AimDK-A2

Version

目录

Agibot Raise A2-AimDK

Agibot Raise A2-AimDK provides secondary development interfaces for the Agibot Raise A2 series robots.

Getting Started

Secondary Development Boundary Guidelines [Important] [Mandatory Reading]

When performing secondary development based on the Raise A2 robot, please strictly adhere to the following boundaries:

1. As of version V1.2, the interfaces provided by AimDK do **not** support self-developed motion control or other low-level functionalities. Secondary development is limited to using AimDK's high-level motion control interfaces and other module interfaces. If you require custom motion control development for research purposes, please contact our business representatives for further discussion. **Note:** Replacing the default motion control program will disable all built-in motion-related functionalities of the robot. Please proceed with caution. Any damage or falls resulting from such modifications are **not covered under warranty** .
2. Deployment of secondary development programs on the x86 industrial control computer is **strictly prohibited** , as it may interfere with scheduling or resource allocation of critical programs (e.g., motion control) running on the x86 system. Secondary development programs must only be deployed on the ORIN development board or third-party development boards.
3. **Do not modify** the system environment on the ORIN board. When deploying secondary development programs, please use Python virtual environments or Docker containers.
4. **Do not arbitrarily modify** system configurations or built-in software algorithm settings, such as DDS configuration files, kernel settings, or internal algorithm parameters. If modifications are made during development and debugging (not recommended), the robot **must** be operated under protective conditions. Modified configurations **must not** be used in production environments. When reporting issues, please clearly disclose any modifications you have made.

Agibot will **not provide support or warranty coverage** for any software/hardware damage or API failures resulting from violations of the above boundaries.

AimDK Capability List

The Base and Flagship models support different capabilities. The specific AimDK capability support is as follows:

Motion Control Capabilities

Provides interfaces for robot body standing, motion, and other related functions.

Interface	Base	Flagship
Switch motion state	✔	✔
Control walking and turning	✔	✔
Position-controlled sitting and standing (specify chair or stand)	✔	✔
Get upper body status	✔	✔
Control upper body joints	✔	✔
Arm end-effector SE3 control	✔	✔
Waist motion control	✔	✔
Play specified full-body dance	✔	✔

System Capabilities

Provides interfaces for system status retrieval, fault alarms, health diagnostics, etc.

Interface	Base	Flagship
Get system status (e-stop, battery level, etc.)	✔	✔
Fault alarm	✔	✔
Health diagnostics	✔	✔

Expressive Interaction Capabilities

Provides interfaces related to human-robot interaction, including sound, motion, facial expressions, etc.

Interface	Base	Flagship
Expression invocation	✘	✔
Motion invocation	✔	✔
Text-to-Speech (TTS)	✘	✔
Audio file playback	✘	✔

Mapping and Navigation Capabilities

Provides interfaces for building and managing environmental maps, path planning, obstacle avoidance, etc.

Interface	Base	Flagship
Get map information	✘	✔
Target-point navigation	✘	✔
General endpoint navigation	✘	✔
Get navigation status	✘	✔
Robot localization (tf from map to base_link)	✘	✔

Perception Capabilities

Provides interfaces to obtain camera RGB data, LiDAR data, etc.

Interface	Base	Flagship
Get RGB data from left/right chest fisheye cameras	✘	✔
Get RGB and Depth data from head camera	✘	✔
Get RGB and Depth data from hip camera	✘	✔
Get LiDAR data (point cloud and IMU)	✘	✔

Interface	Base	Flagship
Get camera intrinsic and extrinsic parameters	❌	✔️
Get RGB data from center chest camera	❌	✔️ (P1 model only)

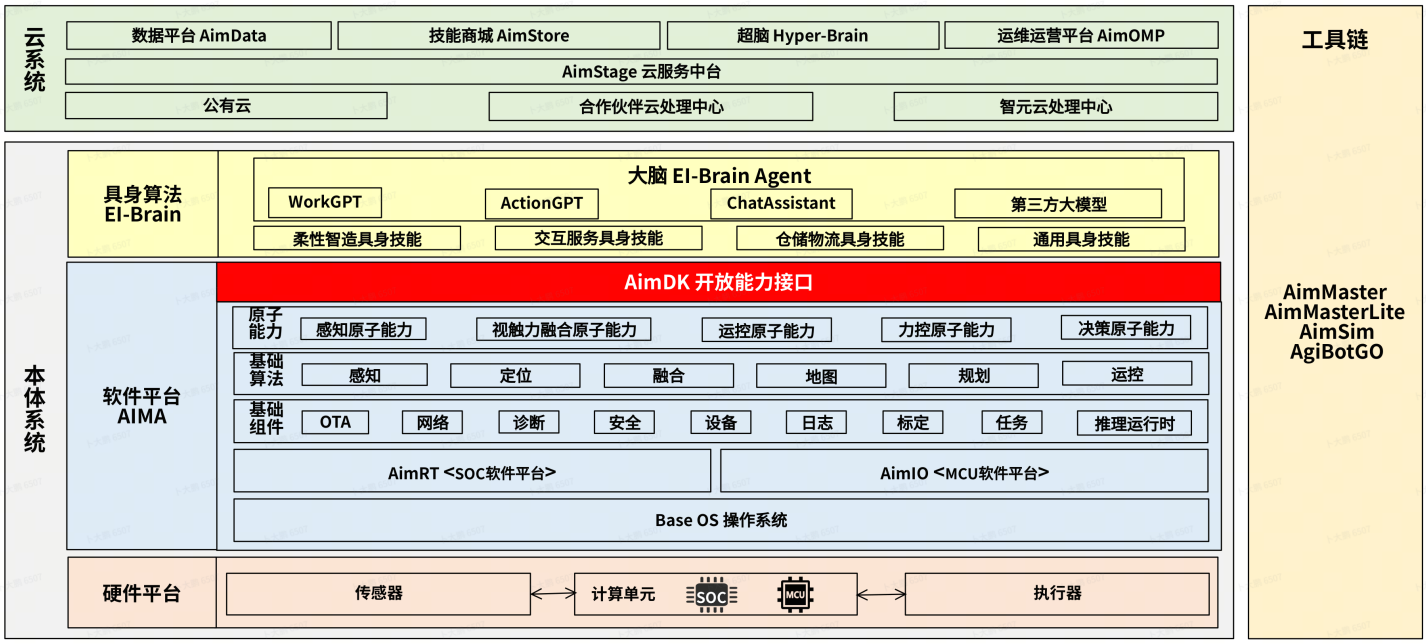
AimDK User Guide

The concept of AimDK differs from traditional SDKs. A traditional SDK provides developers with a software package containing dynamic libraries and header files, allowing developers to implement functionality by calling functions from these libraries. In contrast, AimDK adopts an approach better suited for robotics developers by offering interfaces to various modules on the robot. Developers implement functionality by calling RPC and Channel interfaces of these modules.

This page provides a detailed AimDK user guide to help you understand the concepts and usage methods within AimDK.

System Architecture

The onboard software system architecture is shown in the figure below:



As shown, AimDK sits on top of the robot software platform AIMA. Interfaces for all modules in AimDK are provided in the form of **documentation + protocol** . The documentation describes how to call each interface, while the protocol includes protobuf definitions and ROS2 definitions for these interfaces. Protocol files are located in the `protocol` directory of the A2 AimDK repository. In case of any inconsistency between the documentation and protocol files, the protocol files take precedence.

AimDK interfaces are essentially provided by various onboard software and algorithm modules, and come in two forms: RPC interfaces and Channel interfaces. RPC interfaces correspond to ROS2 Services, while Channel interfaces correspond to ROS2 Topics.

RPC Interfaces

RPC interfaces are typically high-level control interfaces, such as commanding the robot to navigate to a specific point or switching the motion control state machine. AimDK provides HTTP-based invocation methods (marked as HTTP backend in interface documentation). A common invocation method is shown below (using a command-line script as an example; the HTTP protocol can actually be called from any programming language):

```
curl -i     -H 'content-type:application/json' \
            -H 'timeout: 1000' \
            -X POST 'http://192.168.100.100:56421/rpc/aimdk.protocol.HalBmsService/GetBmsState' \
            -d '{}'
```

Where:

- `192.168.100.100` is the IP address of the robot's x86 development board. Use the IP address of the board where the corresponding service module is running. Refer to the Onboard Software Distribution section for details about which modules run on which boards.
- `56421` is the HTTP port number of the corresponding service. Refer to the Onboard Software Distribution section for HTTP port numbers of various modules.
- `content-type:application/json` is a required request header indicating that the request body is in JSON format.
- `timeout: 1000` is a request header specifying the request timeout in milliseconds (1000 ms = 1 second in this example). This field is optional.
- `aimdk.protocol.HalBmsService/GetBmsState` is the RPC interface name for retrieving battery state. Specific interface names can be found in the documentation and examples for each module.
- `-d '{}'` is the request body. An empty body here means all fields use their default values. To pass specific fields, provide a JSON string, e.g., `-d '{"bms_state": 1}'` passes the field `bms_state` with value 1 (for illustration only; refer to module documentation for actual field names and values).

Channel Interfaces

Channel interfaces are typically used for relatively high-frequency control and data/status retrieval, such as getting the robot's current position or motion state. AimDK provides native ROS2 invocation methods (marked as ROS2 backend in interface documentation). A common invocation method is shown below (using a Python script as an example; ROS2 interfaces can actually be developed in any ROS2-supported language):

```python
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from rclpy.qos import QoSHistoryPolicy, QoSProfile, QoSReliabilityPolicy
from sensor_msgs.msg import Image


class ImageSubscriber(Node):
    def __init__(self, topic_name: str):
        super().__init__("image_subscriber")

        self.topic_name = topic_name

        qos_profile = QoSProfile(
            history=QoSHistoryPolicy.KEEP_LAST, depth=10, reliability=QoSReliabilityPolicy.BEST_EFFORT
        )

        self.subscription = self.create_subscription(Image, topic_name, self.listener_callback,
qos_profile)
        self.subscription

    def listener_callback(self, msg: Image):
        self.get_logger().info(f"=== Received {self.topic_name} ===")
        self.get_logger().info(f"  header: {msg.header}")
        self.get_logger().info(f"  width: {msg.width}")
        self.get_logger().info(f"  height: {msg.height}")
        self.get_logger().info(f"  encoding: {msg.encoding}")
        self.get_logger().info(f"  is_bigendian: {msg.is_bigendian}")
        self.get_logger().info(f"  step: {msg.step}")
        self.get_logger().info(f"  data length: {len(msg.data)}")


def main(args=None):
```

```
    rclpy.init(args=args)
    # 头部相机 color: /aima/hal/rgbd_camera/head_front/color
    # 头部相机 depth: /aima/hal/rgbd_camera/head_front/depth
    # 髋部相机 color: /aima/hal/rgbd_camera/waist_front/color
    # 髋部相机 depth: /aima/hal/rgbd_camera/waist_front/depth
    # 胸部左侧相机 color: /aima/hal/fish_eye_camera/chest_left/color
    # 胸部右侧相机 color: /aima/hal/fish_eye_camera/chest_right/color
    image_node = ImageSubscriber("/aima/hal/fish_eye_camera/chest_right/color")
    try:
        rclpy.spin(image_node)
    except KeyboardInterrupt:
        pass
    finally:
        image_node.destroy_node()
        rclpy.shutdown()


if __name__ == "__main__":
    main()
```

Before running the above script on ORIN, set the following environment variables:

```
source /opt/ros/humble/setup.bash
export ROS_DOMAIN_ID=232
export ROS_LOCALHOST_ONLY=0
export FASTRTPS_DEFAULT_PROFILES_FILE=/agibot/software/v0/entry/bin/cfg/ros_dds_configuration.xml
```

Generally, do not modify the DDS configuration file above. Other configurations are untested and cannot guarantee proper communication between onboard modules.

All ROS2 topics onboard use the following default QoS settings:

```
  history: keep_last
  depth: 10
  reliability: best_effort
```

Please ensure your QoS settings match these when publishing or subscribing to topics to avoid communication failures due to QoS incompatibility.## Onboard Software Distribution

The current A2 robot flagship model includes two host development boards: one based on the x86_64 architecture and the other on the arm64 architecture. These are referred to below as the **x86** and **Orin** development boards, respectively. The base model only includes a single x86 development board. The distribution of modules across these boards is as follows:

Module	Abbreviation	Provided Interface Functions	Base Model Board	Flagship Model Board	HTTP Port
Motion Control	mc	Controls robot walking, upper-body motion, dexterous hand movement, head motion, and SE3 control of arm end-effectors	x86	x86	56322
Map Management	mm	Queries map-related information, typically used together with PnC (Planning & Control)	None	Orin	50807
Motion Player	motion_player	Plays pre-recorded motions (low-level interface)	x86	x86	56444
Sensor Data Interface	hal-sensor	Retrieves sensor data	None	Orin	56422
Health Diagnosis System	hds	Queries alarms and anomalies	x86	Orin	50587

Module	Abbreviation	Provided Interface Functions	Base Model Board	Flagship Model Board	HTTP Port
Remote Control	rc	Plays motions and facial expressions (high-level interface)	x86	x86	59001
Planning & Control	pnc	Provides planning, control, and navigation interfaces	None	Orin	53176
Voice Agent	agent	Provides microphone noise-suppressed audio stream interface	None	Orin	59301
Voice Interaction	interaction	Provides TTS playback and audio file playback interfaces	None	Orin	59201
Task Execution Engine	task_engine	Executes exhibition hall tour tasks and queries task status	None	Orin	57881
Other Interfaces	other	Queries battery level, emergency stop status, etc.	None	Orin	Varies by example

Both development boards on the A2 robot are equipped with debug ports located on the rear side of the robot's back. On the flagship model, when viewed from behind, the left debug port is for Orin and the right one is for x86. The Orin debug port has a fixed IP address of `192.168.2.50` , while the x86 debug port obtains its IP address via DHCP. The assigned IP address can be checked using the `ip addr` command. The base model only has an x86 debug port, which also uses DHCP for IP assignment.

On the A2 flagship model, the Orin and x86 development boards are connected via an Ethernet cable. They communicate using the IP addresses `192.168.100.100` (x86 board) and `192.168.100.110` (Orin board).

Secondary Development Program Deployment

1. **Deploying programs on the x86 development board is not allowed**
   - The x86 board runs motion control software. Deploying additional programs on it may cause abnormal behavior in the motion control software due to OS scheduling, CPU load, memory usage, etc., potentially leading to robot malfunctions or even falls.
2. **Simple secondary development programs**
   - Programs that invoke basic RPC interfaces for high-level control, such as those built on Agibot's software and algorithms for exhibition hall tours, voice interaction, etc.
   - It is recommended to deploy these on a third-party computer (e.g., a laptop) or industrial PC and call them via HTTP protocol.
3. **Complex secondary development programs**
   - Programs that require Channel interfaces for continuous data acquisition and robot control, such as embodied AI algorithms.
   - It is recommended to deploy these on the Orin development board or a third-party industrial PC, using ROS2 for development.
   - In this case, RPC interfaces are still called via HTTP, while Channel interfaces use ROS2.

Typical Scenario Calling Guide

Interfaces in AimDK are organized by module, which can sometimes make it difficult to locate a specific interface. Some functionalities require coordinated calls to multiple modules and may have prerequisites. This document provides step-by-step tables for several typical scenarios.

Scenario 1: Basic Robot Walking

No.	Step	Module	Interface Type	Interface Name	Notes
1	Switch motion control Action to a mode whose name contains "LOCOMOTION"	Motion Control Module	RPC	`pb:/aimdk.protocol.McActionService/SetAction`	See related documentation for Action state machine switching
2	Send lower-body control commands to the robot	Motion Control Module	Channel	`/motion/control/locomotion_velocity`	Continuously send commands at 20–100 Hz (including forward, lateral, and rotational velocities)

Refer to the example script `examples/mc/walk.py` .### Scenario 2: Upper Limb Motion Control of the Robot

The robot's upper limb control supports two modes: one uses the `PLANNING_MOVE` interface to send SE3 poses to the end-effector, and the other uses the `JOINT_SERVO` interface to send radian values directly to each joint of the upper limb (this command is passed through to the hardware abstraction layer for direct hardware control; the sent values must avoid abrupt jumps and should not deviate significantly from the current joint states).

End-effector SE3 Control:

No.	Step	Module	Interface Type	Interface Name	Notes
1	Switch the motion control Action to a mode suffixed with `PLANNING_MOVE`	Motion Control Module	RPC	`pb:/aimdk.protocol.McActionService/SetAction`	See relevant documentation for Action state machine transitions
2	Invoke the end-effector SE3 control command	Motion Control Module	RPC	`pb:/aimdk.protocol.McMotionService/PlanningMove`	

Refer to the example script `examples/mc/planning_move.py` .

Joint Radian Control:

No.	Step	Module	Interface Type	Interface Name	Notes
1	Stop `motion_player`	None	Execute on ORIN: `aima em stop-app motion_player`	Prerequisite	If `motion_player` is running and the motion control Action is in a state suffixed with `JOINT_SERVO` , `motion_player` will continuously send motion commands to the arm, causing interference or failure in upper limb control. In severe cases, this may lead to arm jitter or motor damage.
2	Switch the motion control Action to a	Motion Control Module	RPC	`pb:/aimdk.protocol.McActionService/SetAction`	See relevant documentation for Action state machine transitions

No.	Step	Module	Interface Type	Interface Name	Notes
	mode suffixed with `JOINT_SERVO`				
3	Invoke the pass-through arm joint control command	Motion Control Module	Channel	`/motion/control/arm_joint_command`	Continuously send joint radian values for both arms to achieve upper limb motion control, supporting a control frequency of 20–100 Hz.

Refer to the example script `examples/mc/arm.py` .### Scenario 3: Robot Head Control

Head control provides both a Channel interface (`/motion/control/neck_command`) and an RPC interface (`pb:/aimdk.protocol.McMotionService/SetNeckCommand`). Here, we use the head control RPC interface as an example to illustrate the head control process.

No.	Step	Module	Interface Type	Interface Name	Notes
1	Stop `motion_player`	None	Execute on ORIN: `aima em stop-app motion_player`	Prerequisite	If `motion_player` is running and the motion control state ends with `JOINT_SERVO` , `motion_player` will continuously send motion commands to the head, causing head motion control failure or interference. In severe cases, this may lead to head jitter or even motor damage.
2	Call robot head motion interface	Motion Control Module	RPC	`pb:/aimdk.protocol.McMotionService/SetNeckCommand`	Provide radian values for the two head joints (yaw and pitch). The Base model only has one controllable yaw joint.
3	Get robot head motion state (optional)	Motion Control Module	RPC	`pb:/aimdk.protocol.McDataService/GetNeckState`	

For the RPC interface, refer to the example `examples/mc/set_neck_command.py` . For the Channel interface, refer to the example `examples/mc/neck.py` .

Scenario 4: Robot Dexterous Hand Control
Both Channel and RPC interfaces are available.

No.	Step	Module	Interface Type	Interface Name	Notes
1	Stop `motion_player`	None	Execute on ORIN: `aima em stop-app motion_player`	Prerequisite	If `motion_player` is running and the motion control state ends with `JOINT_SERVO` , `motion_player` will continuously send motion commands to the dexterous hand, causing motion control failure or interference. In severe cases, this may lead to hand jitter or even motor damage.

No.	Step	Module	Interface Type	Interface Name	Notes
2	Call robot dexterous hand motion interface	Motion Control Module	RPC	`pb:/aimdk.protocol.McMotionService/SetHandCommand`	Provide radian values and torque values for the five joints of the dexterous hand. The control logic stops motion as soon as either the target radian or torque is reached.
3	Get robot dexterous hand motion state (optional)	Motion Control Module	RPC	`pb:/aimdk.protocol.McDataService/GetHandState`	

For the RPC interface, refer to the example `examples/mc/set_hand_command.py` . For the Channel interface, refer to the example `examples/mc/hand.py` .### Scenario Five: Robot Playing Pre-recorded Motions

Two modules provide interfaces for playing pre-recorded motions: the Motion Player Module and the Remote Control (RC) Module . The Motion Player Module offers low-level capabilities with more advanced controls such as pausing, while the RC Module wraps these capabilities into a simpler interface that only supports starting and resetting motions.

The following example uses the RC Module to illustrate the process of playing a pre-recorded motion.

No.	Step	Module	Type	Interface Name	Notes
1	Switch the motion control Action to a mode suffixed with `JOINT_SERVO`	Motion Control Module	RPC	`pb:/aimdk.protocol.McActionService/SetAction`	See the relevant documentation for Action state machine transitions
2	Get the motion list (optional; skip if the target motion ID is known)	RC Module	RPC	`pb:/aimdk.protocol.RcMotionPlayerService/GetMotionList`	Select a motion from the returned list and use its ID as input for the next step
3	Play the pre-recorded motion	RC Module	RPC	`pb:/aimdk.protocol.RcMotionPlayerService/PlayerMotion`	

Scenario Six: Robot Playing Predefined Emoticons

No.	Step	Module	Type	Interface Name	Notes
1	Get the emoticon list (optional; skip if the target emoticon ID is known)	RC Module	RPC	`pb:/aimdk.protocol.RcEmoticonPlayerService/GetEmotionList`	Select an emoticon from the returned list and use its ID as input for the next step
2	Play the emoticon	RC Module	RPC	`pb:/aimdk.protocol.RcEmoticonPlayerService/PlayerEmotion`	

Scenario Seven: Obtaining the Robot's Current Position

No.	Step	Module	Type	Interface Name	Notes
1	Successfully perform relocalization on AimMaster	None	None	Prerequisite	
2	Get the robot's current pose	tf module	Channel	`\tf`	Obtain the transform from `map` to `base_link` in `\tf` to get the robot's current pose

No.	Step	Module	Type	Interface Name	Notes
1	Perform mapping and mark navigation points on AimMaster	N/A	N/A	Prerequisite	
2	Successfully relocalize on AimMaster	N/A	N/A	Prerequisite	
3	Get current map ID (optional; skip if map ID is already known)	Map Management Module	RPC	`pb:/aimdk.protocol.MappingService/GetCurrentWorkingMap`	Returns the current map ID, which is used as input for steps 4 and 5
4	Get target point ID (optional; skip if target point ID is known)	Map Management Module	RPC	`pb:/aimdk.protocol.LocalizationService/GetTopoMsgs`	Returns a list of all target points; select one and use its ID as input for step 5
5	Set navigation goal and start navigation	Planning & Control Module	RPC	`pb:/aimdk.protocol.PncService/PlanningNaviToGoal`	
6	Get navigation status (optional)	Planning & Control Module	RPC	`pb:/aimdk.protocol.PncService/ActionGetState`	
7	Pause or cancel navigation task (optional)	Planning & Control Module	RPC	`pb:/aimdk.protocol.PncService/ActionPause` or `pb:/aimdk.protocol.PncService/ActionCancel`	

Refer to the example `examples/pnc/pnc_demo.py` .

Scenario 9: Call Robot TTS Interface to Play Voice

No.	Step	Module	Type	Interface Name	Notes
1	Call TTS interface to play voice	Interaction SDK Guide	RPC	`pb:/aimdk.protocol.TTSService/PlayTTS`	Fill `text` with the text to be spoken, `priority_level` with the priority, `domain` with the caller identifier, `trace_id` with the trace ID of the TTS task, and `is_interrupted` (usually `true`) to indicate whether to interrupt ongoing TTS tasks of the same priority
2	Get current TTS playback status (optional)	Interaction SDK Guide	RPC	`pb:/aimdk.protocol.TTSService/GetAudioStatus`	When the status changes from `Playing` to `NotInQue` , the TTS playback has finished

No.	Step	Module Containing the Interface	Interface Type	Interface Name	Notes
1	Upload the audio file to the directory `/agibot/data/var/interaction/audio/` on ORIN	None	None	Prerequisite	Audio files must be in PCM format, little-endian 16-bit, 16kHz, single-channel audio.

No.	Step	Module Containing the Interface	Interface Type	Interface Name	Notes
2	Call the audio file playback interface	Interaction Secondary Development Guide	RPC	`pb:/aimdk.protocol.TTSService/PlayMediaFile`	Fill in `file_name` with the audio filename uploaded in the previous step, `priority_level` with the priority level, `domain` with the caller identifier, `trace_id` with the trace ID of the playback task, and `is_interrupted` with whether to interrupt the current playback task of the same priority (usually `true`).
3	Get the current audio playback status (optional)	Interaction Secondary Development Guide	RPC	`pb:/aimdk.protocol.TTSService/GetAudioStatus`	When the status changes from `Playing` to `NotInQue` , the audio playback has finished.

