytes depth_data = 2;          // 深度数据（H*W*float16 字节流，行优先；尺寸=内
    // 参的 base_width/base_height）
}

// 通用相机请求
message CameraRequest {
    CameraType camera_type = 1;
}

// 深度数据流控制请求
message DepthStreamControlRequest {
    CameraType camera_type = 1;
    bool enable = 2;               // true=开启，false=关闭
}

message CameraEnableRequest {
    CameraType camera_type = 1; // 目标相机（头部/胸部/左手/右手）
    bool enable = 2;            // true：使能，false：禁用（复用通用语义）
}

// 相机服务
service CameraService {
    // 流式读取 2D 数据（仅 JPEG+时间戳）
    rpc StreamGetImage2D(CameraRequest) returns (stream Image2D);
}
```

```

// 获取相机内参（含固定尺寸）
rpc GetCameraIntrinsics(CameraRequest) returns (CameraIntrinsicsResponse);

// 普通读取单帧 RGBD
rpc GetRGBDImage(CameraRequest) returns (RGBDImageResponse);

// 流式读取 RGBD
rpc StreamGetRGBDImage(CameraRequest) returns (stream RGBDImage);

// 控制深度流开关
rpc ControlDepthStream(DepthStreamControlRequest) returns
(ControlDepthStreamResponse);

rpc SetEnabled(CameraEnableRequest) returns (ModuleEnableResponse);
rpc GetEnabled(CameraRequest) returns (ModuleEnableResponse);
}

// 相机内参响应
message CameraIntrinsicsResponse {
    StatusResponse status = 1;
    CameraIntrinsics intrinsics = 2;
}

// 相机内参（含固定尺寸，一次获取后可缓存）
message CameraIntrinsics {
    double fx = 1;      // 焦距 x
    double fy = 2;      // 焦距 y
    double cx = 3;      // 主点 x
    double cy = 4;      // 主点 y
    repeated double distortion = 5; // 畸变系数（k1,k2,p1,p2,k3）
    int32 base_width = 6; // 固定图像宽度（像素，所有帧通用）
    int32 base_height = 7; // 固定图像高度（像素，所有帧通用）
}

// RGBD 数据响应
message RGBDImageResponse {
    StatusResponse status = 1;
    RGBDImage rgbd_image = 2;
}

// 深度流控制响应
message ControlDepthStreamResponse {
    StatusResponse status = 1;
    bool current_state = 2; // 当前开关状态
}

```