Scenario 11: Obtaining Robot Sensor Data (LiDAR, Camera)

Sensor data is provided through a dedicated sensor data module. This module offers topic-based interfaces for sensor data such as LiDAR and camera, as well as an RPC interface for retrieving camera intrinsic parameters. These interfaces have no prerequisites or required interface combinations and can be called directly.

Refer to the examples: `examples/hal_sensor/camera.py` , `examples/hal_sensor/lidar.py` , and `examples/hal_sensor/get_intrinsics.py` .

Scenario 12: AimMaster Task Execution

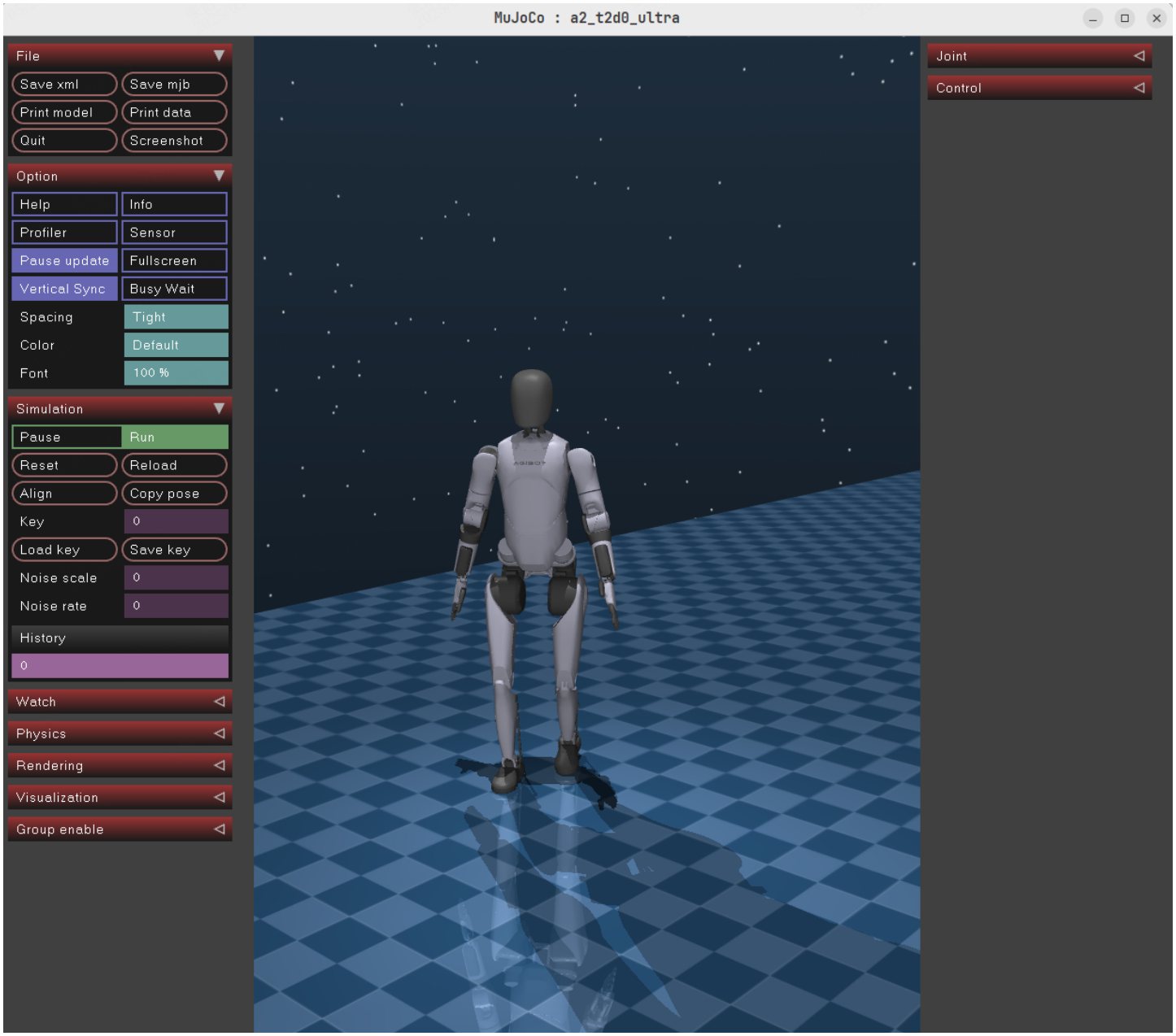
Tasks can be created and executed on AimMaster. Tasks created on AimMaster can also be launched and controlled via the Task Execution Engine module. The Task Execution Engine module provides interfaces for starting, controlling, and querying tasks. For detailed interface information, refer to the corresponding documentation .

No.	Step	Module Containing the Interface	Interface Type	Interface Name	Notes
1	Switch the robot to autonomous mode on AimMaster	None	None	Prerequisite	
2	Create a task on AimMaster	None	None	Prerequisite	
3	Call the Task Execution Engine to set the current task	Task Execution Engine Module	RPC	`pb:/aimdk.protocol.TaskEngineService/SetCurrentTask`	Requires the ID of the corresponding task.
4	Launch the task	Task Execution Engine Module	RPC	`pb:/aimdk.protocol.TaskEngineService/LaunchTask`	Requires the ID of the corresponding task.
5	Get task status (optional)	Task Execution Engine Module	RPC	`pb:/aimdk.protocol.TaskEngineService/GetTask`	This interface returns the execution status of the queried task, including the task ID, name, status, progress, and other information.

A2 Flagship Simulation User Guide

Simulation Description

This simulation is implemented based on MuJoCo Sim, primarily aiming to provide users with a low-cost method to test and verify the motion control module interfaces.



Notes:

1. The dexterous hand and head in the simulation use pure position control. When a step signal is sent, they immediately reach the target position without showing any motion process.
2. The simulation does not include sensors such as cameras or LiDAR; it only includes an IMU and joint position sensors.

Prerequisites

1. An x86_64 Linux system with Docker installed
2. Using the X11 desktop system
   - You can verify this by running `echo $XDG_SESSION_TYPE` ; the expected output is `x11` .

Pull the Docker Image

1. Log in to the read-only account of the image registry

```
docker login tongyong-public-cn-shanghai.cr.volces.com -u crrobot@aima-public-reader -p Aima123456
```

1. Pull the corresponding image

```
docker pull tongyong-public-cn-shanghai.cr.volces.com/aima-public/a2-simulator:v1.2
```

Launch the Docker Container

1. Allow Docker to access the display by configuring the X Window System access control list. Running `xhost +` permits all hosts to connect to the current user's X server. This disables access control for the X server, allowing any user to access and operate it.

```
xhost +
```

1. Run the Docker container

```
docker run -it \
--name=a2-sim \
--privileged \
--net=host \
--ipc=host \
--pid=host \
-e DISPLAY=$DISPLAY \
-v /tmp/.X11-unix:/tmp/.X11-unix \
-v /run/dbus/system_bus_socket:/run/dbus/system_bus_socket:ro \
-d tongyong-public-cn-shanghai.cr.volces.com/aima-public/a2-simulator:v1.2
```

If you observe lag in the simulation interface on machines with NVIDIA GPUs after starting the container using the above command, try launching the container with the following command instead:

```
docker run -it \
  --name=a2-sim \
  --gpus all \
  --privileged \
  --net=host \
  --ipc=host \
  --pid=host \
  -e DISPLAY=$DISPLAY \
  -e NVIDIA_VISIBLE_DEVICES=all \
  -e NVIDIA_DRIVER_CAPABILITIES=all \
  -v /tmp/.X11-unix:/tmp/.X11-unix \
  -v /run/dbus/system_bus_socket:/run/dbus/system_bus_socket:ro \
  -d tongyong-public-cn-shanghai.cr.volces.com/aima-public/a2-simulator:v1.2
```

1. Enter the container

```
docker start a2-sim && docker exec -it a2-sim /bin/zsh
```

Start the Simulation and Motion Control Program

1. After entering the container, run the following command to start the simulation:

```
cd /root/mujoco_sim/bin
./start_a2_t2d0_ultra.sh
```

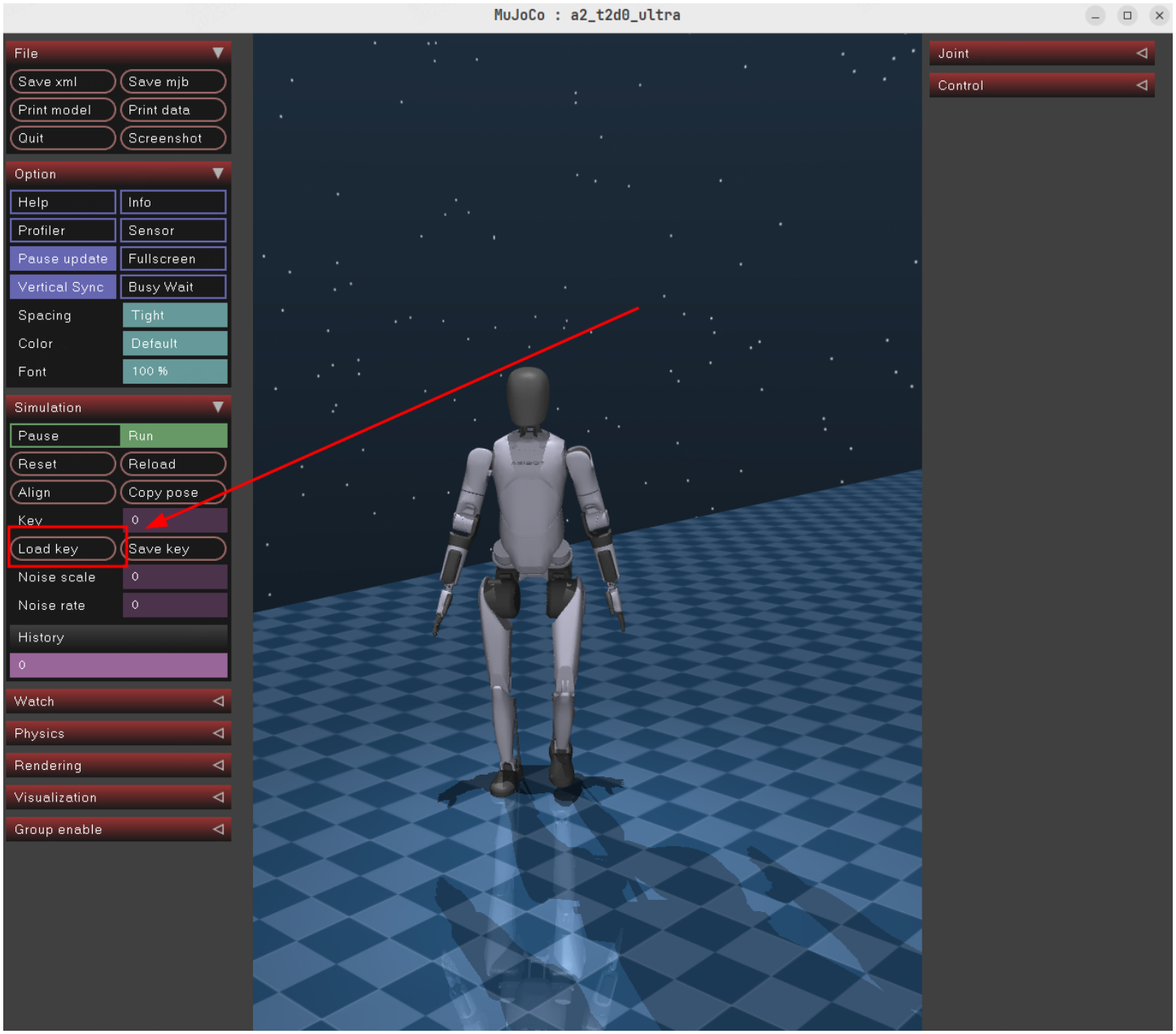
1. Start the motion control program

```
cd /root/motion_control/scripts/motion_control
./start_motion_control.sh
```

1. Run `SetAction.py` to switch the state machine to position-controlled standing mode

```
cd /root/tools
./SetAction.py # 选择 11 RL_JOINT_DEFAULT
```

1. Click **Load-Key** to stabilize the robot (if the robot is unstable, you may click multiple times)



1. Run `SetAction.py` again to switch the state machine to reinforcement learning-based force-controlled walking mode

```
./SetAction.py # 选择 16 RL_LOCOMOTION_DEFAULT
```

1. You can now normally call the motion control module interfaces. For example, you can invoke scripts under the `tools` directory to make the robot walk.

```
source /root/ros2_plugin_proto_install/share/ros2_plugin_proto/local_setup.zsh
./walk.py
```

Interaction Secondary Development Guide

Overview

This document details the recommended workflow and relevant API usage for interaction-related secondary development.

Categories of Interaction Secondary Development

There are two development modes available for client-side secondary development related to interaction:

1. **Agibot Interaction Simple Extension**
   - Retains Agibot's native interaction capabilities. Agibot provides basic APIs such as TTS, actions, and facial expressions. These APIs can be used to control the robot for basic exhibition hall tours, presentations, and similar workflows, while the robot continues to operate normally with Agibot's voice interaction solution.
2. **Full Takeover by Secondary Development**
   - Disables Agibot's native interaction capabilities. Agibot provides access to the robot's microphone input (with onboard noise reduction applied) and grants permission to use the speaker system. The secondary development program fully takes over all voice interaction functionalities.

Below is a detailed comparison table of the two modes:

Mode	Agibot Interaction Retained	APIs Provided by Agibot	Internet Requirement	Suitable Scenarios	Development Difficulty and Effort
Agibot Interaction Simple Extension	Yes	TTS, action, and expression APIs	Internet required for Agibot interaction and TTS APIs	Simple exhibition hall demo or fixed workflow	Relatively simple; effort depends on workflow complexity, generally low
Full Takeover by Secondary Development	No	Denoised audio input API, speaker system API	Internet required only during robot initialization; not needed afterward	Developing a complete custom interaction system	High difficulty; requires a full R&D team; significant effort

Agibot Interaction Simple Extension

TTS API Call Example

```
curl -i \
    -H 'content-type:application/json' \
    -X POST 'http://192.168.100.110:59201/rpc/aimdk.protocol.TTSService/PlayTTS' \
    -d '{"text":"测试文本","priority_level":"INTERACTION_L6","domain":"example",
"trace_id":"hafhjkqwjwefk", "is_interrupted":true}'
```

The meanings of each field are as follows:

Field Name	Type	Required	Description
header	RequestHeader	Yes	Common request header (includes ID and timestamp)
text	string	Yes	Text content to be synthesized
priority_level	PriorityLevel	Yes	Priority level
domain	string	Yes	Caller identifier
trace_id	string	Yes	Required if you need to track playback status; use this value to query status
is_interrupted	bool	Yes	Whether to interrupt playback of the same priority level; default to `true` ; set to `false` if queuing is needed

The priority levels are defined as follows (use `INTERACTION_L6` for standard voice interactions):

Enum Value	Value	Description
BACKGROUND_L1	1	Background service layer (lowest)
SERVICE_L2	2	Proactive service layer
MISSION_L4	4	Task execution layer
INTERACTION_L6	6	Interaction response layer (default)
WARNING_L8	8	Hazard warning layer
SAFETY_L10	10	Life safety layer (highest)

```
curl -i \
    -H 'content-type:application/json' \
    -X POST 'http://192.168.100.110:59201/rpc/aimdk.protocol.TTSService/PlayMediaFile' \
    -d '{"file_name": "custom/wake.pcm","priority_level":"INTERACTION_L6","domain":"example",
"trace_id":"hafhjkqwjwefk", "is_interrupted": true}'
```

The meaning of each field is as follows:

Field Name	Type	Required	Description
header	RequestHeader	No	Common request header
file_name	string	Yes	File name (stored under `/agibot/data/var/interaction/audio/` , e.g., `wake.pcm` . The file must be a 24kHz, 16-bit, mono PCM file. A separate folder should be created, and the value should be passed as `folder_name/file_name` , e.g., `file/wake.pcm`)
priority_level	PriorityLevel	Yes	Playback priority level
trace_id	string	Yes	If you need to retrieve the playback status, this field must be provided, and its value should be used as the parameter for querying the playback status
is_interrupted	bool	Yes	Whether to interrupt playback of the same priority level. Please set this to `true` by default. Set to `false` if you require queued playback behavior

There are two ways to obtain the playback status of TTS and audio files: querying via an RPC interface or subscribing to the announcement status topic. The former is simpler, while the latter is more accurate.

Below is an example of querying the announcement status via the RPC interface. You need to provide the `trace_id` that was passed when calling the TTS or audio playback interface:

```
curl -i \
    -H 'content-type:application/json' \
    -X POST 'http://192.168.100.110:59201/rpc/aimdk.protocol.TTSService/GetAudioStatus' \
    -d '{"trace_id":"hafhjkqwjwefk"}'
```

The response includes a `TTSStatus` field, described as follows:

Field Name	Type	Description
text	string	Text being announced
priority	uint32	Final priority coefficient
trace_id	string	Matches the `event_id` of the voice interaction; i.e., the `trace_id` passed when calling the interface
tts_status	TTSStatusType	Announcement status
domain	string	Identifier of the calling source
error_message	string	Error message (valid only in error states); generally unused and can be ignored

The `TTSStatusType` enumeration values are explained below:

Enum Value	Value	Description
TTSConfigStatusType_Unknown	0	Unknown status
TTSStatusType_Begin	1	Playback started
TTSStatusType_Playing	2	Currently announcing
TTSStatusType_End	3	Announcement finished
TTSStatusType_Stop	4	Announcement paused / canceled / interrupted
TTSStatusType_Error	5	Announcement failed
TTSStatusType_InQue	6	In the announcement queue, not started yet
TTSStatusType_NOTInQue	7	Text not found in the announcement queue and not being played

When using this RPC interface, the typical returned statuses are `InQue` , `Playing` , and `NOTInQue` . The `End` status (announcement finished) exists only briefly and is usually not captured. If you need to detect the end-of-playback status reliably, subscribe to the announcement status topic.

Example of subscribing to the announcement status topic:

```python
#!/usr/bin/env python3

import rclpy
from rclpy.node import Node
from rclpy.qos import QoSHistoryPolicy, QoSProfile, QoSReliabilityPolicy
from ros2_plugin_proto.msg import RosMsgWrapper

from aimdk.protocol_pb2 import TTSStatus

class TTSStatusSubscriber(Node):
```

```
    def __init__(self):
        super().__init__("tts_status_subscriber")

        # 音频缓冲区，按 stream_id 分别存储
        self.audio_buffers = {}  # {stream_id: bytearray()}
        self.recording_state = {}  # {stream_id: bool} 记录是否正在录音

        qos_profile = QoSProfile(
            history=QoSHistoryPolicy.KEEP_LAST,
            depth=10,
            reliability=QoSReliabilityPolicy.BEST_EFFORT,
        )

        self.subscription = self.create_subscription(
            RosMsgWrapper,
            "/interaction/tts_status/pb_3Aaimdk_2Eprotocol_2ETTSStatus",
            self.tts_status_callback,
            qos_profile,
        )

        self.get_logger().info("开始订阅 tts 播报状态。..")

    def tts_status_callback(self, msg):
        try:
            # 检查序列化类型是否为 pb
            if msg.serialization_type != "pb":
                self.get_logger().warn(f"不支持的序列化类型：{msg.serialization_type}")
                return

            # 将 data 字段从 list[bytes] 转换为 bytes
            tts_status_bytes = b"".join(msg.data)

            tts_status = TTSStatus()
            tts_status.ParseFromString(tts_status_bytes)

            self.get_logger().info(f"收到 TTS 播报状态：{tts_status}")

        except Exception as e:
            self.get_logger().error(f"处理 TTS 播报状态消息时出错：{e}")

def main(args=None):
    rclpy.init(args=args)

    tts_status_subscriber = TTSStatusSubscriber()

    try:
        tts_status_subscriber.get_logger().info(
            "正在监听 TTS 播报状态，按 Ctrl+C 退出。.."
        )
        rclpy.spin(tts_status_subscriber)
    except KeyboardInterrupt:
        tts_status_subscriber.get_logger().info("收到退出信号，正在关闭。..")
    finally:
        tts_status_subscriber.destroy_node()
        rclpy.shutdown()

if __name__ == "__main__":
    main()
```

```The above program depends on the Python package `a2_aimdk` and the ROS2 package `ros2_plugin_proto`. The corresponding packages have already been placed in the `prebuilt` directory of the AimDK development package. The Python package can be installed using `pip install prebuilt/a2_aimdk-1.0.0-py3-none-any.whl`, and the ROS2 package requires sourcing `source prebuilt/ros2_plugin_proto_aarch64/share/ros2_plugin_proto/local_setup.bash` before use.

Please note that the above program receives ROS2 messages and requires the following environment variables to be set:

```bash
source /opt/ros/humble/setup.bash
export ROS_DOMAIN_ID=232
export ROS_LOCALHOST_ONLY=0
export FASTRTPS_DEFAULT_PROFILES_FILE=/agibot/software/v0/entry/bin/cfg/ros_dds_configuration.xml
```
```

Broadcast Interruption API Call Examples
Additionally, both TTS and audio file playback can be interrupted. Examples are as follows:

```
# 需传入指定 trace_id 的播报任务
curl -i \
    -H 'content-type:application/json' \
    -X POST 'http://192.168.100.110:59201/rpc/aimdk.protocol.TTSService/StopTTSTraceId' \
    -d '{"trace_id":"hafhjkqwjwefk"}'
```

You can also terminate all TTS and audio file playback, including the current broadcast and all queued tasks:

```
curl -i \
    -H 'content-type:application/json' \
    -X POST 'http://192.168.100.110:59201/rpc/aimdk.protocol.TTSService/StopTTS' \
    -d "{}"
```

Motion API Call Examples

```
#!/bin/bash

if [ $# -eq 0 ]; then
    echo "arg error, need motion_id, ./send_motion_id.sh motion_id, example: "
    echo ./send_motion_id.sh /agibot/data/resources/default/motion/motion_player/default/演讲10s/演讲10s
    exit 0
fi

MC_ID="$1"

if [ "$MC_ID" == "停止动作" ]; then
    # 如果是“停止动作”，则 motion_id 为空，cmd_reset 设为 true
    DATA='{
      "motion_id": "",
      "duration_ms": 10000,
      "cmd_end": true,
      "cmd_pause": false,
      "cmd_reset": true
    }'
elif [ "$MC_ID" == "暂停播放器" ]; then
    # 如果是“暂停播放器”，则 motion_id 保持不变，cmd_pause 设为 true
    DATA='{
      "motion_id": "",
      "duration_ms": 10000,
      "cmd_end": true,
      "cmd_pause": true,
      "cmd_reset": false
    }'
elif [ "$MC_ID" == "下一个动作" ]; then
    # 如果是“下一个动作”，则 播放 list 中的下一个动作
    DATA='{
      "motion_id": "next_motion",
      "duration_ms": 10000,
      "cmd_end": true,
      "cmd_pause": false,
      "cmd_reset": false
    }'
else
    # 如果不是“停止动作”或“暂停播放器”，保持原有逻辑
    DATA='{
      "motion_id": "'"$MC_ID"'",
      "duration_ms": 10000,
      "cmd_end": true,
      "cmd_pause": false,
      "cmd_reset": false
    }'
fi

# 使用 curl 发送 POST 请求
```

```
curl -i \
    -H 'content-type:application/json' \
    -H 'timeout: 1000' \
    -X POST http://192.168.100.100:56444/rpc/aimdk.protocol.MotionCommandService/SendMotionCommand \
    -d "$DATA"

# 打印发送的数据
echo "$DATA"
```

An example of retrieving motion status is as follows:

```
curl -i \
    -H 'content-type:application/json' \
    -H 'timeout: 1000' \
    -X POST http://192.168.100.100:56444/rpc/aimdk.protocol.MotionCommandService/GetMotionStatus \
    -d '{}'
```

The `MotionCommandStatus` enumeration values in the returned motion status are explained below:

Enum Value	Value	Description
MotionCommandStatus_IDLE	0	Idle
MotionCommandStatus_START	1	Start Execution
MotionCommandStatus_OPERATING	2	Operating
MotionCommandStatus_PAUSE	3	Paused
MotionCommandStatus_STOP	4	Stopped

Expression API Call Examples

```
#!/bin/bash

if [ $# -eq 0 ]; then
    echo "arg error, need emoticon_id, ./a2_PlayerEmoticon.sh emoticon_id, example: "
    echo ./a2_PlayerEmoticon.sh 1
    exit 0
fi

curl -i \
    -H 'content-type:application/json' \
    -H 'timeout: 1000' \
    -X POST http://192.168.100.100:59001/rpc/aimdk.protocol.RcEmoticonPlayerService/PlayerEmoticon \
    -d '{"emoticon_id":"'$1'","is_need_data":false}'
```

The expression API does not support playback status queries.

Full Takeover for Secondary Development### Exiting the Agibot Interaction Pipeline

The Agibot interaction pipeline involves two modules: **agent** and **interaction** . In simple terms, the **agent** module is responsible for capturing audio, performing speech recognition, interacting with the cloud-based model, and then sending the results to the **interaction** module. The **interaction** module then executes corresponding actions such as voice playback, movements, facial expressions, etc., based on the received results.

In this mode, you need to exit the Agibot interaction pipeline so that the system only outputs microphone audio with onboard noise reduction applied and releases control of the speaker. This requires the following two operations:

1. Switch the **agent** module to `only_voice` mode by invoking the following RPC (a robot reboot is required for the change to take effect):

```
curl -i \
    -H 'content-type:application/json' \
    -X POST 'http://192.168.100.110:59301/rpc/aimdk.protocol.AgentControlService/
SetAgentPropertiesRequest' \
    -d '{ "contents": { "properties": { "2": "only_voice" } } }'
```

After calling this RPC, you must restart the robot for the change to take effect. You can wait until after completing the second step below before rebooting the robot. (To restore normal operation, simply change `"only_voice"` back to `"normal"` .)

2. Modify the configuration to prevent the **interaction** module from starting by default (this change only needs to be performed once):

   All the following operations should be executed on the ORIN device. First, back up the relevant configuration files:

```
cp /agibot/software/v0/entry/bin/cfg/run_agibot.yaml  /agibot/software/v0/entry/bin/cfg/
run_agibot.yaml.backup
cp /agibot/software/v0/entry/bin/cfg/sm.yaml /agibot/software/v0/entry/bin/cfg/sm.yaml.backup
```

   Then, in the `run_agibot.yaml` file, remove `interaction` from the `default_apps` list. In the `sm.yaml` file, remove `interaction` from the `Manager` entry under `function_groups` . Do not modify any other parts of these files.

3. Additionally, disable the monitoring process's supervision of the **interaction** module by editing `/agibot/software/v0/scripts/agent/process_monitor.sh` and removing the following section:

```
    exit 1
fi

PROCESS_LIST=(
    # <进程名>:<启动命令>:<工作目录（可选）>:<环境变量（可选）>
    "interaction:aima em start-app interaction"
)

LOG_FILE="./process_monitor.log"

sleep 5

check_and_restart() {
    echo "$(date +'%Y-%m-%d %H:%M:%S') 检查进程状态..." >> "$LOG_FILE"
```

   After making these changes, restart the robot. (To restore the original behavior, revert the modified files and restart the robot.)

Once the above steps are completed, the Agibot interaction pipeline will have been successfully exited.### Microphone Audio and Speaker Usage Instructions

Note: To obtain the audio described below, the robot must be connected to the internet for at least 2 minutes after power-on to complete audio-related authentication. Otherwise, no raw audio output will be available. For offline usage, ensure this interface produces audio output before disconnecting from the network.

Note: The speaker volume must not exceed 70%. Exceeding this volume level will cause the speaker to operate beyond its rated capacity after amplification, potentially resulting in speaker damage.

Below is an example program for obtaining noise-suppressed microphone audio from the robot:

```
#!/usr/bin/env python3

import rclpy
from rclpy.node import Node
from rclpy.qos import QoSHistoryPolicy, QoSProfile, QoSReliabilityPolicy
from ros2_plugin_proto.msg import RosMsgWrapper

from aimdk.protocol_pb2 import ProcessedAudioOutput, AudioVADState
```

```
import datetime
import os

class AudioSubscriber(Node):
    def __init__(self):
        super().__init__("audio_subscriber")

        # Audio buffers, stored separately by stream_id
        self.audio_buffers = {}  # {stream_id: bytearray()}
        self.recording_state = {}  # {stream_id: bool} indicating whether recording is active

        # Create directory for storing audio files
        self.audio_output_dir = "audio_recordings"
        os.makedirs(self.audio_output_dir, exist_ok=True)

        qos_profile = QoSProfile(
            history=QoSHistoryPolicy.KEEP_LAST,
            depth=10,
            reliability=QoSReliabilityPolicy.BEST_EFFORT,
        )

        self.subscription = self.create_subscription(
            RosMsgWrapper,
            "/agent/process_audio_output/pb_3Aaimdk_2Eprotocol_2EProcessedAudioOutput",
            self.audio_callback,
            qos_profile,
        )

        self.get_logger().info("Starting to subscribe to noise-suppressed audio data...")

    def audio_callback(self, msg):
        try:
            # Check if serialization type is 'pb'
            if msg.serialization_type != "pb":
                self.get_logger().warn(f"Unsupported serialization type: {msg.serialization_type}")
                return

            # Convert the data field from list[bytes] to bytes
            audio_data_bytes = b"".join(msg.data)

            # Parse the message using the generated protobuf class
            processed_audio = ProcessedAudioOutput()
            processed_audio.ParseFromString(audio_data_bytes)

            self.get_logger().info(
                f"Received audio data: stream_id={processed_audio.stream_id}, "
                f"vad_state={processed_audio.vad_state}, "
                f"audio_size={len(processed_audio.audio_data)} bytes"
            )

            # Handle audio based on VAD state
            self.handle_vad_state(processed_audio)

        except Exception as e:
            self.get_logger().error(f"Error processing audio message: {e}")

    def handle_vad_state(self, processed_audio):
        """Handle different VAD states"""
        vad_state = processed_audio.vad_state
        stream_id = processed_audio.stream_id
        audio_data = processed_audio.audio_data

        # Initialize buffer for this stream_id if not already present
        if stream_id not in self.audio_buffers:
            self.audio_buffers[stream_id] = bytearray()
            self.recording_state[stream_id] = False

        # VAD state name mapping
        vad_state_names = {
            AudioVADState.AUDIO_VAD_STATE_NONE: "No speech",
            AudioVADState.AUDIO_VAD_STATE_BEGIN: "Speech started",
            AudioVADState.AUDIO_VAD_STATE_PROCESSING: "Speech in progress",
```

```
            AudioVADState.AUDIO_VAD_STATE_END: "Speech ended",
        }

        stream_names = {1: "Built-in microphone", 2: "External microphone"}

        self.get_logger().info(
            f"[{stream_names.get(stream_id, f'Unknown stream {stream_id}')}] "
            f"VAD state: {vad_state_names.get(vad_state, f'Unknown state {vad_state}')} "
            f"Audio data: {len(audio_data)} bytes"
        )

        # Process audio data according to VAD state
        if vad_state == AudioVADState.AUDIO_VAD_STATE_BEGIN:
            self.get_logger().info("🎤 Speech detected - starting")
            # Start a new recording and clear the buffer
            self.audio_buffers[stream_id].clear()
            self.recording_state[stream_id] = True
            # Append current audio data
            if len(audio_data) > 0:
                self.audio_buffers[stream_id].extend(audio_data)

        elif vad_state == AudioVADState.AUDIO_VAD_STATE_PROCESSING:
            self.get_logger().info("🔄 Speech in progress...")
            # If recording, continue appending audio data to the buffer
            if self.recording_state[stream_id] and len(audio_data) > 0:
                self.audio_buffers[stream_id].extend(audio_data)

        elif vad_state == AudioVADState.AUDIO_VAD_STATE_END:
            self.get_logger().info("✅ Speech ended")
            # Append final audio data
```
```
            if self.recording_state[stream_id] and len(audio_data) > 0:
                self.audio_buffers[stream_id].extend(audio_data)

            # Save the complete audio segment
            if (
                self.recording_state[stream_id]
                and len(self.audio_buffers[stream_id]) > 0
            ):
                self.save_audio_segment(bytes(self.audio_buffers[stream_id]), stream_id)

            # End recording
            self.recording_state[stream_id] = False

        elif vad_state == AudioVADState.AUDIO_VAD_STATE_NONE:
            # No speech detected; do not record
            if self.recording_state[stream_id]:
                self.get_logger().info("⏹️ Recording state reset")
                self.recording_state[stream_id] = False

        # Log current buffer status
        if stream_id in self.audio_buffers:
            buffer_size = len(self.audio_buffers[stream_id])
            recording = self.recording_state[stream_id]
            self.get_logger().debug(
                f"[Stream {stream_id}] Buffer size: {buffer_size} bytes, Recording state: {recording}"
            )

    def save_audio_segment(self, audio_data, stream_id):
        """Save audio segment as 16kHz, 16-bit, mono PCM"""
        if len(audio_data) > 0:
            timestamp = datetime.datetime.now().strftime("%Y%m%d_%H%M%S_%f")

            # Create subdirectory by stream_id
            stream_dir = os.path.join(self.audio_output_dir, f"stream_{stream_id}")
            os.makedirs(stream_dir, exist_ok=True)

            # Generate filename
            stream_names = {1: "internal_mic", 2: "external_mic"}
            stream_name = stream_names.get(stream_id, f"stream_{stream_id}")
            filename = f"{stream_name}_{timestamp}.pcm"
            filepath = os.path.join(stream_dir, filename)

            try:
```

```
                with open(filepath, "wb") as f:
                    f.write(audio_data)
                self.get_logger().info(
                    f"Audio segment saved: {filepath} (size: {len(audio_data)} bytes)"
                )

                # Log audio duration (assuming 16kHz, 16-bit, mono)
                sample_rate = 16000
                bits_per_sample = 16
                channels = 1
                bytes_per_sample = bits_per_sample // 8
                total_samples = len(audio_data) // (bytes_per_sample * channels)
                duration_seconds = total_samples / sample_rate

                self.get_logger().info(
                    f"Audio duration: {duration_seconds:.2f} seconds ({total_samples} samples)"
                )

            except Exception as e:
                self.get_logger().error(f"Failed to save audio file: {e}")

    def get_buffer_info(self):
        """Get information about all buffers (for debugging)"""
        info = {}
        for stream_id in self.audio_buffers:
            info[stream_id] = {
                "buffer_size": len(self.audio_buffers[stream_id]),
                "recording": self.recording_state[stream_id],
            }
        return info

def main(args=None):
    rclpy.init(args=args)

    audio_subscriber = AudioSubscriber()

    try:
        audio_subscriber.get_logger().info("Listening to denoised audio data. Press Ctrl+C to exit...")
        rclpy.spin(audio_subscriber)
    except KeyboardInterrupt:
        audio_subscriber.get_logger().info("Exit signal received. Shutting down...")
    finally:
        audio_subscriber.destroy_node()
        rclpy.shutdown()

if __name__ == "__main__":
    main()
```

以上程序依赖 python 协议包 a2_aimdk 和 ros2 协议包 ros2_plugin_proto，相应的包已经放置在 AimDK 开发包的 `prebuilt` 目录下，python 包使用 `pip install prebuilt/a2_aimdk-1.0.0-py3-none-any.whl` 安装，ros2 包需要 `source prebuilt/ros2_plugin_proto_aarch64/share/ros2_plugin_proto/local_setup.bash` 后使用。

请注意，以上程序会接收 ros2 消息，需要设置如下环境变量：

```
source /opt/ros/humble/setup.bash
export ROS_DOMAIN_ID=232
export ROS_LOCALHOST_ONLY=0
export FASTRTPS_DEFAULT_PROFILES_FILE=/agibot/software/v0/entry/bin/cfg/ros_dds_configuration.xml
```

The audio data is 24kHz sample rate, 16-bit mono PCM data (little-endian). The output audio contains clean human speech that has already undergone noise suppression and echo cancellation, and can be directly used for ASR recognition.

The `ProcessedAudioOutput` message contains the following fields:

| Field Name | Type | Description | | ---------- | ---- | ----------- || header | Header | Common message header containing timestamp and message ID | | stream_id | uint32 | Audio stream ID (1: built-in microphone, 2: external microphone) | | vad_state | AudioVADState | Voice Activity Detection (VAD) state | | audio_data | bytes | Denoised PCM audio data |

AudioVADState enum definition:

Enum Value	Value	Description
AUDIO_VAD_STATE_NONE	0	No speech
AUDIO_VAD_STATE_BEGIN	1	Speech started
AUDIO_VAD_STATE_PROCESSING	2	Speech in progress
AUDIO_VAD_STATE_END	3	Speech ended

Speakers can directly use the operating system's API for audio playback. Taking `aplay` as an example:

```
aplay -D multiplay_def <file> # 2 声道下行播放，并回传 1 声道回采信号
```

Interaction Interface Summary

Example files for interaction-related interfaces are provided in both the `examples/agent` and `examples/interaction` directories within the AimDK package for reference.

Interface Name / Subscription Topic	Description	Request Message Type	Response Message Type	Notes	Bac
`/agent/process_audio_output`	Subscribe to denoised audio stream	`aimdk::protocol::ProcessedAudioOutput`	-	Audio data obtained via ROS2 subscription	ros2
`TTSService.PlayTTS`	Text-to-speech playback	`PlayTTSRequest`	`PlayTTSResponse`	Supports priority control and interruption policies	http
`TTSService.PlayMediaFile`	Play local audio file	`PlayMediaFileRequest`	`PlayTTSResponse`	Supports PCM/WAV format file playback	http
`TTSService.StopTTS`	Stop all TTS announcements	`CommonRequest`	`CommonResponse`	Terminates current and queued announcement tasks	http
`TTSService.StopTTSTraceId`	Stop TTS announcement for specified trace_id	`StopTTSTraceIdRequest`	`CommonResponse`	Terminates only the announcement task with the specified trace_id	http
`TTSService.GetAudioStatus`	Query TTS status	`GetTTSStatusRequest`	`GetTTSStatusResponse`	Retrieves current playback status and queue information	http

Communication Protocol Interfaces

AimDK Public Protocol

This page contains protocol definitions shared (common) across all modules.

Protobuf Message Types

Protobuf Message Types

`aimdk::protocol::JointState`
Joint information

Field	Type	Description
name	string	Joint name
sequence	uint32	Sequence number of the joint message
position	double	Joint position, unit: radians or meters
velocity	double	Joint velocity, unit: radians/second or m/s
effort	double	Joint effort (torque/force), unit: N or N·m

`aimdk::protocol::JointCommand`

Field	Type	Description
name	string	Joint name
sequence	uint32	Sequence number of the joint message
position	double	Joint position, unit: radians or meters
velocity	double	Joint velocity, unit: radians/second or m/s
effort	double	Joint effort (torque/force), unit: N
stiffness	double	Stiffness, unit: N·m/rad or N/m
damping	double	Damping, unit: N·s/m or N·m·s/rad

`aimdk::protocol::JointParam`

Field	Type	Description
name	string	Joint name
position_upper_limit	double	Upper position limit, unit: radians or meters
position_lower_limit	double	Lower position limit, unit: radians or meters
velocity_limit	double	Velocity limit, unit: radians/second or m/s

Field	Type	Description
acceleration_limit	double	Acceleration limit, unit: rad/s² or m/s²
effort_limit	double	Effort (torque/force) limit, unit: N or N·m

`aimdk::protocol::Rpy`

RPY (Roll-Pitch-Yaw) angular representation

Field	Type	Description
rx	double	Rotation angle around fixed X-axis, unit: rad
ry	double	Rotation angle around fixed Y-axis, unit: rad
rz	double	Rotation angle around fixed Z-axis, unit: rad

`aimdk::protocol::SE2Acceleration`

SE2 acceleration

Field	Type	Description
linear	aimdk::protocol::Vec2	Linear acceleration, unit: m/s²
angular	aimdk::protocol::Vec2	Angular acceleration, unit: rad/s²

`aimdk::protocol::Vec2`

2D vector

Field	Type	Description
x	double	
y	double	

`aimdk::protocol::PubImageData`

Image data format for streaming video frames to clients

Field	Type	Description
height	int32	Frame height
width	int32	Frame width
timestamp	aimdk::protocol::Timestamp	Timestamp
image_data	bytes	Image data

`aimdk::protocol::SE3Velocity`

SE3 velocity

Field	Type	Description
linear	aimdk::protocol::Vec3	Linear velocity, unit: m/s
angular	aimdk::protocol::Vec3	Angular velocity, unit: rad/s

`aimdk::protocol::Odometry`

Odometry

Field	Type	Description
timestamp	aimdk::protocol::Timestamp	
child_frame_id	string	
pose	aimdk::protocol::SE3Pose	
twist	aimdk::protocol::Twist	

Scalar trajectory point

Field	Type	Description
time_since_reference	double	Time since reference. Unit: s.
positions	double[]	The position of the trajectory. Unit: rad or m.
velocities	double[]	The velocity of the trajectory. Unit: rad/s or m/s.
accelerations	double[]	The acceleration of the trajectory. Unit: rad/s² or m/s².

`aimdk::protocol::ScalarTrajectory`

Scalar trajectory

Field	Type	Description
reference_time	double	All trajectories specify times relative to this reference time.
points	aimdk::protocol::ScalarTrajectoryPoint []	The trajectory points.

`aimdk::protocol::Box`

A box in a 2D space.

Field	Type	Description
x1	float	
y1	float	
x2	float	
y2	float	

`aimdk::protocol::Header`

Header for general messages

Field	Type	Description
seq	uint32	Sequence ID: consecutively increasing ID
timestamp	aimdk::protocol::Timestamp	Timestamp
frame_id	string	Frame this data is associated with
control_source	aimdk::protocol::ControlSource	Control source (algorithm modules can keep the default value and do not need to set it)
uuid	string	Device ID (used to identify AIMMASTER devices; usually no need to set)

`aimdk::protocol::RequestHeader`

Header for RPC request body

Field	Type	Description
timestamp	aimdk::protocol::Timestamp	Timestamp
control_source	aimdk::protocol::ControlSource	Control source (algorithm modules can keep the default value and do not need to set it)
uuid	string	Device ID (used to identify AIMMASTER devices; usually no need to set)
trace_id	string	RPC requester ID
domin	string	Requester's source domain

Header of the RPC response body.

Field	Type	Description
code	uint64	Result code: 0 indicates success; non-zero indicates failure.
msg	string	Description of the result
timestamp	aimdk::protocol::Timestamp	Timestamp
trace_id	string	ID of the RPC requester
domin	string	Source domain of the requester

`aimdk::protocol::BlockableRequestHeader`

Header of a blockable RPC request body.

Field	Type	Description
timestamp	aimdk::protocol::Timestamp	Timestamp
blocked	bool	Whether the call is blocking (default: false)
control_source	aimdk::protocol::ControlSource	

Field	Type	Description
		Control source (algorithm modules can generally keep the default and need not set this)
uuid	string	Device ID (used to identify AIMMASTER devices; usually no need to set)
trace_id	string	ID of the RPC requester
domin	string	Source domain of the requester

`aimdk::protocol::BlockableResponseHeader`

Field	Type	Description
code	uint64	Result code: 0 indicates success; non-zero indicates failure.
msg	string	Description of the result
timestamp	aimdk::protocol::Timestamp	Timestamp
id	uint64	RPC ID associated with the request. If non-blocking mode is enabled, this field can be used by the caller to periodically query the executor about the RPC execution status. This value must be maintained and set by the executor; default is zero (invalid).
trace_id	string	ID of the RPC requester
domin	string	Source domain of the requester

`aimdk::protocol::Wrench`

Torque and force.

Field	Type	Description
force	aimdk::protocol::Vec3	Force, unit: N
torque	aimdk::protocol::Vec3	Torque, unit: N·m

Wrench with timestamp

Field	Type	Description
timestamp	aimdk::protocol::Timestamp	
wrench	aimdk::protocol::Wrench	

`aimdk::protocol::SE3Pose`

SE3 pose (represented by quaternion orientation)

Field	Type	Description
position	aimdk::protocol::Vec3	3D position, unit: meters
orientation	aimdk::protocol::Quaternion	Quaternion

`aimdk::protocol::SE3RpyPose`
SE3 pose (represented by RPY angles)

Field	Type	Description
position	aimdk::protocol::Vec3	3D position, unit: meters
rpy	aimdk::protocol::Rpy	RPY angles

`aimdk::protocol::SE3Acceleration`
SE3 acceleration

Field	Type	Description
linear	aimdk::protocol::Vec3	Linear acceleration, unit: m/s^2
angular	aimdk::protocol::Vec3	Angular acceleration, unit: rad/s^2

`aimdk::protocol::CommonState`
State definitions

Name	Number	Description
CommonState_UNKNOWN	0	Unknown
CommonState_SUCCESS	1	Success
CommonState_FAILURE	2	Failure
CommonState_ABORTED	3	Aborted
CommonState_TIMEOUT	4	Timeout
CommonState_INVALID	5	Invalid
CommonState_IN_MANUAL	6	Manual
CommonState_NOT_READY	100	Not ready
CommonState_PENDING	200	Pending
CommonState_CREATED	300	Created
CommonState_RUNNING	400	Running

`aimdk::protocol::CommonRequest`
Common request type

Field	Type	Description
header	aimdk::protocol::RequestHeader	

`aimdk::protocol::CommonResponse`
Common response type

Field	Type	Description
header	aimdk::protocol::ResponseHeader	
state	aimdk::protocol::CommonState	

`aimdk::protocol::CommonTaskRequest`

Common Task request type

Field	Type	Description
header	aimdk::protocol::RequestHeader	
task_id	uint64	

`aimdk::protocol::CommonTaskResponse`

Common Task response type

Field	Type	Description
header	aimdk::protocol::ResponseHeader	
task_id	uint64	
state	aimdk::protocol::CommonState	

SE2 velocity

Field	Type	Description
linear	aimdk::protocol::Vec2	Linear velocity, in m/s
angular	aimdk::protocol::Vec2	Angular velocity, in rad/s

`aimdk::protocol::IoPacket`

Generic binary IO packet

Field	Type	Description
timestamp	aimdk::protocol::Timestamp	Timestamp
io_data	bytes	Image data

`aimdk::protocol::IoPackets`

Generic binary IO packet queue

Field	Type	Description
timestamp	aimdk::protocol::Timestamp	Timestamp
data_queue	bytes[]	Image data

`aimdk::protocol::Vec3`

3D vector

Field	Type	Description
x	double	
y	double	
z	double	

`aimdk::protocol::Timestamp`

Timestamp

Field	Type	Description
seconds	int64	Represents seconds of UTC time since Unix epoch 1970-01-01T00:00:00Z. Must be from 0001-01-01T00:00:00Z to 9999-12-31T23:59:59Z inclusive.
nanos	int32	Non-negative fractions of a second at nanosecond resolution. Negative second values with fractions must still have non-negative nanos values that count forward in time. Must be from 0 to 999,999,999 inclusive.
ms_since_epoch	int64	Milliseconds since epoch for counts used by frontend.

`aimdk::protocol::SE3TrajectoryPoint`

SE3 trajectory point

Field	Type	Description
time_since_reference	double	Time since reference. Unit: s.
pose	aimdk::protocol::SE3Pose	The SE3 pose of the trajectory.
velocity	aimdk::protocol::SE3Velocity	The SE3 velocity of the trajectory.
acceleration	aimdk::protocol::SE3Acceleration	The SE3 acceleration of the trajectory.

SE3 trajectory

Field	Type	Description
reference_time	double	Time since reference. Unit: s.
points	aimdk::protocol::SE3TrajectoryPoint []	The SE3 trajectory points.

`aimdk::protocol::QRPose`

QR pose

Field	Type	Description
qr_code	int32	QR code value
pose	aimdk::protocol::SE3Pose	6D pose

`aimdk::protocol::ControlSource`

Control source, set by the safety module to the MC, used by the MC to filter out mismatched control source inputs.

Name	Number	Description
ControlSource_AUTO	0	Default value, usually initiated by autonomous algorithms
ControlSource_MANUAL	1	Manual control, initiated from remote control module
ControlSource_SAFE	2	Safety module, initiated from safety module
ControlSource_DISABLED	255	Invalid control source, default value for MC

`aimdk::protocol::SE2Pose`

SE2 pose

Field	Type	Description
position	aimdk::protocol::Vec2	2D position, unit: meters
angle	double	Angle, unit: radians

`aimdk::protocol::Pixel`

2D pixel coordinates

Field	Type	Description
u	int32	Pixel x-coordinate
v	int32	Pixel y-coordinate

`aimdk::protocol::PixelPose`

2D pixel pose

Coordinate system illustration:

±-------> u

| - (angle)

| -

| *

v V

Field	Type	Description
position	aimdk::protocol::Pixel	2D pixel coordinates
angle	double	Angle from u-axis toward v-axis, unit: radians

`aimdk::protocol::Quaternion`

Quaternion

Field	Type	Description
x	double	
y	double	
z	double	
w	double	

`aimdk::protocol::Twist`
Velocity

Field	Type	Description
linear	aimdk::protocol::Vec3	Linear velocity
angular	aimdk::protocol::Vec3	Angular velocity

`aimdk::protocol::TwistStamped`
Velocity with timestamp

Field	Type	Description
timestamp	aimdk::protocol::Timestamp	Timestamp
twist	aimdk::protocol::Twist	Velocity

Field	Type	Description
name	string	
id	uint32	Motion capture tracker ID
status	uint32	Motion capture status
err_code	uint32	Motion capture error code
position	aimdk::protocol::Vec3	
orientation	aimdk::protocol::Quaternion	

Map Management Module

[This module is only available for Flagship models; Base model users do not need to read this section.]

The Map Management (MM) module handles robot map management, including map retrieval, storage, updates, and other related functionalities.

This module resides on the Orin development board, with its HTTP backend listening on port 50807.

RPC Interfaces

Interface Name	Description	Request Message Type	Response Message Type
`pb:/aimdk.protocol.MappingService/Get2DWholeMap`	Retrieve 2D map data	aimdk::protocol::Get2DWholeMapReq	aimdk::protocol::Get2DWholeMapRsp
`pb:/aimdk.protocol.MappingService/GetStoredMapNames`	Retrieve list of stored maps	aimdk::protocol::GetStoredMapNamesReq	aimdk::protocol::GetStoredMapNamesRs
`pb:/aimdk.protocol.MappingService/GetCurrentWorkingMap`	Get current working map ID	aimdk::protocol::GetCurrentWorkingMapReq	aimdk::protocol::GetCurrentWorkingMap
`pb:/aimdk.protocol.LocalizationService/GetTopoMsgs`	Retrieve all topological data for a given map	aimdk::protocol::GetTopoMsgsReq	aimdk::protocol::GetTopoMsgsRsp

Protobuf Message Types

`aimdk::protocol::PubCurrentPose`

Pushes real-time pose data to the client.

Field	Type	Description
timestamp	aimdk::protocol::Timestamp	Timestamp of message transmission
pose	aimdk::protocol::SE3Pose	Current pose data
position	aimdk::protocol::PixelPose	Current pixel coordinate data

Field	Type	Description
map_id	uint64	Map ID associated with current localization

`aimdk::protocol::Get2DWholeMapReq`
Request to retrieve 2D map data.

Field	Type	Description
header	aimdk::protocol::RequestHeader	Request header
command	aimdk::protocol::MappingCommand	Request command; value must be `MappingCommand_GET_2D_WHOLE_MAP`
map_id	string	Map ID

`aimdk::protocol::Get2DWholeMapRsp`
Response to retrieve 2D map data.

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Response header
data	aimdk::protocol::Get2DWholeMapData	Map data

`aimdk::protocol::Get2DWholeMapData`
2D map data.

Field	Type	Description
map_id	string	Map ID
map_name	string	Map name
map_width	int32	Map width
map_height	int32	Map height
map_resolution	float	Map resolution
origin_x	float	Map origin X coordinate
origin_y	float	Map origin Y coordinate
map_data	bytes	Map data (PNG binary)
rotate_angle	float	Map rotation angle

`aimdk::protocol::GetStoredMapNamesReq`
Request to retrieve the list of stored maps.

Field	Type	Description
header	aimdk::protocol::RequestHeader	Request header

Field	Type	Description
command	aimdk::protocol::MappingCommand	Request command; value must be `MappingCommand_GET_STORED_MAP_NAME`

`aimdk::protocol::GetStoredMapNamesRsp`
Response to retrieve the list of stored maps.

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Response header
data	aimdk::protocol::GetStoredMapNamesData	Map list

`aimdk::protocol::GetStoredMapNamesData`
Data containing the list of stored maps.

Field	Type	Description
map_lists	aimdk::protocol::MapIdNameCombine []	List of maps

`aimdk::protocol::MapIdNameCombine`
Combination of map ID and name.

Field	Type	Description
map_id	uint64	Map ID
map_name	string	Map name

Field	Type	Description
header	aimdk::protocol::RequestHeader	Request header
command	aimdk::protocol::MappingCommand	Request command, value is `MappingCommand_GET_CURRENT_WORKING_MAP`

`aimdk::protocol::GetCurrentWorkingMapData`

Field	Type	Description
map_id	uint64	Current map ID

`aimdk::protocol::GetCurrentWorkingMapRsp`

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Response header
data	aimdk::protocol::GetCurrentWorkingMapData	Response data

`aimdk::protocol::GetTopoMsgsReq`

Field	Type	Description
header	aimdk::protocol::RequestHeader	Request header
command	aimdk::protocol::TopoCommand	Request command, value is `TopoCommand_GET_TOPO_MSG`
map_id	uint64	Map ID

`aimdk::protocol::GetTopoMsgsData`

Field	Type	Description
points	aimdk::protocol::PixelNaviPointDataType []	Pixel navigation point topology information
paths	aimdk::protocol::PixelPathDataType []	Pixel path topology information
regions	aimdk::protocol::RegionReqDataType []	Region information

`aimdk::protocol::GetTopoMsgsRsp`

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Response header
data	aimdk::protocol::GetTopoMsgsData	Response data

`aimdk::protocol::RegionType`

Name	Number	Description
RegionType_UNDEFINED	0	Undefined type
RegionType_WORKSPACE	1	Workspace
RegionType_WALL	2	Virtual wall

`aimdk::protocol::RegionDrawingType`

Name	Number	Description
RegionDrawingType_UNDEFINED	0	Undefined type
RegionDrawingType_CLOSURE	1	Closed polygon
RegionDrawingType_LINES	2	Line segments

`aimdk::protocol::RegionReqDataType`

Field	Type	Description
name	string	Region name
type	aimdk::protocol::RegionType	Region type

Field	Type	Description
drawing_type	aimdk::protocol::RegionDrawingType	Region drawing type
region_id	int32	Region ID
vertices	aimdk::protocol::Pixel []	Region vertices

Field	Type	Description
name	string	Path name
path_id	int32	Path point ID
poses	aimdk::protocol::PixelPose []	Collection of path points
is_recording	bool	Flag indicating whether it is being recorded

`aimdk::protocol::PixelNaviPointDataType`

Field	Type	Description
point_id	int32	Navigation point ID
name	string	Navigation point name
point_type	aimdk::protocol::NaviPointType	Navigation point type
pixel_pose	aimdk::protocol::PixelPose	Pixel coordinates of the navigation point

Usage Examples

The following are HTTP cURL examples. Replace `127.0.0.1` with the IP address of the robot's Orin development board, and replace `map_id` with the ID of the desired map:

Get 2D map data

```
curl --location --request POST 'http://127.0.0.1:50807/rpc/aimdk.protocol.MappingService/Get2DWholeMap'
\
--header 'Content-Type: application/json' \
--data-raw '{"header":{},"command":"MappingCommand_GET_2D_WHOLE_MAP","map_id":"1732275337238"}'
```

Get the list of stored maps

```
curl --location --request POST 'http://127.0.0.1:50807/rpc/aimdk.protocol.MappingService/
GetStoredMapNames' \
--header 'Content-Type: application/json' \
--data-raw '{"header":{},"command":"MappingCommand_GET_STORED_MAP_NAME"}'
```

Get the current working map ID

```
curl --location --request POST 'http://127.0.0.1:50807/rpc/aimdk.protocol.MappingService/
GetCurrentWorkingMap' \
--header 'Content-Type: application/json' \
--data-raw '{"header":{},"command":"MappingCommand_GET_CURRENT_WORKING_MAP"}'
```

Get the topology data of the current map

```
curl --location --request POST 'http://127.0.0.1:50807/rpc/aimdk.protocol.LocalizationService/
GetTopoMsgs' \
--header 'Content-Type: application/json' \
--data-raw '{"header":{},"command":"TopoCommand_GET_TOPO_MSG","map_id":"1732275337238"}'
```

Motion Control Module

[The current motion control supports two modes: the traditional bent-leg walking mode and the reinforcement learning-based straight-leg walking mode. The traditional mode is not recommended for use. Starting from version V1.0, interfaces for the traditional mode will no longer be maintained and the traditional walking mode will be completely removed. Please use the reinforcement learning mode and its corresponding interfaces.]

This document describes the topics and services provided by the Motion Control (MC) module to support the system's motion control functionality.

This module runs on the x86 development board for both the Flagship and Base models, with the HTTP backend listening on port 56322.

Important Notes [Mandatory Reading]

1. Under actions with the `JOINT_SERVO` suffix—including `RL_LOCOMOTION_ARM_EXT_JOINT_SERVO` and `RL_WHOLE_BODY_EXT_JOINT_SERVO` modes—the arms, head, fingers, and waist are by default controlled by the Motion Player module and will not execute user commands.
    - You can dynamically stop the MotionPlayer service by calling the `DisableMotionPlayer` interface of the Motion Player module. Using this interface does not require restarting the `motion_player` service.

    ```
    curl -i     -H 'content-type:application/json' \
                -H 'timeout: 1000' \
                -X POST 'http://192.168.100.100:56444/rpc/aimdk.protocol.MotionCommandService/
    DisableMotionPlayer' \
                -d '{}'
    ```

2. When using joint pass-through interfaces, strictly adhere to the following requirements:
    - Input joint angle values must comply with joint limits and dynamic range constraints. The corresponding tables are listed below. For arm, head, and finger joints, users must ensure that the input joint angles remain within their respective limits.
    - A control frequency of 100 Hz is recommended for pass-through command interfaces. Sudden jumps, step changes, or large deviations between the input command and the current joint state are not allowed. Users must carefully compute joint velocities and accelerations to ensure smooth motion.

Joint limits and dynamic range constraints for upper-body joint control:

Joint Name	Min Angle (rad)	Max Angle (rad)	Max Speed (rad/s)	Max Acceleration (rad/s²)
Left Arm J1	-2.91	2.91	3.00	6.28
Left Arm J2	-0.46	1.60	3.00	6.28
Left Arm J3	-2.91	2.91	3.00	6.28
Left Arm J4	-2.00	-0.03	3.00	6.28
Left Arm J5	-2.94	2.94	3.00	6.28

Joint Name	Min Angle (rad)	Max Angle (rad)	Max Speed (rad/s)	Max Acceleration (rad/s²)
Left Arm J6	-0.45	0.45	3.00	6.28
Left Arm J7	-0.35	0.35	3.00	6.28
Right Arm J1	-2.91	2.91	3.00	6.28
Right Arm J2	-1.60	0.46	3.00	6.28
Right Arm J3	-2.91	2.91	3.00	6.28
Right Arm J4	0.03	2.00	3.00	6.28
Right Arm J5	-2.94	2.94	3.00	6.28
Right Arm J6	-0.45	0.45	3.00	6.28
Right Arm J7	-0.35	0.35	3.00	6.28

Head joint limits (position control only):

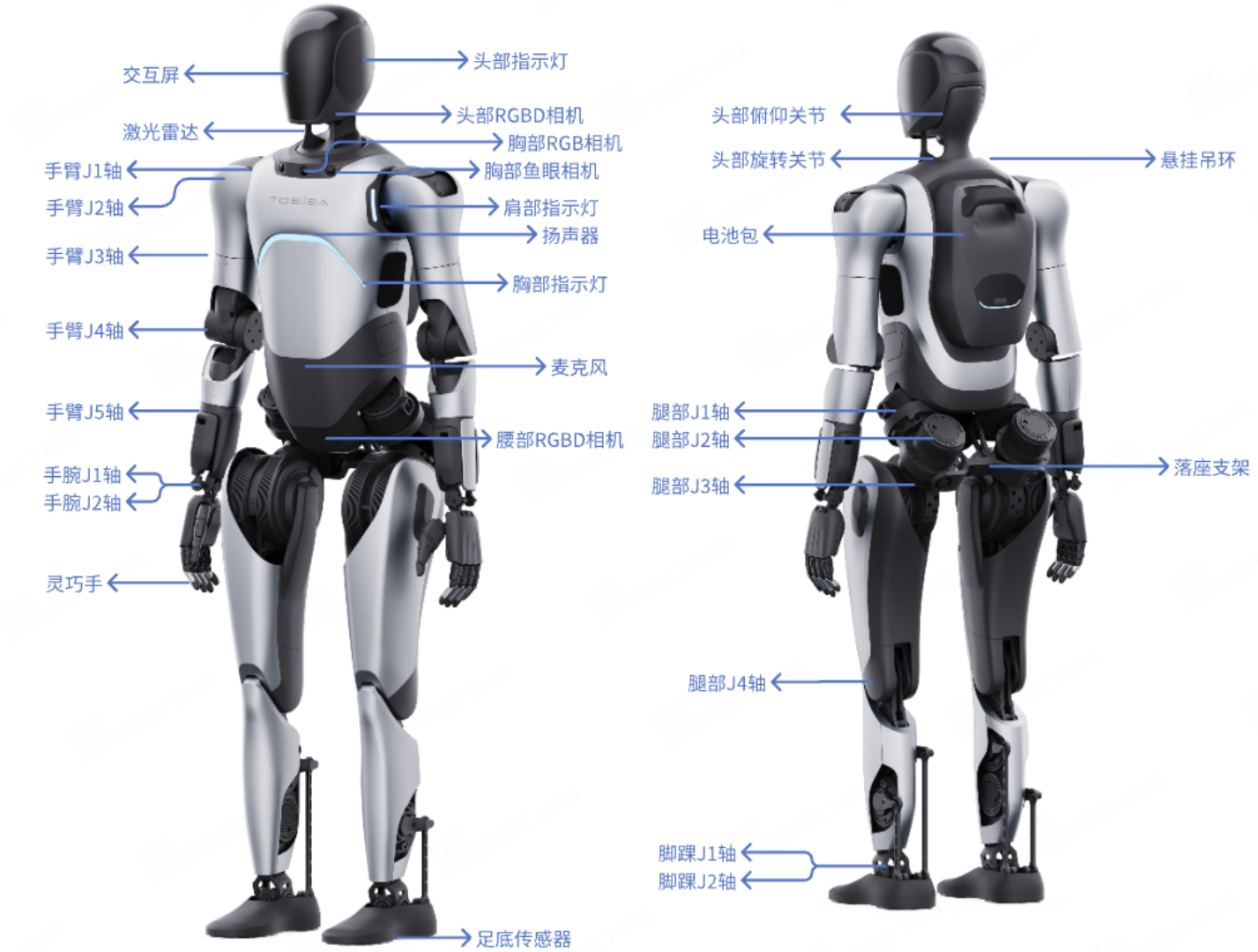
Joint Name	Min Angle (rad)	Max Angle (rad)
idx27_head_joint1 (摇头 / yaw)	-0.785	0.785
idx28_head_joint2 (点头 / pitch)	-0.401	0.401

Dexterous hand joint limits (support both position and torque control; limits are identical for left and right hands):

Joint Name	Min Position	Max Position	Min Torque Value	Max Torque Value
left_thumb_0	0	2000	0	5700
left_thumb_1	0	2000	0	5700
left_index	0	2000	0	5700
left_middle	0	2000	0	5700
left_ring	0	2000	0	5700
left_pinky	0	2000	0	5700

Waist motion limits:

Degree of Freedom	Min Angle/Position (rad / m)	Max Angle/Position (rad / m)
lift	-0.15	0.00
roll	-0.5236	0.5236
pitch	-0.5236	0.5236
yaw	-0.5236	0.5236



Motion Control State Machine 【MUST READ】

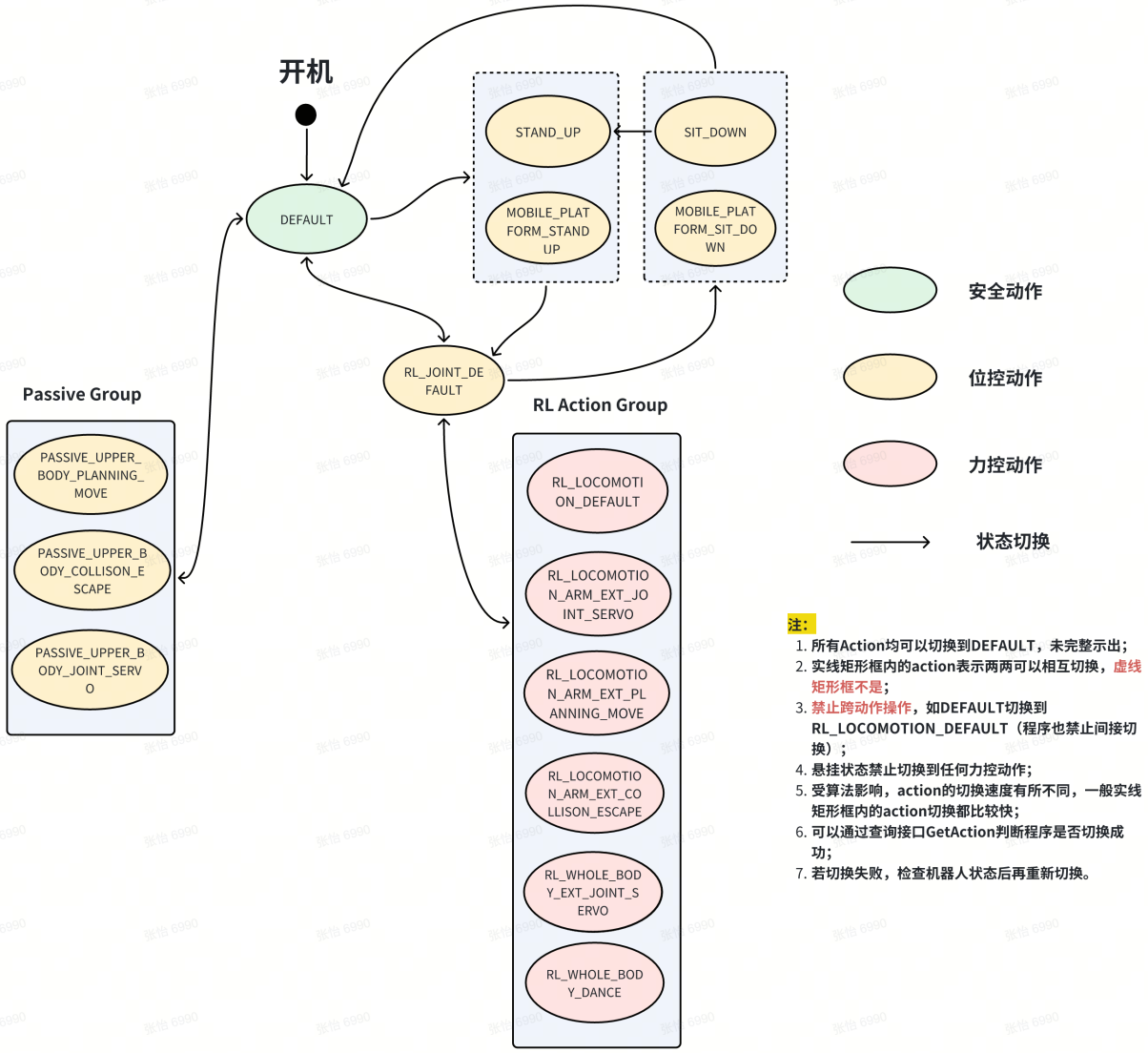
There are specific rules for calling motion control interfaces—only certain interfaces can be invoked under specific states (Actions). Please refer to the following Action list for details (only publicly available Actions are listed; do not use any other Actions).

No.	Action Type	Action Code	Chinese Name	Detailed Description	Callable Interfaces
0	Safety Action	DEFAULT	Default Mode	The default action after motion control starts	
1	Position Control Action	RL_JOINT_DEFAULT	Reinforcement Position-Control Standing Mode	Reinforcement-based position-controlled standing on the robot, with straight-leg stance	
2	Position Control Action	PASSIVE_UPPER_BODY_JOINT_SERVO	Lower Body Passive, Upper Body Servo Mode	Lower limbs disabled; arms can receive external joint servo commands	T2

No.	Action Type	Action Code	Chinese Name	Detailed Description	Callable Interfaces
3	Position Control Action	PASSIVE_UPPER_BODY_PLANNING_MOVE	Lower Body Passive, Upper Body RRT Planning Mode	Lower limbs disabled; upper body path planning; accepts target joint angles or SE3 poses via RPC	S1
4	Position Control Action	PASSIVE_UPPER_BODY_COLLISION_ESCAPE	Lower Body Passive, Upper Body Collision Escape Mode	Used together with `PASSIVE_UPPER_BODY_PLANNING_MOVE`; activated when collision occurs and planning interface becomes unavailable; automatically separates limbs (rarely used)	
4	Position Control Action	SIT_DOWN	Position-Control Sitting Mode	Lower limb motion; upper body remains static; used with a chair for sitting down	
5	Position Control Action	STAND_UP	Position-Control Standing-Up Mode	Lower limb motion; upper body remains static; used with a chair for standing up	
7	Position Control Action	MOBILE_PLATFORM_SIT_DOWN	Mobile Platform Position-Control Sitting Mode	Lower limb motion; upper body remains static; used for sitting down during seated unpacking tasks	
6	Position Control Action	MOBILE_PLATFORM_STAND_UP	Mobile Platform Position-Control Standing-Up Mode	Lower limb motion; upper body remains static; used for standing up during seated unpacking tasks	
8	Force Control Action	RL_LOCOMOTION_DEFAULT	Reinforcement Locomotion Mode	Humanoid walking mode trained via reinforcement learning; arms swing naturally during walking	T1
9	Force Control Action	RL_LOCOMOTION_ARM_EXT_JOINT_SERVO	Reinforcement Locomotion with Upper Body Servo Mode	Lower body performs humanoid walking or standing; upper body accepts external joint servo commands; enables upper body motion during walking/standing (uses whole-body reinforcement model for enhanced stability)	T1, T2
10	Force Control Action	RL_LOCOMOTION_ARM_EXT_PLANNING_MOVE	Reinforcement Locomotion with Upper Body RRT Planning Mode	Lower body stands; upper body performs path planning; accepts target joint angles or SE3 poses via RPC	S1
11	Force Control Action	RL_LOCOMOTION_ARM_EXT_COLLISON_ESCAPE	Reinforcement Locomotion with Upper Body Collision Escape Mode	Used together with `RL_LOCOMOTION_ARM_EXT_PLANNING_MOVE`; activated when collision occurs and planning interface becomes unavailable; automatically separates limbs (rarely used)	

No.	Action Type	Action Code	Chinese Name	Detailed Description	Callable Interfaces
12	Force Control Action	RL_WHOLE_BODY_EXT_JOINT_SERVO	Whole-Body Reinforcement Locomotion Mode	Similar to `RL_LOCOMOTION_ARM_EXT_JOINT_SERVO`, but additionally supports waist motion control; on A2 robot, the waist has no actual DOF—waist motion is achieved via coordinated leg motors	T1, T2, T3
13	Force Control Action	RL_WHOLE_BODY_DANCE	Whole-Body Reinforcement Dance Mode	Executes predefined full-body dance motions	S2

Action Transition State Machine:



Channel Interface

Interface Code	Topic Name	Topic Description	Subscribe or Publish	Message Type	Notes
T1	`/motion/control/locomotion_velocity`	Lower Limb Control Command	Subscribe	aimdk::protocol::McLocomotionVelocityChannel	There are two modes: one for b remote-controlled walking, u `LocomotionMode_DEFAULT` (e value 0), and another navigation walking, u

Interface Code	Topic Name	Topic Description	Subscribe or Publish	Message Type	Notes
					`LocomotionMode_NAVIGATION` (enum value 1). Navigation wal responds more sensitively velocity commands, while rem controlled walking is smoother. Maximum forward speed is 0.8 maximum backward speed is m/s, maximum lateral speed is m/s, and maximum rotati speed is 1.0 rad/s. Backward movement is recommended, and la movement is discouraged. For b locomotion, use forward velo combined with rotation.
T2	`/motion/control/move_waist`	Waist Control Command	Subscribe	aimdk::protocol::McMoveWaistChannel	Callable only RL_WHOLE_BODY_EXT_JOINT_SE mode.
T2	`/motion/control/move_waist_lift`	Squat Command	Subscribe	aimdk::protocol::McMoveWaistLiftChannel	Range limited from -0.15 to (callable only RL_WHOLE_BODY_EXT_JOINT_SE mode.
T2	`/motion/control/move_waist_pitch`	Waist Pitch (Bending) Command	Subscribe	aimdk::protocol::McMoveWaistPitchChannel	Range limited to -0.5236 to 0.5 can only be invoked RL_WHOLE_BODY_EXT_JOINT_SE mode
T2	`/motion/control/move_waist_twist`	Waist Twist (Turning) Command	Subscribe	aimdk::protocol::McMoveWaistTwistChannel	Range limited to -0.5236 to 0.5 can only be invoked RL_WHOLE_BODY_EXT_JOINT_SE mode
T3	`/motion/control/arm_joint_command`	Direct Arm Joint Control Command	Subscribe	sensor_msgs::msg::JointState	Can only be invoked JOINT_SERVO mode. Motion co will not perform trajectory plan for arm joints; commands passed directly through to Only position control is support Must provide angles for all 14 jo at once: the first 7 correspon the left arm, and the last 7 to right arm.
T3	`/motion/control/neck_joint_command`	Direct Neck Joint Control Command	Subscribe	sensor_msgs::msg::JointState	Can only be invoked JOINT_SERVO mode. The neck two joints na [idx27_head_joint1, idx28_head_joint2]. Only pos control is supported.
T*	`/motion/control/arm_joint_state`	Arm joint state	Publish	sensor_msgs::msg::JointState	Arm joint state includes 7 joints arm. For the first five joints of arms, the `position` , `velocit` and `effort` fields contain data. For the last two joints, the `position` field is

Interface Code	Topic Name	Topic Description	Subscribe or Publish	Message Type	Notes
					(representing equivalent s joint values). Published at 100 Hz.
T*	`/motion/control/neck_joint_state`	Neck joint state	Publish	sensor_msgs::msg::JointState	Neck joint state includes two jo with valid `position` , `velocit` and `effort` fields. Published at 100 Hz.
T*	`/motion/control/hand_joint_state`	Hand joint state	Publish	sensor_msgs::msg::JointState	Hand joint state includes 6 jo per dexterous hand. The `posit` and `effort` fields are valid, `position` ranging from 0 to 2 and `effort` ranging from 0 5700. Published at 100 Hz. Due to mechanical linka rotating shafts, springs, and o internal mechanisms within fingers, torque readings occasionally show slightly nega values (within -1000), which normal. Negative values may also ap when external forces act on fingers—this is also no behavior.

Interfaces marked with an asterisk (*) can be called under any action mode.## RPC Interfaces

Considering current software logging and network resource usage, it is recommended that all RPC service interfaces be called at a frequency of 1 Hz or lower.

SetAction/GetAction Operation Guidelines:

- The `SetAction` interface is asynchronous. It returns immediately upon invocation. A successful return does **not** indicate that the Action switch has completed. You must use the `GetAction` interface to poll until the Action switch is confirmed complete before proceeding to the next operation.
- The recommended polling interval for `GetAction` is 1 second (to avoid high-frequency RPC calls). If a 1-second delay is considered too long, it may be set to 500 ms. Intervals shorter than this are **not recommended** .

Interface Code	Interface Name	Description	Request Message Type	Response Message Type
S*	`pb:/aimdk.protocol.McActionService/SetAction`	Set Action	aimdk::protocol::McActionRequest	aimdk::protocol::CommonResponse
S*	`pb:/aimdk.protocol.McActionService/GetAction`	Get Action	aimdk::protocol::CommonRequest	aimdk::protocol::McActionResponse
S*	`pb:/aimdk.protocol.McActionService/GetAvailableActions`	Get Available Actions	aimdk::protocol::CommonRequest	aimdk::protocol::AvailableActionsRes

Interface Code	Interface Name	Description	Request Message Type	Response Message Type
S*	pb:/aimdk.protocol.McActionService/GetAvailableCommands	Get Action Switch Commands	aimdk::protocol::CommonRequest	aimdk::protocol::AvailableCommandR
S*	pb:/aimdk.protocol.McDataService/GetJointState	Get Joint State	aimdk::protocol::CommonRequest	aimdk::protocol::McJointStatesRespo
S*	pb:/aimdk.protocol.McDataService/GetJointAngle	Get Joint Angles	aimdk::protocol::CommonRequest	aimdk::protocol::McJointAnglesRespo
S*	pb:/aimdk.protocol.McMotionService/SetHandCommand	Set Hand Command	aimdk::protocol::HandCommandRequest	aimdk::protocol::CommonResponse
S*	pb:/aimdk.protocol.McDataService/GetHandState	Get Dexterous Hand State	aimdk::protocol::CommonRequest	aimdk::protocol::HandStateResponse

Interface Code	Interface Name	Description	Request Message Type	Response Message Type
S*	`pb:/aimdk.protocol.McMotionService/SetNeckCommand`	Set Neck Command	aimdk::protocol::NeckCommandRequest	aimdk::protocol::CommonResponse
S*	`pb:/aimdk.protocol.McDataService/GetNeckState`	Get Neck State	aimdk::protocol::CommonRequest	aimdk::protocol::NeckStateResponse
S2	`pb:/aimdk.protocol.McMotionService/GetDanceTypeList`	Get Dance List	aimdk::protocol::CommonRequest	aimdk::protocol::McDanceResponse
S2	`pb:/aimdk.protocol.McMotionService/SelectDanceType`	Select Dance	aimdk::protocol::DanceTypeRequest	aimdk::protocol::CommonResponse

Functionality

Send an SE3 target pose for the desired left or right arm to the robot via RPC. The robot will plan a motion based on its current state and execute the planned trajectory to reach the target pose. This interface is only available on the Flagship model.

Procedure

1. Switch to an Action that supports `PLANNING_MOVE`

   The following Actions support upper limb planning functionality:

Action Name	Description	Switching Sequence After Power-On
PASSIVE_UPPER_BODY_PLANNING_MOVE	Lower limbs fully disabled; upper limbs perform planned motion	DEFAULT → PASSIVE_UPPER_BODY_PLANNING_MOVE
RL_LOCOMOTION_ARM_EXT_PLANNING_MOVE	Lower limbs can perform reinforcement-learning-based locomotion; upper limbs perform planned motion	DEFAULT → RL_JOINT_DEFAULT → RL_LOCOMOTION_ARM_EXT_PLANNING_MOVE

2. Send the target pose

Use an RPC service with the interface: `pb:/aimdk.protocol.McMotionService/PlanningMove`

Example request:

```
import numpy as np
import json
import requests
import time
from datetime import datetime

def create_header():
    now = datetime.now()
    header = {
        "timestamp": {
            "seconds": int(now.timestamp()),
            "nanos": now.microsecond * 1000,
            "ms_since_epoch": int(now.timestamp() * 1000),
        },
        "control_source": "ControlSource_MANUAL"
    }
    return header

def send_json_data(json_data, url):
    json_str = json.dumps(json_data, indent=2)
    print("Sending JSON data:")
    print(json_str)
    print("")

    headers = {'content-type': 'application/json'}
    response = requests.post(url, headers=headers, data=json_str)

    print(f"Response: {response.status_code}")
    print(response.text)
    print("")

def main():
    url = "http://192.168.100.100:56322/rpc/aimdk.protocol.McMotionService/PlanningMove"
    leftTrl = [0.35, 0.35, 0.25]
    leftQuat = [0.818758, 0.570822, -0.0606913, -0.0106989]
    rightTrl = [0.35, -0.35, 0.25]
    rightQuat = [0.818761, -0.570817, -0.0606912, 0.0106992]
    data = {
        "header": create_header(),
        "group": "McPlanningGroup_LEFT_ARM",
        "mode": "McPlanningMode_DEFAULT",
        "target": {
            "type": "SE3",
            "left": {"position":{"x":leftTrl[0],"y":leftTrl[1],"z":leftTrl[2]},"orientation":
{"x":leftQuat[1],"y":leftQuat[2],"z":leftQuat[3],"w":leftQuat[0]}},
            "right":{"position":{"x":rightTrl[0],"y":rightTrl[1],"z":rightTrl[2]},"orientation":
{"x":rightQuat[1],"y":rightQuat[2],"z":rightQuat[3],"w":rightQuat[0]}},
        },
        "reference": {
            "joint_position": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
        },
    }
```

```
    send_json_data(data, url)

if __name__ == "__main__":
    main()
```

In the request data:

- `"group"` specifies which arm to move: `"McPlanningGroup_LEFT_ARM"` for the left arm, `"McPlanningGroup_RIGHT_ARM"` for the right arm.
- `"target"` defines the desired end-effector pose:
  - `"position"` specifies the target Cartesian position.
  - `"orientation"` specifies the target orientation as a quaternion.
- `"reference"` contains `"joint_position"` , which provides reference joint angles. Typically, this can be set to all zeros. If non-zero values are provided and the length matches the robot arm's joint count (7 values), the planner will attempt to reach the target pose while staying as close as possible to these reference joint angles.

3. Query task status (optional)

Refer to `get_task_state.sh` . After calling the RPC interface, the current task status will be returned. Common status codes and their meanings are as follows:

ID	Status	Description
0	CommonState_UNKNOWN	Task does not exist
1	CommonState_SUCCESS	Planning and motion completed successfully
2	CommonState_FAILURE	Planning or motion execution failed
3	CommonState_ABORTED	Planning motion not allowed in current state; task aborted
4	CommonState_PENDING	Previous task is being planned or executed; current task is waiting
5	CommonState_RUNNING	Task is being planned or executed

During polling, if the returned status is 1, 2, or 3, it indicates the task has already finished. The motion controller will no longer retain the task information. Therefore, any subsequent queries for the same task will return status 0, indicating the task no longer exists.### Notes

4. Description of target points

A target point represents the position and orientation of the robot's palm in the robot's base coordinate frame. It has a certain offset relative to the arm end-effector defined in the URDF, as follows:

Arm	End-effector Link Name	Position Offset (x, y, z)	Orientation Offset (Quaternion w, x, y, z)
Left	left_arm_link07	[-0.1, 0.0, 0.0]	[0.0, 0.0, 0.70710678, 0.70710678]
Right	right_arm_link07	[0.1, 0.0, 0.0]	[-0.70710678, 0.70710678, 0.0, 1.0]

This offset ensures consistent end-effector orientations for both arms, facilitating computation. To compute the palm pose, you can first subscribe to the ROS 2 `tf` topic and then calculate the palm position. An example calculation is shown below:

```python
import numpy as np
from scipy.spatial.transform import Rotation as R

def pose_array_to_se3(pose_array, w_first=True):
    trl = pose_array[:3]
    quat = np.array([pose_array[4], pose_array[5], pose_array[6], pose_array[3]]) if w_first else
pose_array[3:]
    rotm = R.from_quat(quat).as_matrix()
    se3 = np.eye(4)
    se3[:3, :3] = rotm
    se3[:3, 3] = trl
    return se3

def se3_to_pose_array(se3, w_first=True):
    rotm = se3[:3, :3]
    trl = se3[:3, 3]
    quat_xyzw = rot.as_quat()
    if w_first:
        quat = np.array([quat_xyzw[3], quat_xyzw[0], quat_xyzw[1], quat_xyzw[2]])
    else:
        quat = quat_xyzw
    return np.concatenate([trl, quat])

def main():
    left_flange_array = np.array([0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0]) # 改成自己的从tf订阅到的数据
    right_flange_array = np.array([0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0]) # 改成自己的从tf订阅到的数据

    left_flange_tcp_array = np.array([-0.1, 0.0, 0.0, 0.0, 0.0, 0.70710678, 0.70710678])
    right_flange_tcp_array = np.array([0.1, 0.0, 0.0, -0.70710678, 0.70710678, 0.0, 0.0])

    left_flange_se3 = pose_array_to_se3(left_flange_array)
    right_flange_se3 = pose_array_to_se3(right_flange_array)
    left_flange_tcp_se3 = pose_array_to_se3(left_flange_tcp_array)
    right_flange_tcp_se3 = pose_array_to_se3(right_flange_tcp_array)

    left_tcp_se3 = left_flange_se3 @ left_flange_tcp_se3
    right_tcp_se3 = right_flange_se3 @ right_flange_tcp_se3

    left_tcp_array = se3_to_pose_array(left_tcp_se3)
    right_tcp_array = se3_to_pose_array(right_tcp_se3)
```

5. Motion range boundaries

   Based on the current URDF model parameters, for the left arm, the rotation center of joints 1 and 2 is located at (-0.03779, 0.191, 0.3421). Therefore, any target point sent to the left arm must satisfy:
   0.33690 < ||(x + 0.03779, y - 0.191, z - 0.3421)|| < 0.60939

   Similarly, any target point sent to the right arm must satisfy:
   0.33690 < ||(x - 0.03779, y + 0.191, z - 0.3421)|| < 0.60939

   Target points outside this range are definitely unreachable. Even if the position satisfies the above condition, an unreasonable orientation may still result in unreachable configurations.

6. Positioning accuracy

   Positioning accuracy is primarily affected by three factors:

Source	Impact
Planning Algorithm	Minimal: maximum positional error is approximately 0.005 m. Generally, the longer the robot's motion path, the smaller the error tends to be.
Hardware	Significant: maximum positional error is typically around 0.02 m. This may vary due to differences between individual robots; actual measurements should be used for precise assessment.

Source	Impact
Base Movement	Significant: if the robot is standing with lower-limb force control enabled, large or distant arm motions may cause the robot's base to shift or even take steps to maintain balance. If the robot only shifts its base (without stepping), the maximum positional error is about 0.02 m. However, if the robot takes steps to maintain balance, the error becomes unpredictable. If the lower limbs are disabled and the base is fixed, this error source can be ignored.

Protobuf Message Types### `aimdk::protocol::ToolInertia`

Field	Type	Description
ixx	double	Principal moment of inertia along X-axis
iyy	double	Principal moment of inertia along Y-axis
izz	double	Principal moment of inertia along Z-axis
ixy	double	Product of inertia in the XY plane
ixz	double	Product of inertia in the XZ plane
iyz	double	Product of inertia in the YZ plane

`aimdk::protocol::McToolPayload`

Field	Type	Description
mass	double	Payload mass
mass_center	aimdk::protocol::Vec3	Payload center of mass
ToolInertia	aimdk::protocol::ToolInertia	Payload inertia

`aimdk::protocol::McToolConfig`

Field	Type	Description
id	int32	Tool ID
name	string	Tool name
config_file	string	Configuration file path
payload	aimdk::protocol::McToolPayload	Payload parameters
tcp	aimdk::protocol::SE3Pose	Tool Center Point (TCP)

`aimdk::protocol::McAddToolRequest`

Request to configure the robot end-effector

Field	Type	Description
header	aimdk::protocol::RequestHeader	Request header

Field	Type	Description
data	aimdk::protocol::McToolConfig []	Request payload

`aimdk::protocol::McGetToolResponse`

Response for retrieving robot end-effector configuration

Field	Type	Description
header	aimdk::protocol::RequestHeader	Response header
data	aimdk::protocol::McToolConfig []	Response payload

`aimdk::protocol::McRobotToolConfig`

Field	Type	Description
left_arm	aimdk::protocol::McToolConfig	Left arm tool configuration
right_arm	aimdk::protocol::McToolConfig	Right arm tool configuration

`aimdk::protocol::McRobotToolRequest`

Request to configure robot end-effectors for both arms

Field	Type	Description
header	aimdk::protocol::RequestHeader	Request header
data	aimdk::protocol::McRobotToolConfig	Request payload

Robot hand control request

Field	Type	Description
header	aimdk::protocol::RequestHeader	Request header
data	aimdk::protocol::McRobotToolConfig	Request data

`aimdk::protocol::McToolPayloadRequest`

Field	Type	Description
left_arm	aimdk::protocol::McToolPayload	
right_arm	aimdk::protocol::McToolPayload	

`aimdk::protocol::AimMasterControlSource`

AimMaster control source

Name	Number	Description
AimMasterControlSource_AUTO	0	Autonomous control

Name	Number	Description
AimMasterControlSource_MANUAL	1	Manual control

`aimdk::protocol::AimMasterControlAxis`

AimMaster joystick axes

Field	Type	Description
up_down	double	Vertical axis
left_right	double	Horizontal axis

`aimdk::protocol::AimMasterControlButton`

AimMaster joystick buttons

Field	Type	Description
up_down	oneof {bool up} {bool down}	Joystick button: up/down
left_right	oneof {bool left} {bool right}	Joystick button: left/right

`aimdk::protocol::AimMasterControlInfo`

AimMaster control

Field	Type	Description
type	aimdk::protocol::AimMasterControlInfo::Type	Joystick type
data	oneof {AimMasterControlAxis axis} {AimMasterControlButton button}	Joystick data

`aimdk::protocol::AimMasterControlInfo::Type`

Name	Number	Description
Type_UNKNOWN	0	
Type_AXIS	1	Joystick type: axis type, data type is DOUBLE
Type_BUTTON	2	Joystick type: button type, data type is BOOL

`aimdk::protocol::AimMasterControl`

AimMaster control message

Field	Type	Description
mode	aimdk::protocol::AimMasterControl::Mode	Control mode
left	aimdk::protocol::AimMasterControlInfo	Joystick: left
right	aimdk::protocol::AimMasterControlInfo	Joystick: right

`aimdk::protocol::AimMasterControl::Mode`

Name	Number	Description
Mode_UNKNOWN	0	
Mode_UPPER_BODY	1	Control mode: upper body
Mode_WHOLE_BODY	2	Control mode: whole body
Mode_HEAD	3	Control mode: head

`aimdk::protocol::AimMasterControlSourceRequest`

Interface structure request for interaction with the client

Field	Type	Description
header	aimdk::protocol::RequestHeader	
target	aimdk::protocol::AimMasterControlSource	

Interface structure response for interaction with the client.

Field	Type	Description
header	aimdk::protocol::ResponseHeader	
current	aimdk::protocol::AimMasterControlSource	

`aimdk::protocol::AimMasterControlCommandRequest`

AimMaster control operation command request.

Field	Type	Description
header	aimdk::protocol::RequestHeader	Request header
command	aimdk::protocol::AimMasterControlCommandRequest::Command	AimMaster control command

`aimdk::protocol::AimMasterControlCommandRequest::Command`

AimMaster control command.

Name	Number	Description
Command_UNKNOWN	0	Unknown command
Command_ROBOT_INIT	1	Pose initialization command

`aimdk::protocol::QuestProVr`

VR

Field	Type	Description
pose	aimdk::protocol::SE3Pose	VR pose

Field	Type	Description
axis_x	double	
axis_y	double	
index_trig	double	
hand_trig	double	
key_one	bool	
key_two	bool	

`aimdk::protocol::QuestProVrController`

VR controller

Field	Type	Description
left	aimdk::protocol::QuestProVr	
right	aimdk::protocol::QuestProVr	

`aimdk::protocol::McRemoteControlMode`

Name	Number	Description
McRemoteControlMode_UNKNOWN	0	Unknown mode
McRemoteControlMode_AUTO	100	Autonomous control mode
McRemoteControlMode_FULL_BODY	200	Full-body control mode
McRemoteControlMode_UPPER_BODY	300	Upper-body control mode

`aimdk::protocol::McRemoteControlAxis`

Remote control joystick

Field	Type	Description
up_down	double	Vertical joystick: 0–1 = up, -1–0 = down
left_right	double	Horizontal joystick: -1–0 = left, 0–1 = right

`aimdk::protocol::McRemoteControlButton`

Remote control buttons

Field	Type	Description
up	bool	Vertical joystick
down	bool	
left	bool	Horizontal joystick

Field	Type	Description
right	bool	

Complete remote control message

Field	Type	Description
mode	aimdk::protocol::McRemoteControlMode	Remote control mode
left_axis	aimdk::protocol::McRemoteControlAxis	Left joystick
right_button	aimdk::protocol::McRemoteControlButton	Right button

`aimdk::protocol::UniversalGamepad`

Universal gamepad

Field	Type	Description
left_left_right	float	axis0
left_up_down	float	axis1
lt	float	axis2
right_left_right	float	axis3
right_up_down	float	axis4
rt	float	axis5
a	int32	button0
b	int32	button1
x	int32	button2
y	int32	button3
lb	int32	button4
rb	int32	button5
back	int32	button6
start	int32	button7
logitech	int32	button8
left_ga	int32	button9
right_ga	int32	button10
hat0_x	int32	hat0_x
hat0_y	int32	hat0_y

`aimdk::protocol::QuestProVrControllerChannel`
VR controller channel

Field	Type	Description
header	aimdk::protocol::Header	Message header
data	aimdk::protocol::QuestProVrController	Data

`aimdk::protocol::UniversalGamepadChannel`
Universal gamepad channel

Field	Type	Description
header	aimdk::protocol::Header	Message header
data	aimdk::protocol::UniversalGamepad	Data

`aimdk::protocol::AimMasterControlChannel`
AimMaster control channel

Field	Type	Description
header	aimdk::protocol::Header	Message header
data	aimdk::protocol::AimMasterControl	Data

`aimdk::protocol::McRemoteControlChannel`
Remote controller control channel

Field	Type	Description
header	aimdk::protocol::Header	Message header
data	aimdk::protocol::McRemoteControlInfo	Data

`aimdk::protocol::ServoMoveGoalReachCondition`
Servo motion goal reach condition

Field	Type	Description
position_tolerance	double	
orientation_tolerance	double	
keep_time	double	

`aimdk::protocol::ServoMoveVelocity`
Servo motion velocity

Field	Type	Description
linear	double	
angular	double	

Servo Move Command

Field	Type	Description
left	aimdk::protocol::SE3Pose	
right	aimdk::protocol::SE3Pose	
servo_velocity	aimdk::protocol::ServoMoveVelocity	
goal_reach_condition	aimdk::protocol::ServoMoveGoalReachCondition	

`aimdk::protocol::ServoMoveResult`

Servo Move Pose Result

Field	Type	Description
status	aimdk::protocol::ServoMoveResult::Status	

`aimdk::protocol::ServoMoveResult::Status`

Name	Number	Description
SUCCESS	0	
FAILURE	1	

`aimdk::protocol::GetPlanningMoveTargetResponse`

Field	Type	Description
header	aimdk::protocol::ResponseHeader	
planning_move_request	aimdk::protocol::PlanningMoveRequest	

`aimdk::protocol::McControlInfo`

Field	Type	Description
mode	aimdk::protocol::ControlSource	

`aimdk::protocol::McWorkState`

Robot Work State

Name	Number	Description
McWorkState_DISABLED	0	

Name	Number	Description
McWorkState_ENABLED	1	

`aimdk::protocol::McState`

Robot State

Field	Type	Description
timestamp	uint64	
work_state	aimdk::protocol::McWorkState	
action_info	aimdk::protocol::McActionInfo	
control_info	aimdk::protocol::McControlInfo	
is_collisioned	bool	
sit_stand_state	aimdk::protocol::McSitStandState	
is_walking	bool	
enable_collision	bool	
will_be_collided	bool	
enable_future_collision	bool	

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Response header
state	aimdk::protocol::McWorkState	

`aimdk::protocol::McStateResponse`

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Response header
state	aimdk::protocol::McState	

`aimdk::protocol::McSitStandState`

Robot sit/stand state publication

Name	Number	Description
McSitStandState_DEFAULT	0	
McSitStandState_SIT_SOON	1	
McSitStandState_STAND_BACKWARD	2	

Name	Number	Description
McSitStandState_STAND_FORWARD	3	
McSitStandState_STAND_SOON	4	
McSitStandState_SIT_DONE	5	
McSitStandState_STAND_DONE	6	
McSitStandState_SITING	7	
McSitStandState_STANDING	8	
McSitStandState_SIT_BCK	9	Reserved interface
McSitStandState_STAND_BCK	10	
McSitStandState_MOBILE_PLATFORM_SIT_SOON	100	
McSitStandState_MOBILE_PLATFORM_STAND_BACKWARD	101	
McSitStandState_MOBILE_PLATFORM_STAND_FORWARD	102	
McSitStandState_MOBILE_PLATFORM_STAND_SOON	103	
McSitStandState_MOBILE_PLATFORM_SIT_DONE	104	
McSitStandState_MOBILE_PLATFORM_STAND_DONE	105	

`aimdk::protocol::McWorkModeRequest`

Set working mode

Field	Type	Description
header	aimdk::protocol::RequestHeader	Request header
mode	aimdk::protocol::ControlSource	Mode

`aimdk::protocol::McWorkModeResponse`

Get working mode

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Response header
mode	aimdk::protocol::ControlSource	Mode

`aimdk::protocol::McStateChannel`

Remote controller control channel

topic : /mc/base/state

Field	Type	Description
header	aimdk::protocol::Header	Message header
state	aimdk::protocol::McState	

MC Jaka working mode

Name	Number	Description
McJakaWorkMode_DEFAULT	0	
McJakaWorkMode_SERVO	1	

`aimdk::protocol::McJakaStateResponse`

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Response header
work_mode	aimdk::protocol::McJakaWorkMode	

`aimdk::protocol::StandRequest`

Stand-in-place request

Field	Type	Description
header	aimdk::protocol::RequestHeader	Common request header
stance	aimdk::protocol::StandRequest::Stance	

`aimdk::protocol::StandRequest::Stance`

Upper limb control type: stay still or return to standard posture

Name	Number	Description
FREEZE	0	Stay still
STAND	1	Return to standard posture

`aimdk::protocol::AdjustCommand`

Field	Type	Description
name	string	Joint or end-effector name
arm_type	uint32	1: left arm, 2: right arm, 3: both arms
mode	aimdk::protocol::AdjustCommand::PressType	Press mode
step	double	Step size
velocity	double	Velocity

`aimdk::protocol::AdjustCommand::PressType`

Name	Number	Description
UNKNOWN	0	
Short	1	Mode: 1. Tap; 2. Long press
Long	2	

`aimdk::protocol::AdjustCommandRequest`

Field	Type	Description
header	aimdk::protocol::RequestHeader	
command	aimdk::protocol::AdjustCommand	

`aimdk::protocol::AdjustEndPosition`

Field	Type	Description
name	string	End-effector name
x	double	
y	double	
z	double	
r_x	double	
r_y	double	
r_z	double	
around_singularity_flag	bool	Near singularity flag

Field	Type	Description
header	aimdk::protocol::RequestHeader	
end_positions	aimdk::protocol::AdjustEndPosition []	

`aimdk::protocol::CollisionRequest`

Collision feature request message

Field	Type	Description
header	aimdk::protocol::RequestHeader	
enable	bool	

`aimdk::protocol::CollisionResponse`

Collision feature status response message

Field	Type	Description
header	aimdk::protocol::ResponseHeader	
CollisionDetectionState	bool	

`aimdk::protocol::DanceTypeRequest`

Dance type request

Field	Type	Description
header	aimdk::protocol::RequestHeader	
dance_name	string	Dance name

`aimdk::protocol::McDanceResponce`

Retrieve dance list

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Response header
dance_name	string[]	Dance names
dance_model_path	string[]	Model storage paths
dance_data_path	string[]	Dance data storage paths
dance_time	double[]	Duration of each dance (s)

`aimdk::protocol::MovJRequest`

MovJ request

Field	Type	Description
header	aimdk::protocol::RequestHeader	Request header
group	aimdk::protocol::McPlanningGroup	Planning group
param	aimdk::protocol::MotionParam	Motion parameters (e.g., velocity, acceleration)
angles	double[]	Joint angles (unit: radians)

MovJ Request

Field	Type	Description
header	aimdk::protocol::RequestHeader	Request Header
group	aimdk::protocol::JointControlRequest::JointGroup	Joint Group
cmds	aimdk::protocol::JointControlRequest::Control[]	Control Commands

`aimdk::protocol::JointControlRequest::JointGroup`

Name	Number	Description
UNKNOWN	0	
LEFT_ARM	1	
RIGHT_ARM	4	
DUAL_ARM	7	
WAIST_LIFT	11	
WAIST_PITCH	12	
HEAD	13	

`aimdk::protocol::Control::Control`

Field	Type	Description
name	string	Joint Name
mode	aimdk::protocol::Control::Mode	Control Mode
angle	double	Joint Angle (unit: radian)

`aimdk::protocol::Control::Mode`

Name	Number	Description
ABSOLUTE	0	Absolute Angle Mode
RELATIVE	1	Relative Angle Mode

`aimdk::protocol::MotionParam`

Motion Parameters

Field	Type	Description
velocity_scale	double	
acceleration_scale	double	

`aimdk::protocol::McLocomotionVelocityChannel`

Remote Controller Locomotion Velocity Channel

topic : /motion/control/locomotion_velocity

Field	Type	Description
header	aimdk::protocol::Header	Message Header
data	aimdk::protocol::LocomotionVelocity	Data

`aimdk::protocol::McMoveWaistChannel`
Humanoid Waist Control

topic : /motion/control/move_waist

Field	Type	Description
header	aimdk::protocol::Header	Message Header
data	aimdk::protocol::WaistMoveValue	Data

`aimdk::protocol::McMoveWaistLiftChannel`
Humanoid Waist Control – Waist Lift

topic : /motion/control/move_waist_lift

Field	Type	Description
header	aimdk::protocol::Header	Message Header
waist_lift_value	double	Vertical Waist Displacement (range: -0.15 ~ +0.04, unit: m)

Humanoid waist control – waist tilt left/right

topic: /motion/control/move_waist_sideways

Field	Type	Description
header	aimdk::protocol::Header	Message header
waist_sideways_value	double	Data

`aimdk::protocol::McMoveWaistPitchChannel`
Humanoid waist control – waist pitch forward/backward

topic: /motion/control/move_waist_pitch

Field	Type	Description
header	aimdk::protocol::Header	Message header
waist_pitch_value	double	Data

`aimdk::protocol::McMoveWaistTwistChannel`
Humanoid waist control – waist twist

topic: /motion/control/move_waist_twist

Field	Type	Description
header	aimdk::protocol::Header	Message header
waist_twist_value	double	Data

`aimdk::protocol::McHandCommandChannel`
Hand command channel

topic: /motion/control/hand_command

Field	Type	Description
header	aimdk::protocol::Header	Message header
data	aimdk::protocol::HandCommand	Hand command

`aimdk::protocol::MotorAckermannJoystickChannel`
Single-joystick Ackermann control

Field	Type	Description
header	aimdk::protocol::Header	Message header
turn_pos	double	rad/s; + counterclockwise, - clockwise; gamepad_control: turn_range
move_vel	double	m/s; + forward, - backward; gamepad_control: max_velocity

`aimdk::protocol::MotorRotationChannel`
In-place rotation

Field	Type	Description
header	aimdk::protocol::Header	Message header
rot_vel	double	rad/s; + counterclockwise, - clockwise; gamepad_control: max_w

Head pitch

Field	Type	Description
header	aimdk::protocol::Header	Message header
head_pitch_vel	double	rad/s; + downward pitch, - upward pitch; arm_joint_limits: - name: “head_link”

`aimdk::protocol::MotorWaistPitchChannel`
Waist pitch (bending forward/backward)

Field	Type	Description
header	aimdk::protocol::Header	Message header
waist_pitch_vel	double	rad/s; + forward bend, - backward bend; arm_joint_limits: - name: “idx10_joint_body_link3”

`aimdk::protocol::MotorWaistLiftChannel`
Waist lift (vertical movement)

Field	Type	Description
header	aimdk::protocol::Header	Message header
waist_lift_vel	double	m/s; + upward, - downward; arm_joint_limits: - name: “idx09_joint_body_link2”

`aimdk::protocol::McArmControlChannel`

Dual-arm control message

Topic: /motion/control/joint_arm_control

Field	Type	Description
header	aimdk::protocol::Header	Message header containing sequence number, timestamp, etc.
flag	uint32	Control flag: 0 for left arm, 1 for right arm, 2 for both arms
positions	double[]	Joint positions, velocities, accelerations, and efforts; typically contains 14 elements: first 7 for left arm, last 7 for right arm. Unit: rad
velocities	double[]	The velocity of the trajectory. Unit: rad/s
accelerations	double[]	The acceleration of the trajectory. Unit: rad/s²
effort	double[]	The effort of the trajectory. Unit: N

Linear motion pose request

Field	Type	Description
left	aimdk::protocol::SE3Pose	
right	aimdk::protocol::SE3Pose	

`aimdk::protocol::LinearMoveRequest`

Field	Type	Description
header	aimdk::protocol::RequestHeader	Request header
group	aimdk::protocol::McPlanningGroup	Planning group
param	aimdk::protocol::MotionParam	Motion parameters, including velocity, acceleration, deceleration, etc.
target	aimdk::protocol::LinearMoveTarget	Target pose

`aimdk::protocol::MovePoseRequest`

Pose motion request

Field	Type	Description
header	aimdk::protocol::RequestHeader	Request header
type	aimdk::protocol::ArmType	Arm type

Field	Type	Description
mode	aimdk::protocol::MovePoseMode	Motion mode: absolute or relative position
param	aimdk::protocol::MotionParam	Motion parameters, including velocity, acceleration, deceleration, etc.
is_block	bool	Whether to block
pose	aimdk::protocol::SE3Pose	Target pose

`aimdk::protocol::MoveTrajectoryTarget`

Field	Type	Description
left	aimdk::protocol::ScalarTrajectory	Left arm trajectory
right	aimdk::protocol::ScalarTrajectory	Right arm trajectory

`aimdk::protocol::MoveTrajectoryRequest`

Trajectory motion request

Field	Type	Description
header	aimdk::protocol::RequestHeader	Request header
target	aimdk::protocol::MoveTrajectoryTarget	

Field	Type	Description
x	double	Waist horizontal displacement (x and y ranges are set to 0, adjustment not allowed)
y	double	
z	double	Waist vertical displacement (range: -0.15 ~ +0.04, unit: m)
roll	double	Waist rotation angle (RPY ranges are all -30° ~ +30°; transmitted as radians: -0.5236 ~ 0.5236, unit: rad)
pitch	double	
yaw	double	

`aimdk::protocol::LocomotionGaitType`

Name	Number	Description
LocomotionGaitType_UNKNOWN	0	Gait: Unknown; should be set when not in walking action
LocomotionGaitType_STANCE	1	Gait: Standing still
LocomotionGaitType_WALK	2	Gait: Normal walking; steps in place when no velocity command is given
LocomotionGaitType_SMART_WALK	3	Gait: Normal walking; stands still when no velocity command is given
LocomotionGaitType_RUN	4	Gait: Running

Name	Number	Description
LocomotionGaitType_JUMP	5	Gait: Two-footed jump
LocomotionGaitType_HOP	6	Gait: One-footed hop

`aimdk::protocol::LocomotionMode`

Name	Number	Description
LocomotionMode_DEFAULT	0	Default mode, suitable for remote control scenarios
LocomotionMode_NAVIGATION	1	Navigation mode: faster velocity response; stepping starts at 0.05 m/s, and stops within 500 ms if no velocity command follows; suitable for autonomous navigation scenarios

`aimdk::protocol::LocomotionVelocity`

Field	Type	Description
mode	aimdk::protocol::LocomotionMode	Velocity control mode
forward_velocity	double	Forward velocity (unit: m/s; direction: + for forward, - for backward; range: limited by robot's actual speed capability)
lateral_velocity	double	Lateral velocity (unit: m/s; direction: + for left, - for right; range: limited by robot's actual speed capability)
angular_velocity	double	Angular (rotational) velocity (unit: rad/s; direction: + for counterclockwise/left turn, - for clockwise/right turn; range: limited by robot's actual speed capability)

`aimdk::protocol::LocomotionGaitRequest`

Robot gait request

Field	Type	Description
header	aimdk::protocol::RequestHeader	Request header
gait	aimdk::protocol::LocomotionGaitType	Robot gait type

Robot gait query response

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Response header
gait	aimdk::protocol::LocomotionGaitType	Robot gait type

`aimdk::protocol::PlanningMoveObjectInfo`

Field	Type	Description
name	string	
pose	aimdk::protocol::SE3Pose	

Field	Type	Description
type	aimdk::protocol::PlanningMoveObjectInfo::GeometryType	
parent_frame	string	

`aimdk::protocol::PlanningMoveObjectInfo::GeometryType`

Name	Number	Description
UNKNOWN	0	
BOX	1	
CYLINDER	2	
SPHERE	3	

`aimdk::protocol::PlanningTarget`

Field	Type	Description
type	aimdk::protocol::PlanningTarget::Type	
left	aimdk::protocol::SE3Pose	
right	aimdk::protocol::SE3Pose	
joints	double[]	

`aimdk::protocol::PlanningTarget::Type`

Name	Number	Description
UNKNOWN	0	
SE3	1	
JOINT	2	
EE	3	

`aimdk::protocol::PlanningReference`

Field	Type	Description
joint_position	double[]	Reference joint angles; the count must match the group's DOF
trajectory	aimdk::protocol::ScalarTrajectory	

Field	Type	Description
header	aimdk::protocol::RequestHeader	Request header
world_model	aimdk::protocol::McPlanningWorldModel	

Field	Type	Description
group	aimdk::protocol::McPlanningGroup	Planning group
mode	aimdk::protocol::McPlanningMode	Mode
target	aimdk::protocol::PlanningTarget	Target position
targets	aimdk::protocol::PlanningTarget []	Multiple target positions
reference	aimdk::protocol::PlanningReference	Reference position
obj_info	aimdk::protocol::PlanningMoveObjectInfo	
param	aimdk::protocol::MotionParam	Motion parameters

`aimdk::protocol::ComputeKinematicsRequest`

Field	Type	Description
header	aimdk::protocol::RequestHeader	Request header
group	aimdk::protocol::McPlanningGroup	Planning group
target	aimdk::protocol::PlanningTarget	Target position
reference	aimdk::protocol::PlanningReference	Reference position

`aimdk::protocol::ComputeKinematicsResponse`

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Response header
success_flag	bool	
ik_results	aimdk::protocol::ScalarTrajectory	Planned trajectory

`aimdk::protocol::ActiveMotionControlRequest`

RPC for pausing/resuming active motion.

Field	Type	Description
header	aimdk::protocol::RequestHeader	Request header
control_flag	bool	control_flag: 0 to resume; 1 to pause;

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Response header
task_id	uint64	
state	aimdk::protocol::CommonState	

Field	Type	Description
trajectory	aimdk::protocol::ScalarTrajectory	Planned trajectory

Name	Number	Description
McAction_DEFAULT	0	Default mode. The default action after motion control startup.
McAction_PASSIVE_UPPER_BODY_JOINT_SERVO	1	// Passive mode // Must switch to this action upon initial startup to enable subsequent operations. McAction_PASSIVE_DEFAULT = 1; Lower limbs disabled, upper limbs can receive external joint servo commands.
McAction_PASSIVE_UPPER_BODY_PLANNING_MOVE	2	Lower limbs passive, upper limbs RRT planning mode. Lower limbs disabled; upper limbs perform path planning, receiving target joint angles or SE3 poses via RPC.
McAction_PASSIVE_UPPER_BODY_COLLISON_ESCAPE	3	Lower limbs passive, upper limbs collision-avoidance mode. Lower limbs disabled; when upper limb collision occurs, planning becomes invalid, and switching to this mode allows movement to a non-collision pose.
McAction_JOINT_DEFAULT	100	Position-controlled standing mode. Both upper and lower limbs move to the standard standing posture.
McAction_JOINT_FREEZE	101	Position-controlled joint freeze mode. Both upper and lower limbs are locked at their current positions.
McAction_STAND_DEFAULT	201	Force-controlled standing mode. Lower limbs maintain force-controlled standing; upper limbs move to a predefined default pose and cannot be externally controlled. This mode allows servo control of the torso (waist), including squatting, bending (pitch), turning (yaw), and side tilting (roll).
McAction_STAND_ARM_FREEZE	202	Force-controlled standing with upper limbs frozen. Lower limbs in force-controlled standing; upper limbs locked at current positions.
McAction_STAND_ARM_EXT_JOINT_SERVO	203	Force-controlled standing with upper limb servo mode. Lower limbs maintain standing; arms can accept external joint servo commands.
McAction_STAND_ARM_EXT_JOINT_TRAJ	204	Force-controlled standing with upper limb trajectory planning mode. Designed for user-implemented upper limb path planning; receives joint angles via RPC, supporting any number of waypoints (recommended <500).
McAction_STAND_ARM_EXT_JOINT_ONLINE_TRAJ	205	Force-controlled standing with online upper limb trajectory mode. Designed for user-implemented upper limb path planning; receives real-time joint angles, velocities, etc., for 14 joints via topics, executing sequentially from a waypoint queue.
McAction_LOCOMOTION_DEFAULT	301	Locomotion mode. Arms swing naturally during walking. Supports omnidirectional walking on flat ground and slopes ≤5°, including slope holding. External control of upper limbs is prohibited.
McAction_LOCOMOTION_ARM_EXT_JOINT_SERVO	302	Locomotion with upper limb servo mode. Supports omnidirectional walking on flat terrain, allowing simultaneous external control of upper limbs while walking.

Name	Number	Description
McAction_RL_JOINT_DEFAULT	400	Reinforcement position-controlled standing mode. Robot stands with straight legs under reinforced position control.
McAction_RL_LOCOMOTION_DEFAULT	401	Reinforcement locomotion mode. Reinforcement learning-based walking with straight legs.
McAction_RL_LOCOMOTION_ARM_EXT_JOINT_SERVO	402	Reinforcement locomotion with upper limb servo mode. Lower limbs perform reinforcement-based walking; upper limbs accept external joint servo commands for simultaneous motion.
McAction_RL_LOCOMOTION_ARM_EXT_PLANNING_MOVE	403	Reinforcement standing with upper limb RRT planning mode. Lower limbs in force-controlled standing; upper limbs perform path planning, receiving target joint angles or SE3 poses via RPC.
McAction_RL_LOCOMOTION_ARM_EXT_COLLISON_ESCAPE	404	Reinforcement standing with upper limb collision-avoidance mode. Lower limbs in force-controlled standing; upper limbs avoid self-collision.
McAction_RL_WHOLE_BODY_DANCE	406	Whole-body dance mode
McAction_SIT_DOWN	501	Sit-down mode: robot sits down
McAction_STAND_UP	502	Stand-up mode: robot stands up
McAction_MOBILE_PLATFORM_SIT_DOWN	503	Mobile platform position-controlled sit-down mode: robot's mobile platform sits down
McAction_MOBILE_PLATFORM_STAND_UP	504	Mobile platform position-controlled stand-up mode: robot's mobile platform stands up
McAction_USE_EXT_CMD	5000	Robot uses extended command

Name	Number	Description
McActionStatus_UNKNOWN	0	Unknown status
McActionStatus_PREPARE	100	Preparing
McActionStatus_RUNNING	200	Running
McActionStatus_DONE	300	Completed

`aimdk::protocol::McActionInfo`

Field	Type	Description
current_action	aimdk::protocol::McAction	Currently running Action
ext_action	string	Extended Action
status	aimdk::protocol::McActionStatus	Action status

`aimdk::protocol::McActionCommand`

Field	Type	Description
action	aimdk::protocol::McAction	Action
ext_action	string	Extended Action

`aimdk::protocol::McActionRequest`

Robot Action command request

Field	Type	Description
header	aimdk::protocol::RequestHeader	Request header
command	aimdk::protocol::McActionCommand	

`aimdk::protocol::McActionResponse`

Robot Action command response

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Response header
info	aimdk::protocol::McActionInfo	

`aimdk::protocol::AvailableActionsResponse`

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Response header
actions	string[]	

`aimdk::protocol::AvailableCommandResponse`

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Response header
commands	aimdk::protocol::McActionCommand []	

`aimdk::protocol::GetSimulationObjRequest`

Get simulation object request

Field	Type	Description
header	aimdk::protocol::RequestHeader	Request header
obj_id	string[]	Object IDs

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Response header
obj_pose	aimdk::protocol::SE3Pose []	Object poses
error_code	int32	Error code

`aimdk::protocol::ArmType`

Arm type

Name	Number	Description
ArmType_UNKNOWN	0	Undefined
ArmType_LEFT	1	Left arm
ArmType_RIGHT	2	Right arm

`aimdk::protocol::MovePoseMode`

Motion pose mode: absolute or relative position

Name	Number	Description
MovePoseMode_UNKNOWN	0	Undefined
MovePoseMode_ABSOLUTE	1	Absolute position
MovePoseMode_RELATIVE	2	Relative position

`aimdk::protocol::McPlanningGroup`

Planning group

Name	Number	Description
McPlanningGroup_UNKNOWN	0	Unknown
McPlanningGroup_LEFT_ARM	1	Left arm
McPlanningGroup_LEFT_ARM_WAIST	2	Left arm + waist
McPlanningGroup_LEFT_ARM_WAIST_WHEEL	3	Left arm + waist + wheels
McPlanningGroup_RIGHT_ARM	4	Right arm
McPlanningGroup_RIGHT_ARM_WAIST	5	Right arm + waist
McPlanningGroup_RIGHT_ARM_WAIST_WHEEL	6	Right arm + waist + wheels
McPlanningGroup_DUAL_ARM	7	Dual arms
McPlanningGroup_DUAL_ARM_WAIST	8	Dual arms + waist
McPlanningGroup_DUAL_ARM_WAIST_WHEEL	9	Dual arms + waist + wheels

Name	Number	Description
McPlanningGroup_DUAL_ARM_WAIST_HEAD	10	Dual arms + waist + head
McPlanningGroup_WAIST_LIFT	11	Waist lift
McPlanningGroup_WAIST_PITCH	12	Waist pitch
McPlanningGroup_HEAD	13	Head

`aimdk::protocol::McPlanningMode`

Name	Number	Description
McPlanningMode_DEFAULT	0	Default obstacle-avoidance planning. left_target right_target
McPlanningMode_CLOSURE	100	Closed-chain planning, called after grasping an object, obj_center
McPlanningMode_GRAB	200	Grasping, obj_center
McPlanningMode_PLACE	300	Placing
McPlanningMode_LINEAR_MOVE	400	Linear motion

`aimdk::protocol::McPlanningWorldModel`

Task planning world model

This type has no fields.

`aimdk::protocol::McPlanningScene`

Task planning scene

This type has no fields.

`aimdk::protocol::CollisionRecoverRequest`

Collision recovery request

Field	Type	Description
header	aimdk::protocol::RequestHeader	Request header
type	aimdk::protocol::ArmType	Arm type

1. Action command interfaces related to assembly tasks

Name	Number	Description
PrimitiveType_UNKNOWN	0	
PrimitiveType_SLIDE_SPIRAL	1	Spiral search
PrimitiveType_SLIDE_SPIRAL_DUAL	2	Dual-arm spiral search
PrimitiveType_SLIDE_ZIGZAG	3	Zigzag search

Name	Number	Description
PrimitiveType_SLIDE_ZIGZAG_DUAL	4	Dual-arm zigzag search
PrimitiveType_SLEEVE	5	Force-controlled insertion/removal
PrimitiveType_SLEEVE_DUAL	6	Dual-arm force-controlled insertion/removal
PrimitiveType_CONTACT	7	Force-controlled contact
PrimitiveType_CONTACT_DUAL	8	Dual-arm force-controlled contact

`aimdk::protocol::PtTriggerConditionType`

Name	Number	Description
PtTriggerConditionType_UNKNOWN	0	
PtTriggerConditionType_AND	1	AND
PtTriggerConditionType_OR	2	OR
PtTriggerConditionType_EQUAL	3	EQUAL
PtTriggerConditionType_LESS	4	LESS
PtTriggerConditionType_GREATER	5	GREATER
PtTriggerConditionType_GREATER_EQUAL	6	GREATER_EQUAL
PtTriggerConditionType_LESS_EQUAL	7	LESS_EQUAL

`aimdk::protocol::PtTriggerConditionParam`

Parameters defining the stop condition for action command triggering

Field	Type	Description
name	string	Name of the stop condition
data	double	Threshold value for evaluation
state	double	Current state value of the condition

`aimdk::protocol::PtTriggerCondition`

Stop condition for action command triggering

Field	Type	Description
type	aimdk::protocol::PtTriggerConditionType	Type of stop condition (compound types: “AND”, “OR”; simple types: “EQUAL”, “LESS”, “GREATER”, “GREATER_EQUAL”, “LESS_EQUAL”)
param	aimdk::protocol::PtTriggerConditionParam	Parameters of the stop condition

`aimdk::protocol::PtTriggerConfig`

Configuration of stop conditions for action command triggering

Field	Type	Description
type	aimdk::protocol::PtTriggerConditionType	Type of stop condition (typically used to combine multiple PtTriggerCondition instances)
conditions	aimdk::protocol::PtTriggerCondition []	List of stop conditions

`aimdk::protocol::PtWiggleParam`

Pose oscillation parameters during search

Field	Type	Description
range	double	Oscillation amplitude
period	double	Oscillation period

`aimdk::protocol::ExternalForceCtrlParam`

External force admittance parameters

Field	Type	Description
external_force_axis	int32[]	Force-controlled axes
external_goal_wrench	double[]	Target force/torque
external_kp	double[]	Force gain
external_kv	double[]	Damping
external_ki	double[]	Force integral gain
external_max_integral	double[]	Integral limit

Internal force admittance parameters

Field	Type	Description
internal_force_axis	int32[]	Force-controlled axes
internal_goal_wrench	double[]	Target forces
internal_kp	double[]	Force gains
internal_kv	double[]	Damping
internal_ki	double[]	Force integral gains
internal_max_integral	double[]	Integral limits

`aimdk::protocol::AssemblyCtrlParam`

Assembly control task parameters

Field	Type	Description
external_admittance	aimdk::protocol::ExternalForceCtrlParam	External force admittance
internal_admittance	aimdk::protocol::InternalForceCtrlParam []	List of internal force admittance parameters

`aimdk::protocol::PtSlideCommonParam`

Common parameters for force-controlled search

Field	Type	Description
contact_axis	double[]	Contact axis
search_axis	double[]	Search axis
contact_force	double	Contact force
start_density	double	Initial search trajectory density
time_factor	int32	Controls search speed; smaller values mean faster motion
random_factor	double	Noise amplitude scaling factor
start_search_immediately	bool	Whether to start searching immediately upon entry

`aimdk::protocol::PtSlideSpiralParam`

Spiral search parameters

Field	Type	Description
search_radius	double	Maximum search radius
wiggle	aimdk::protocol::PtWiggleParam	Pose oscillation parameters during search
common	aimdk::protocol::PtSlideCommonParam	Common force-controlled search parameters
pt_slide_trigger	aimdk::protocol::PtTriggerConfig	Stop condition triggered by motion command
pt_control_param	aimdk::protocol::ExternalForceCtrlParam	Control parameters

`aimdk::protocol::PtDualSlideSpiralParam`

Dual-arm spiral search parameters

Field	Type	Description
search_radius	double	Maximum search radius
wiggle	aimdk::protocol::PtWiggleParam	Pose oscillation parameters during search
common	aimdk::protocol::PtSlideCommonParam	Common force-controlled search parameters
pt_dual_slide_trigger	aimdk::protocol::PtTriggerConfig	Trigger condition
pt_control_param	aimdk::protocol::AssemblyCtrlParam	Control parameters

Zigzag search parameters

Field	Type	Description
length	double	Search area length
height	double	Search area width
wiggle	aimdk::protocol::PtWiggleParam	Pose oscillation parameters during search
common	aimdk::protocol::PtSlideCommonParam	General force-controlled search parameters
pt_slide_trigger	aimdk::protocol::PtTriggerConfig	Stop condition triggered by motion command
pt_control_param	aimdk::protocol::ExternalForceCtrlParam	Control parameters

`aimdk::protocol::PtDualSlideZigzagParam`

Dual-arm zigzag search parameters

Field	Type	Description
length	double	Search area length
height	double	Search area width
wiggle	aimdk::protocol::PtWiggleParam	Pose oscillation parameters during search
common	aimdk::protocol::PtSlideCommonParam	General force-controlled search parameters
pt_dual_slide_trigger	aimdk::protocol::PtTriggerConfig	Stop condition triggered by motion command
pt_control_param	aimdk::protocol::AssemblyCtrlParam	Control parameters

`aimdk::protocol::PtSleeveParam`

Force-controlled insertion/extraction and trigger stop conditions

Field	Type	Description
insert_dir	double[]	Insertion direction (TCP)
alignment_dir	int32[]	Directions requiring compliance
max_contact_force	double	Maximum force limit for protection
insert_vel	double	Insertion velocity
deadband_scale	double	Deadband scaling factor during force control compliance
pt_sleeve_trigger	aimdk::protocol::PtTriggerConfig	Trigger condition
pt_control_param	aimdk::protocol::ExternalForceCtrlParam	Control parameters

Dual-arm force-controlled insertion/extraction and trigger stop conditions

Field	Type	Description
insert_dir	double[]	Insertion direction (in TCP)
alignment_dir	int32[]	Directions requiring compliance
max_contact_force	double	Maximum force limit for safety
insert_vel	double	Insertion velocity
deadband_scale	double	Deadband scaling factor for force control compliance
pt_dual_sleeve_trigger	aimdk::protocol::PtTriggerConfig	Trigger condition
pt_control_param	aimdk::protocol::AssemblyCtrlParam	Control parameters

`aimdk::protocol::PtContactParam`

Force-controlled contact and trigger stop conditions

Field	Type	Description
refer_coordinate	string	Contact reference frame (tcp/world)
freespace_vel	double	Velocity in free space
contact_dir	double[]	Contact direction
max_contact_force	double	Maximum contact force protection
pt_contact_trigger	aimdk::protocol::PtTriggerConfig	General force-controlled contact parameters

`aimdk::protocol::PrimitiveAction`

Force-controlled motion command

Field	Type	Description
name	string	Force-controlled action name: unused (not recommended)
type	aimdk::protocol::PrimitiveType	Force-controlled action type
param	google.protobuf.Any	Action parameters. The following are valid parameter types; refer to the definitions above for details: - PtSlideSpiralParam: spiral search parameters - PtSlideZigzagParam: zigzag search parameters - PtSleeveParam: compliance parameters - PtContactParam: contact parameters - PtDualSlideSpiralParam: dual-arm spiral search parameters - PtDualSlideZigzagParam: dual-arm zigzag search parameters - PtDualSleeveParam: dual-arm compliance parameters

Force Control Action Group

Field	Type	Description
actions	aimdk::protocol::PrimitiveAction []	

`aimdk::protocol::ForcePrimitiveRequest`

Field	Type	Description
header	aimdk::protocol::RequestHeader	
task	oneof {ForcePrimitive left} {ForcePrimitive right} {ForcePrimitive dual}	

`aimdk::protocol::McJointStatesResponse`

Motion Control Common State Response

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Response header
states	aimdk::protocol::JointState []	Joint states, including angle, velocity, and torque

`aimdk::protocol::McJointCommandsResponse`

Motion Control Common Command Response

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Response header
commands	aimdk::protocol::JointCommand []	Joint commands

`aimdk::protocol::McJointParamsResponse`

Motion Control Common Parameter Response

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Response header
params	aimdk::protocol::JointParam []	Joint parameters

`aimdk::protocol::McJointPosition`

Field	Type	Description
name	string	Joint name
position	double	Joint angle

`aimdk::protocol::McJointAnglesResponse`

Motion Control Common Angle Response

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Response header
joints	aimdk::protocol::McJointPosition []	Joint angles, unit: radians or meters

`aimdk::protocol::ArmStatus`

Arm Status

Field	Type	Description
error_code	bool	Error code
inpos	bool	In position
enable	bool	Enabled or not
protective_stop	bool	Collision protection
on_soft_limit	bool	Soft limit reached

`aimdk::protocol::ArmStatusRequest`

Arm Status Request

Field	Type	Description
header	aimdk::protocol::RequestHeader	Request header
arm_type	aimdk::protocol::ArmType	Arm type

Arm Status Request Response

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Response header
arm_status	aimdk::protocol::ArmStatus	Arm status info

`aimdk::protocol::GetTcpPositionResponse`

Get TCP Position Response

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Response header
tcp_position	aimdk::protocol::SE3Pose []	TCP pose

`aimdk::protocol::MotionGPTState`

Motion GPT State

Name	Number	Description
MotionGPTState_WAIT_MGPT_COMMAND	0	Waiting for command
MotionGPTState_MOVING_TO_FIRST_POINT	1	Moving to first point
MotionGPTState_ARRIVED_FIRST_POINT	2	Arrived at first point
MotionGPTState_MOVING_TO_FINAL_POINT	3	Moving to final point

Name	Number	Description
MotionGPTState_ARRIVED_FINAL_POINT	4	Arrived at final point
MotionGPTState_MGPT_FINISHED	5	Finished

`aimdk::protocol::MotionGPTStateRequest`

Motion GPT State Request

Field	Type	Description
header	aimdk::protocol::RequestHeader	Request header
state	aimdk::protocol::MotionGPTState	State

`aimdk::protocol::MotionGPTStateResponse`

Motion GPT State Response

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Response header
state	aimdk::protocol::MotionGPTState	State

`aimdk::protocol::McPlanningGroupAvailableResponse`

Motion Control Planning Group Availability Response

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Response header
groups	aimdk::protocol::McPlanningGroup []	Available groups

`aimdk::protocol::KineInverseRequest`

Inverse Kinematics Request

Field	Type	Description
header	aimdk::protocol::RequestHeader	Request header
type	aimdk::protocol::ArmType	Arm type
ref_angles	double[]	Joint angles, unit: radians or m
cartesian_pose	aimdk::protocol::SE3Pose	Target pose

Inverse Kinematics Response

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Response header

Field	Type	Description
type	aimdk::protocol::ArmType	Arm type
angles	double[]	Joint angles, unit: radians or m
error_code	int32	Error code
error_msg	string	Error message

`aimdk::protocol::KineForwardRequest`

Forward Kinematics Request

Field	Type	Description
header	aimdk::protocol::RequestHeader	Request header
type	aimdk::protocol::ArmType	Arm type
angles	double[]	Joint angles, unit: radians or m

`aimdk::protocol::KineForwardResponse`

Forward Kinematics Response

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Response header
type	aimdk::protocol::ArmType	Arm type
cartesian_pose	aimdk::protocol::SE3Pose	Target pose
error_code	int32	Error code
error_msg	string	Error message

`aimdk::protocol::FootKinematicsChannel`

Foot Kinematics Message

topic: /mc/kinematics/foot_kine

Field	Type	Description
header	aimdk::protocol::Header	Message header
data	aimdk::protocol::FootKinematicsData	Foot kinematics data

`aimdk::protocol::FootKinematics`

Inverse Kinematics Request

Field	Type	Description
type	aimdk::protocol::FootKinematics::Type	

Field	Type	Description
pose	aimdk::protocol::SE3Pose	Body frame, xyz
velocity	aimdk::protocol::SE3Velocity	Body frame, m/s
contact_phase	double	0.0~1.0: 0 = swing; 1 = contact

Name	Number	Description
UNKNOWN	0	
LEFT	1	
RIGHT	2	

`aimdk::protocol::FootKinematicsData`

Field	Type	Description
foot	aimdk::protocol::FootKinematics []	Typically contains two entries: one for the left foot and one for the right foot.

Motion Control Example Scripts

The motion control module provides various motion control capabilities. Example scripts demonstrating these features are included in the AimDK package under the `examples/mc` directory. You can refer to these scripts when developing motion control applications. Descriptions of some example scripts are listed below:

Name	Functionality	Notes
SetAction.py	Set Action	Robot actions must be switched according to the state machine. See the Action state machine description above.
actions.py	Get Current Action	Retrieves the current Action.
GetAvailableCommands.py	Get Action Switch Commands	Retrieves all available action-switching commands, which can be used to set an action.
GetJointState.py	Get Joint State	Retrieves state information for all robot joints (excluding head and hands), including name, position, velocity, etc.
GetJointAngle.py	Get Joint Angles	Retrieves angle information for all robot joints (excluding head and hands), including name and position.
GetHandState.py	Get Hand Joint State	Retrieves hand joint state information, including name, position, velocity, etc.
GetNeckState.py	Get Neck Joint State	Retrieves neck (head) joint state information, including name, position, velocity, etc.
GetTaskState.py	Query Task Status	Queries the status of a motion task by providing a task ID. Primarily used for `PlanningMove` task queries.
SetHandCommand.py	Set Hand Gesture	Sets finger positions and torques for both hands.
SetNeckCommand.py	Set Neck Joints	Sets target positions for the robot's neck joints.

Name	Functionality	Notes
PlanningMove.py	Plan Motion	Plans a motion trajectory. Requires target and reference positions as input.
walk.py	Robot Locomotion Command	Sets forward, lateral, and rotational motion commands for the robot.
arm.py	Direct Arm Control	Directly passes through arm joint commands without motion planning (requires data smoothing). Accepts 14 joint angles: the first 7 for the left arm, the last 7 for the right arm.
hand.py	Direct Finger Control	Controls all 12 finger joints (each thumb has two joints).
neck.py	Direct Neck Control	Controls the two neck joints directly.
joint_state.py	Subscribe to Upper Body Joint Data	Can subscribe to arm, neck, and hand joint data by modifying the topic name.
move_waist.py	Waist Motion	Controls waist movement, including lift, pitch, yaw, and roll.
move_waist_lift.py	Waist Lift	Controls vertical waist movement within the range of -0.15 m to +0.00 m.
SelectDanceType.py	Select Dance	Selects a dance routine for the robot to perform.

Motion Player Module

The Motion Player (MP) module is responsible for managing motion playback on the A2 robot. It provides an RPC interface to accept motion-related requests, such as commands to start playback, pause, stop, loop, etc.

This module runs on the x86 development board for both the Base and Flagship models, with its HTTP backend listening on port 56444.

RPC Interfaces

This module uses the RPC protocol to control motion playback and supports commands including motion playback, stop, pause, loop, etc. Below is the protocol specification for the `motion_player` module:

`SendMotionCommand` is the core RPC command of the `motion_player` module, used to control the robot's motion playback.

- **Motion Playback** : Starts playing a specific motion based on the given motion name (`motion_id`).
- **Pause and Reset** : The `cmd_pause` flag pauses the currently executing motion, while the `cmd_reset` flag resets the motion back to its initial pose.
- **Loop Playback** : The `cmd_repeat` flag enables the motion to loop continuously until stopped by an external command.
- **Auto Termination** : Depending on the `cmd_end` setting, the motion can either automatically reset or remain at the last frame after playback completes.

Interface Name	Description	Request Message Type	Response Message Type
`pb:/aimdk.protocol.MotionCommandService/SendMotionCommand`	Motion Playback Cmd	aimdk::protocol::MotionCommandRequest	aimdk::protocol::CommonResponse
`pb:/aimdk.protocol.MotionCommandService/GetMotionStatus`	Motion Status Query	aimdk::protocol::CommonRequest	aimdk::protocol::MotionCommandRespon

Interface Name	Description	Request Message Type	Response Message Type
`pb:/aimdk.protocol.MotionCommandService/EnableMotionPlayer`	Enable Motion Player	aimdk::protocol::CommonRequest	aimdk::protocol::CommonResponse
`pb:/aimdk.protocol.MotionCommandService/DisableMotionPlayer`	Disable Motion Player	aimdk::protocol::CommonRequest	aimdk::protocol::CommonResponse

Protobuf Message Types### `aimdk::protocol::MotionCommandRequest`

Field Name	Type	Description
header	aimdk::protocol::RequestHeader	Request header containing metadata such as a unique request identifier, timestamp, and request type.
motion_id	string	Name of the motion to execute. This is the absolute file path combined with the motion name.
duration_ms	int64	Maximum allowed execution time for the motion, in milliseconds.
cmd_pause	bool	Pause flag: `true` pauses the motion by stopping control signals sent to the robot.
cmd_reset	bool	Reset flag: `true` resets the current motion to its initial state; `false` means no reset.
cmd_repeat	bool	Deprecated flag. Retained for backward compatibility; do not use.
cmd_end	bool	Whether to automatically return to the initial pose after motion execution: `true` enables auto-reset.

`aimdk::protocol::MotionCommandResponse`

Field Name	Type	Description
header	aimdk::protocol::RequestHeader	Request header containing metadata such as a unique request identifier, timestamp, and request type.

Field Name	Type	Description
time_to_end_ms	int64	Remaining playback time (in milliseconds) of the currently playing motion.
status	aimdk::protocol::MotionCommandStatus	Current motion status (Idle, Start, Operating, Pause, or Stop).
is_neck_enable	bool	Indicates whether the motion_player is currently controlling head movement.
motion_id	string	Name of the last triggered motion.

`aimdk::protocol::MotionCommandStatus`

Name	Number	Description
MotionCommandStatus_IDLE	0	Idle
MotionCommandStatus_START	1	Starting
MotionCommandStatus_OPERATING	2	Operating
MotionCommandStatus_PAUSE	3	Paused
MotionCommandStatus_STOP	4	Stopped

Usage Example

Below is a bash script example that uses the `SendMotionCommand` interface for simple secondary development. The motion name is specified as a command-line argument when running the script. Users can implement custom logic in any similar way to achieve tailored motion playback functionality.

```
# Usage Examples
./send_motion_id.sh /agibot/data/resources/default/motion/motion_player/default/Speech10s/Speech10s
```

注意以上动作位于 x86 上，如需使用其他动作请查看 x86 上动作文件夹下的所有动作文件，自定义动作请放置在 `/agibot/data/var/rc/custom` 目录中（需使用遥操作相关功能才可以录制自定义动作）。

```
#!/bin/bash

if [ $# -eq 0 ]; then
    echo "arg error, need motion_id, ./send_motion_id.sh motion_id, example: "
    echo ./send_motion_id.sh /agibot/data/resources/default/motion/motion_player/default/演讲10s/演讲10s
    exit 0
fi

MC_ID="$1"

if [ "$MC_ID" == "停止动作" ]; then
    # 如果是“停止动作”，则 motion_id 为空，cmd_reset 设为 true
    DATA='{
      "motion_id": "",
      "duration_ms": 10000,
      "cmd_end": true,
      "cmd_pause": false,
      "cmd_reset": true
    }'
elif [ "$MC_ID" == "暂停播放器" ]; then
    # 如果是“暂停播放器”，则 motion_id 保持不变，cmd_pause 设为 true
    DATA='{
      "motion_id": "",
```

```
      "duration_ms": 10000,
      "cmd_end": true,
      "cmd_pause": true,
      "cmd_reset": false
    }'
elif [ "$MC_ID" == "下一个动作" ]; then
    # 如果是“下一个动作”，则 播放 list 中的下一个动作
    DATA='{
      "motion_id": "next_motion",
      "duration_ms": 10000,
      "cmd_end": true,
      "cmd_pause": false,
      "cmd_reset": false
    }'
else
    # 如果不是“停止动作”或“暂停播放器”，保持原有逻辑
    DATA='{
      "motion_id": "'"$MC_ID"'",
      "duration_ms": 10000,
      "cmd_end": true,
      "cmd_pause": false,
      "cmd_reset": false
    }'
fi

# 使用 curl 发送 POST 请求
curl -i \
    -H 'content-type:application/json' \
    -H 'timeout: 1000' \
    -X POST http://192.168.100.100:56444/rpc/aimdk.protocol.MotionCommandService/SendMotionCommand \
    -d "$DATA"

# 打印发送的数据
echo "$DATA"
```

Below is an example script for querying motion status using the `GetMotionStatus` interface.

```
# 调用示例
./get_motion_status.sh
```

```
#!/bin/bash

curl -i \
    -H 'content-type:application/json' \
    -H 'timeout: 1000' \
    -X POST http://192.168.100.100:56444/rpc/aimdk.protocol.MotionCommandService/GetMotionStatus \
    -d '{}'
```

Examples for enabling and disabling motion playback:

```
# 开启动作播放
curl -i \
    -H 'content-type:application/json' \
    -H 'timeout: 1000' \
    -X POST http://192.168.100.100:56444/rpc/aimdk.protocol.MotionCommandService/EnableMotionPlayer \
    -d '{}'

# 关闭动作播放
curl -i \
    -H 'content-type:application/json' \
    -H 'timeout: 1000' \
    -X POST http://192.168.100.100:56444/rpc/aimdk.protocol.MotionCommandService/DisableMotionPlayer \
    -d '{}'
```

Default Motion List

The default motions are located in the `/agibot/data/resources/default/motion/motion_player/default` directory.

Chinese Name	English Name
右手比心	Finger heart_right hand
左手握手_长	Handshake_left hand_Long
通用讲解动作_29s	General explanation actions_29s
右手碰拳_长	Bump fists_right hand_Long
右手加油	Come on_right hand
左转头	Turn head to the left
左手握手	Handshake_left hand
右手点赞	Like_right hand
右手比耶	Yeah_right hand
点点头	Nod head
通用讲解动作_11s	General explanation actions_11s
通用讲解动作_8s	General explanation actions_8s
打太极拳	Tai Chi
通用讲解动作_22s	General explanation actions_22s
通用讲解动作_25s	General explanation actions_25s
右手碰拳	Bump fists_right hand
生活化敬礼	Life-like salute
右手握手	Handshake_right hand
左手比耶	Yeah_left hand
左手点赞	Like_left hand
左手挥手	Wave hand_left hand
右手挥手	Wave hand_right hand
右转头	Turn head to the right
右手握手_长	Handshake_right hand_Long
左手碰拳_长	Bump fists_left hand_Long
方向_向右欢迎 Welcome to the right_10s	Welcome to the right_10s

Chinese Name	English Name
方向_向左欢迎	Welcome to the left_8s
左手碰拳	Bump fists_left hand
双手点赞	Like_both hands
右手点赞_长	Like_right hand_Long
左手点赞_长	Like_left hand_Long
双手挥手	Wave hand_both hands
左手加油	Come on_left hand
双手加油	Come on_both hands
双手比耶	Yeah_both hands
双手向下比心	Finger heart_both hands
双手擦眼泪	Wipe the tears
击掌_长	High-five_right hand_Long
右手NO	NO_right hand
右手OK	OK_right hand
跳舞	Dance
摆拍_右手胸前比耶	Pose_“V” sign in front of right chest
摆拍_左手胸前比耶	Pose_Both hands “V” sign
摆拍_右手胸前点赞	Pose_Thum-up in front of right chest
Pose_Wing Chun with both hands	Pose_Wing Chun with both hands
Posed_I’m a strongman	Posed_I’m a strongman
Pose_Right hand_01	Pose_Right hand_01
Pose_Right hand_02	Pose_Right hand_02
Pose_Right hand_03	Pose_Right hand_03
Pose_Right hand_04	Pose_Right hand_04
Pose_Right hand_05	Pose_Right hand_05
Number gesture “1”	Number gesture “1”
Number gesture “2”	Number gesture “2”
Number gesture “3”	Number gesture “3”

Chinese Name	English Name
Number gesture “4”	Number gesture “4”
Number gesture “5”	Number gesture “5”
Number gesture “6”	Number gesture “6”
Number gesture “7”	Number gesture “7”
Number gesture “8”	Number gesture “8”
Number gesture “9”	Number gesture “9”
Number gesture “10”	Number gesture “10”
Direction_upper right	Direction_upper right
Direction_lower right	Direction_lower right
Direction_upper left	Direction_upper left
Direction_lower left	Direction_lower left
Direction_point to forward_right hand	Direction_point to forward_right hand
Direction_point to forward_left hand	Direction_point to forward_left hand
Direction_point to back_right hand	Direction_point to back_right hand
Direction_point to back_left hand	Direction_point to back_left hand
Direction_points to current position	Direction_points to current position
Direction_point to the right	Direction_point to the right
Direction_point to the left	Direction_point to the left
Direction_Indicated on the right	Direction_Indicated on the right
Direction_Indicated on the left	Direction_Indicated on the left
Greeting at the beginning_12s	Greeting at the beginning_12s
Personal Introduction_7s	Personal Introduction_7s
Highlight 2 key points_12s	Highlight 2 key points_12s
Highlight 3 key points_11s	Highlight 3 key points_11s
Highlight 4 key points_16s	Highlight 4 key points_16s
Quickly highlight 5 key points_8s	Quickly highlight 5 key points_8s
Elaborate 1 key point_14s	Elaborate 1 key point_14s
Elaborate 2 key points_18s	Elaborate 2 key points_18s
Elaborate 3 key points_15s	Elaborate 3 key points_15s

Chinese Name	English Name
Briefly describe 6 key points_12s	Briefly describe 6 key points_12s
“Top-down” gesture_10s	“Top-down” gesture_10s
Emphasize the key point with fingers_09s	Emphasize the key point with fingers_09s
Make a fist and emphasize 2 key points_12s	Make a fist and emphasize 2 key points_12s
Both hands emphasize “the whole”_08s	Both hands emphasize “the whole”_08s
Right hand emphasize the “core”_10s	Right hand emphasize the “core”_10s
Summary after the lecture_9s	Summary after the lecture_9s
Acknowledgements at the end_7s	Acknowledgements at the end_7s
语音对话专属动作_02	Exclusive actions in voice chat _02
语音对话专属动作_03	Exclusive actions in voice chat _03
语音对话专属动作_04	Exclusive actions in voice chat _04
语音对话专属动作_05	Exclusive actions in voice chat _05
语音对话专属动作_06	Exclusive actions in voice chat _06
语音对话专属动作_07	Exclusive actions in voice chat _07
语音对话专属动作_08	Exclusive actions in voice chat _08
语音对话专属动作_09	Exclusive actions in voice chat _09
语音对话专属动作_10	Exclusive actions in voice chat _10
语音对话专属动作_11	Exclusive actions in voice chat _11
语音对话专属动作_12	Exclusive actions in voice chat _12
语音对话专属动作_13	Exclusive actions in voice chat _13
语音对话专属动作_14	Exclusive actions in voice chat _14
语音对话专属动作_15	Exclusive actions in voice chat _15
语音对话专属动作_16	Exclusive actions in voice chat _16
语音对话专属动作_17	Exclusive actions in voice chat _17
语音对话专属动作_18	Exclusive actions in voice chat _18
语音对话专属动作_19	Exclusive actions in voice chat _19
右手握手收回	Withdraw the right hand after handshake.

Sensor Data Module

[This module is exclusively for Flagship model users; Base model users do not need to refer to this section.]

This page contains documentation and protocols regarding sensor-related topics and services, providing system-level sensor data.

This module resides on the Orin development board, with the HTTP backend listening on port 56422.

The resolution and frame rate for each sensor data stream are as follows:

Sensor Type	Location	Data Stream	Resolution	Frame Rate
Fisheye Camera (Senyun ISX031C-H190XA)	Chest Left	Color Image (color)	640x480	30 FPS
Fisheye Camera (Senyun ISX031C-H190XA)	Chest Right	Color Image (color)	640x480	30 FPS
RGB-D Camera (Realsense D415)	Head Front	Color Image (color)	1280x720	15 FPS
RGB-D Camera (Realsense D415)	Head Front	Depth Image (depth)	1280x720	15 FPS
RGB-D Camera (Orbbec)	Waist Front	Color Image (color)	640x480	10 FPS
RGB-D Camera (Orbbec)	Waist Front	Depth Image (depth)	640x400	10 FPS
LiDAR (Mid360)	Neck	Point Cloud (pcd_option)	-	10 FPS
LiDAR (Mid360)	Neck	IMU Data (imu_option)	-	200 FPS
IMU (YISENSE YI 320)	Waist Center	IMU Data (imu_option)	-	1000 FPS
Interaction Camera (Senyun ISX031C-H100F1A)	Waist Center	Color Image (color)	1920x1536	30 FPS

Channel Interface

Topic Name	Topic Description	Publish or Subscribe	Message Type	Notes	Communication Backend
`/aima/hal/fish_eye_camera/chest_left/color`	RGB image from chest-left fisheye camera	Publish	sensor_msgs::msg::Image		ros2
`/aima/hal/fish_eye_camera/chest_right/color`	RGB image from chest-right fisheye camera	Publish	sensor_msgs::msg::Image		ros2
`/aima/hal/rgbd_camera/head_front/color`	RGB image from head-front camera	Publish	sensor_msgs::msg::Image		ros2
`/aima/hal/rgbd_camera/head_front/depth`	Depth image from head-front camera	Publish	sensor_msgs::msg::Image		ros2
`/aima/hal/rgbd_camera/waist_front/color`	RGB image from waist-front camera	Publish	sensor_msgs::msg::Image		ros2

Topic Name	Topic Description	Publish or Subscribe	Message Type	Notes	Communication Backend
`/aima/hal/rgbd_camera/waist_front/depth`	Depth image from waist-front camera	Publish	sensor_msgs::msg::Image		ros2
`/aima/hal/lidar/neck/pointcloud`	Point cloud from neck LiDAR	Publish	sensor_msgs::msg::PointCloud2		ros2
`/body_drive/imu/data`	Waist-center IMU	Publish	sensor_msgs::msg::Imu		ros2
`/aima/hal/camera/interactive/color`	RGB image from chest-center interactive camera	Publish	sensor_msgs::msg::Image	Supported only on P1 models. This is not a standard ROS 2 topic; it uses shared memory for communication and requires AimRT (version V1.0.0 or later) to access. Relevant examples are provided. The model type can be identified by checking the file `/etc/bsp_version` : if the file exists, it is a P1 model; otherwise, it is a T3 model.	iceoryx

RPC Interfaces

Interface Name	Interface Description	Request Message Type	Response Message Type
`pb:/aimdk.protocol.CamerasIntrinsicService/GetCamerasIntrinsic`	Get camera intrinsic parameters	aimdk::protocol::GetCamerasIntrinsicRequest	aimdk::protocol::GetCamerasIntrin

Interface Name	Interface Description	Request Message Type	Response Message Type

Protobuf Message Types

`aimdk::protocol::DistortionType`
Distortion type

Name	Number	Description
DistortionType_UNDEFINED	0	Undefined
DistortionType_MODIFIED_BROWN_CONRADY	1	Modified Brown-Conrady
DistortionType_INVERSE_BROWN_CONRADY	2	Inverse Brown-Conrady
DistortionType_FTHETA	3	F-theta
DistortionType_BROWN_CONRADY	4	Brown-Conrady
DistortionType_KANNALA_BRANDT4	5	Kannala-Brandt4
DistortionType_COUNT	6	Count

`aimdk::protocol::StreamType`
Stream type

Name	Number	Description
StreamType_UNDEFINED	0	Undefined
StreamType_DEPTH	1	Depth
StreamType_COLOR	2	Color
StreamType_IR	3	Infrared

`aimdk::protocol::CameraIntrinsicInfo`
Camera intrinsic information

Field	Type	Description
id	string	Camera ID
name	string	Camera name
stream_type	aimdk::protocol::StreamType	Stream type
width	int32	Image width
height	int32	Image height

Field	Type	Description
ppx	float	Principal point X
ppy	float	Principal point Y
fx	float	Focal length X
fy	float	Focal length Y
model	aimdk::protocol::DistortionType	Distortion model
coeffs	float[]	Distortion coefficients

`aimdk::protocol::GetCamerasIntrinsicRequest`
Request for retrieving camera intrinsic parameters.

Field	Type	Description
header	aimdk::protocol::RequestHeader	Request header

`aimdk::protocol::GetCamerasIntrinsicResponse`
Response containing the retrieved camera intrinsic parameters.

Field	Type	Description
info	aimdk::protocol::CameraIntrinsicInfo []	List of camera intrinsic info

Calibration Parameters

The directory storing the camera intrinsic and extrinsic calibration results (located on the Orin development board):

```
/agibot/data/param/calibration/
```

The folder contains the following files:

- **Extrinsic parameters** :
  - `extrinsic_baselink_T_chest_left.txt` : Extrinsic from `base_link` to the left fisheye camera (x, y, z, qw, qx, qy, qz)
  - `extrinsic_chest_right_T_chest_left.txt` : Extrinsic from the left camera to the right camera (x, y, z, qw, qx, qy, qz)
  - `extrinsic_baselink_T_head_front.txt` : Extrinsic from `base_link` to the head camera (x, y, z, qw, qx, qy, qz)
  - `estrinsic_waist_front_T_baselink.txt` : Extrinsic from `base_link` to the waist camera (x, y, z, qw, qx, qy, qz)
  - `estrinsic_baselink_T_lidar.txt` : Extrinsic from LiDAR to `base_link` (x, y, z, qw, qx, qy, qz)
  - `gravity_T_imu.txt` : Rotation from IMU to `base_link` (qw, qx, qy, qz)
- **Intrinsic parameters** :
  - `intrinsic_chest_left.yaml` : Intrinsic parameters of the left fisheye camera
  - `intrinsic_chest_right.yaml` : Intrinsic parameters of the right fisheye camera
  - `intrinsic_head_front.yaml` : Intrinsic parameters of the head camera

- `intrinsic_wasit_front.yaml` : Intrinsic parameters of the waist camera

Reading Camera Intrinsic Parameters

It is recommended to obtain intrinsic parameters via the interface `pb:/aimdk.protocol.CamerasIntrinsicService/GetCamerasIntrinsic` , which retrieves more accurate intrinsic parameters directly from the camera SDK. Example usage:

```
# Retrieve camera intrinsic parameters. On the P1 model, fisheye and interactive cameras are managed
separately by the camera module, so the HTTP port should be changed to 59324
# The robot model can be determined by checking the file `cat /etc/bsp_version`. If this file exists, it
is a P1 model; otherwise, it is a T3 model.

curl --request POST \
  --url http://192.168.100.110:56422/rpc/aimdk.protocol.CamerasIntrinsicService/GetCamerasIntrinsic \
  --header 'content-type: application/json' \
  --data '{}'
```

针孔相机内参读取

针孔相机内参（intrinsic_head_front.yaml）示例：

```
%YAML:1.0
---
model_type: PINHOLE
camera_name: ""
image_width: 1280
image_height: 720
distortion_parameters:
  k1: 8.4721332081814996e-02
  k2: -2.2328724059363472e-01
  p1: 1.4980714426687237e-02
  p2: 5.8573350999184884e-03
projection_parameters:
  fx: 9.4067882179688206e+02
  fy: 9.3408163057525098e+02
  cx: 6.3075913699847933e+02
  cy: 3.7669033617329160e+02
```

`fx` , `fy` , `cx` , `cy` correspond to the camera intrinsic matrix, and `k1` , `k2` , `p1` , `p2` correspond to distortion coefficients.

The corresponding reading function is as follows:

```
// 针孔相机内参读取函数 (Pinhole camera intrinsic reading function)
bool ReadFromYamlFile(const std::string& filename)
{
    cv::FileStorage fs(filename, cv::FileStorage::READ);

    if (!fs.isOpened())
    {
        return false;
    }

    if (!fs["model_type"].isNone())
    {
        std::string sModelType;
        fs["model_type"] >> sModelType;

        if (sModelType.compare("PINHOLE") != 0)
        {
            return false;
        }
    }

    std::string model_type_ = "PINHOLE";
    std::string camera_name_;
```

```
    fs["camera_name"] >> camera_name_;
    int image_width_  = static_cast<int>(fs["image_width"]);
    int image_height_ = static_cast<int>(fs["image_height"]);

    cv::FileNode n = fs["distortion_parameters"];
    double k1_            = static_cast<double>(n["k1"]);
    double k2_            = static_cast<double>(n["k2"]);
    double p1_            = static_cast<double>(n["p1"]);
    double p2_            = static_cast<double>(n["p2"]);

    n   = fs["projection_parameters"];
    double fx_ = static_cast<double>(n["fx"]);
    double fy_ = static_cast<double>(n["fy"]);
    double cx_ = static_cast<double>(n["cx"]);
    double cy_ = static_cast<double>(n["cy"]);

    return true;
}
```

Reading Intrinsic Parameters of Fisheye Camera
Example of fisheye camera intrinsic parameters (`intrinsic_chest_left.yaml`):

```
%YAML:1.0
---
model_type: KANNALA_BRANDT
camera_name: ""
image_width: 640
image_height: 480
projection_parameters:
  k2: 4.0154631284881101e-02
  k3: -1.9502762632935614e-02
  k4: 8.9682432239682577e-04
  k5: -5.0758203706675745e-04
  mu: 1.9728857021424918e+02
  mv: 1.9697037865238804e+02
  u0: 3.1897257040247649e+02
  v0: 2.4312745081143589e+02
```

`u0` , `v0` , `mu` , `mv` correspond to the camera intrinsic matrix, and `k2` , `k3` , `k4` , `k5` correspond to distortion coefficients.

The corresponding reading function is as follows:

```
// 鱼眼相机内参读取函数 (Fisheye camera intrinsic reading function)
bool ReadFromYamlFile(const std::string& filename)
{
    cv::FileStorage fs(filename, cv::FileStorage::READ);

    if (!fs.isOpened())
    {
        return false;
    }

    if (!fs["model_type"].isNone())
    {
        std::string sModelType;
        fs["model_type"] >> sModelType;

        if (sModelType.compare("KANNALA_BRANDT") != 0)
        {
            return false;
        }
    }

    std::string model_type_ = "KANNALA_BRANDT";
    std::string camera_name_;
    fs["camera_name"] >> camera_name_;
    int image_width_  = static_cast<int>(fs["image_width"]);
```

```
    int image_height_ = static_cast<int>(fs["image_height"]);

    cv::FileNode n = fs["projection_parameters"];
    double k2_            = static_cast<double>(n["k2"]);
    double k3_            = static_cast<double>(n["k3"]);
    double k4_            = static_cast<double>(n["k4"]);
    double k5_            = static_cast<double>(n["k5"]);
    double mu_            = static_cast<double>(n["mu"]);
    double mv_            = static_cast<double>(n["mv"]);
    double u0_            = static_cast<double>(n["u0"]);
    double v0_            = static_cast<double>(n["v0"]);

    return true;
}
```

Camera Extrinsic Parameter Reading
The data format for camera extrinsic parameters is:

```
x, y, z, qw, qx, qy, qz
```

The reading function is as follows:

```
// 外参读取函数 (Extrinsic parameter reading function)
bool ReadPoseFromFile(const std::string& file_name, SE3d& pose)
{
    if (!std::filesystem::exists(file_name))
    {
        LOG(WARNING) << "pose file don't exists!" << file_name;
        return false;
    }

    std::ifstream     pose_ifs(file_name);
    Eigen::Quaternion rot   = pose.unit_quaternion();
    Vec3d             trans = pose.translation();
    pose_ifs >> trans.x() >> trans.y() >> trans.z() >> rot.w() >> rot.x() >> rot.y() >> rot.z();
    pose = SE3d(rot, trans);
    pose_ifs.close();
    return true;
}
```

Health Diagnosis Module

This page contains the documentation and protocols for topics and services included in the Health Diagnosis System (HDS) module, supporting system health diagnosis functionalities.

This module runs on the Orin development board for the Flagship model and on the x86 development board for the Base model. The HTTP backend listens on port 50587 in both cases.

Channel Interface

Topic Name	Topic Description	Subscribe or Publish	Message Type	Notes	Communication Backend
`/aima/hds/exception`	Reporting interface for module exception codes	Subscribe	aimdk::protocol::ModuleExceptionChannel		ros2, http

RPC Interface

Interface Name	Interface Description	Request Message Type	Response Message Type	Notes
`pb:/aimdk.protocol.HDSService/GetAlertList`	Retrieve the current alert list	aimdk::protocol::GetAlertListReq	aimdk::protocol::GetAlertListRsp	
`pb:/aimdk.protocol.HDSService/ClearAlert`	Attempt to clear current alerts	aimdk::protocol::ClearAlertReq	aimdk::protocol::ClearAlertRsp	Clearing fai alerts contin be rep afterward
`pb:/aimdk.protocol.HDSService/GetTotalAlertList`	Retrieve all alerts in the system (including history)	aimdk::protocol::GetTotalAlertListReq	aimdk::protocol::GetTotalAlertListRsp	Requires `start_times` and `limit_` for paginatio
`pb:/aimdk.protocol.HDSService/GetAlertCount`	Get the total number of currently active alerts	aimdk::protocol::CommonRequest	aimdk::protocol::GetAlertCountRsp	
`pb:/aimdk.protocol.HDSService/GetExceptionEvent`	Retrieve exception events in the system	aimdk::protocol::GetExceptionEventReq	aimdk::protocol::GetExceptionEventRsp	Requires `start_times` and `limit_` for paginatio
`pb:/aimdk.protocol.HDSService/SetShieldAlertLevel`	Suppress all alerts at a specified level and below	aimdk::protocol::SetShieldAlertLevelReq	aimdk::protocol::CommonResponse	Use with ca generally unnecessary except called by SM

Alerts are clusters of Exceptions. An Alert represents a broad category of faults, while an Exception refers to a specific fault instance. For example, “relocalization failure” is an Alert, which may correspond to many different specific failure reasons—each being a distinct Exception. AimMaster typically displays Alerts, as Exceptions can be numerous and less intuitive to interpret.

If continuous monitoring of the robot’s abnormal status is required, it is recommended to periodically query the current alert list (at a low frequency, e.g., ≤1 Hz).### `aimdk::protocol::AlertImpactName`

Modules affected by alerts

Name	Number	Description
AlertImpactModule_UNDEFINED	0	
AlertImpactModule_Task_Start	1	Task Start
AlertImpactModule_Task_Run	2	Task Execution
AlertImpactModule_Autonomous_Move	3	Autonomous Movement
AlertImpactModule_Remote_Control_Move	4	Remote-Control Movement

Name	Number	Description
AlertImpactModule_Single_Arm_Action	5	Single Arm Action
AlertImpactModule_Dual_Arm_Action	6	Dual Arm Action
AlertImpactModule_Head_Action	7	Head Action
AlertImpactModule_Waist_Action	8	Waist Action
AlertImpactModule_Leg_Action	9	Leg Action
AlertImpactModule_Mapping_And_Loc	10	Mapping and Localization
AlertImpactModule_Obstacle_Avoidance	11	Obstacle Avoidance
AlertImpactModule_Motion_Control	12	Motion Control
AlertImpactModule_Skill	13	Skill

`aimdk::protocol::Fallback`

Degradation information

Field	Type	Description
timestamp	aimdk::protocol::Timestamp	Timestamp when this degradation decision was made
code	uint32	Degradation policy code, mapping to a set of degradation actions
alert_level	aimdk::protocol::AlertLevel	Alert level, determined by the most urgent degradation policy
warn_policy	uint32	Interactive alert type

`aimdk::protocol::FallbackChannel`

Degradation information message

topic: /aima/hds/fallback

Field	Type	Description
header	aimdk::protocol::Header	Message header
data	aimdk::protocol::Fallback	Data content

`aimdk::protocol::CpuCoreUsage`

Per-core CPU monitoring metrics

Field	Type	Description
cpuid	uint32	CPU ID, consistent with the index in cpu_info list
usage	uint32	CPU core utilization percentage
freq_mhz	uint32	Average CPU frequency, in MHz

Field	Type	Description
temp	uint32	CPU core temperature, in °C
usage_usr_space	uint32	CPU utilization - user space
usage_sys_space	uint32	CPU utilization - kernel space
usage_idle	uint32	CPU utilization - idle

`aimdk::protocol::CpuUsage`

CPU monitoring metrics

Field	Type	Description
cpu_infos	aimdk::protocol::CpuCoreUsage []	
avg_freq_mhz	uint32	Average CPU frequency, MHz
interrupts	uint32	Number of interrupts

`aimdk::protocol::MemoryUsage`

Memory usage metrics

Field	Type	Description
usage	float	Memory usage percentage
swap_usage	float	Swap space usage percentage
total_mem_mb	uint64	Total memory size in MB
available_mem_mb	uint64	Available memory size

Disk partition usage metrics

Field	Type	Description
partition_name	string	Partition name
read_mbs	float	Partition read speed, MB/s
write_mbs	float	Partition write speed, MB/s
usage	float	Partition usage ratio
read_times	uint64	Read requests per second
write_times	uint64	Write requests per second
read_await_us_total	uint64	Total read latency (microseconds)
write_await_us_total	uint64	Total write latency (microseconds)
read_await_us_max	uint64	Maximum read latency (microseconds)

Field	Type	Description
write_await_us_max	uint64	Maximum write latency (microseconds)

`aimdk::protocol::DiskUsage`

Disk usage metrics

Field	Type	Description
partition_infos	aimdk::protocol::PartitionUsage []	Partition usage information
total_read_mbs	float	Total read speed, MB/s
total_write_mbs	float	Total write speed, MB/s
max_partition_read_await	uint64	Maximum read latency (microseconds)
max_partition_write_await	uint64	Maximum write latency (microseconds)

`aimdk::protocol::NetworkIterfaceUsage`

Field	Type	Description
nic_name	string	Network interface name
upload_speed_mbs	float	Upload speed, MB/s
download_speed_mbs	float	Download speed, MB/s

`aimdk::protocol::NetworkUsage`

Network usage metrics

Field	Type	Description
nic_infos	aimdk::protocol::NetworkIterfaceUsage []	Network interface info

`aimdk::protocol::BatteryUsage`

Battery status metrics

Field	Type	Description
percent	float	Remaining battery percentage
voltage	float	Battery voltage, mV
current	float	Battery current, mA
temp	float	Battery temperature, °C
is_plugin	bool	Whether external power is plugged in

`aimdk::protocol::AppProcInfo`

Field	Type	Description
app_name	string	Application name (note: not process name)
cpu_usage	float	CPU usage of the application
mem_usage	float	Memory usage of the application, in percentage
pid	uint64	Process ID
thread_count	uint32	Number of threads used
fd_count	uint32	Number of file descriptors used

`aimdk::protocol::ProcessList`

Application process list

Field	Type	Description
app_infos	aimdk::protocol::AppProcInfo []	Process list info

`aimdk::protocol::SCHED_POLICY`

Name	Number	Description
TYPE_SCHED_NORMAL	0	
TYPE_SCHED_FIFO	1	
TYPE_SCHED_RR	2	
TYPE_SCHED_BATCH	3	
TYPE_SCHED_IDLE	5	
TYPE_SCHED_DEADLINE	6	

Field	Type	Description
app_name	string	Application name (note: not the process name)
cpu_usage	float	CPU usage of the application
mem_usage	float	Memory usage percentage of the application
mem_usage_kb	float	Memory usage of the application, in KB
pid	uint64	Process ID (PID)
thread_count	uint32	Number of threads used
cpu_sched_policy	aimdk::protocol::SCHED_POLICY	Process scheduling policy
cpu_sched_priority	uint32	Process scheduling priority

`aimdk::protocol::AppHeartBeatList`

Field	Type	Description
app_hearbeats	aimdk::protocol::AppHeartBeatInfo []	

`aimdk::protocol::GpuProcInfo`

Field	Type	Description
app_name	string	Application name (note: not the process name)
pid	uint64	Process ID
gpu_mem_usage_mb	float	GPU memory usage, in MB

`aimdk::protocol::GPUPowerMode`

GPU power mode for NVIDIA GPUs

Name	Number	Description
GPUPowerMode_MAXN	0	Maximum performance (currently 30W)
GPUPowerMode_15W	1	
GPUPowerMode_10W	2	
GPUPowerMode_5W	3	
GPUPowerMode_UNKNOWN	99	

`aimdk::protocol::GpuDeviceUsage`

Field	Type	Description
app_gpu_infos	aimdk::protocol::GpuProcInfo []	List of GPU resource usage per application
gpu_usage	float	System-wide GPU utilization
gpu_mem_use_percent	float	System-wide GPU memory utilization percentage
mem_bandwidth_useage	float	GPU memory bandwidth utilization
temp	uint32	GPU temperature
power_mode	aimdk::protocol::GPUPowerMode	Power mode

`aimdk::protocol::GpuUsage`

Field	Type	Description
gpu_infos	aimdk::protocol::GpuDeviceUsage []	

Name	Number	Description
SystemMsgType_NONE	0	
SystemMsgType_CPU_USAGE	1	
SystemMsgType_MEMORY_USAGE	2	
SystemMsgType_DISK_USAGE	3	
SystemMsgType_NETWORK_USAGE	4	
SystemMsgType_BATTERY_USAGE	5	
SystemMsgType_PROCESS_LIST	6	
SystemMsgType_HEARTBEAT_LIST	7	
SystemMsgType_GPU_USAGE	8	

`aimdk::protocol::SystemStatusChannel`

Reported every second to `/aima/hds/system_status_monitor_{orin/x86}`

Field	Type	Description
timestamp	uint64	Timestamp in milliseconds
cpu_usage	aimdk::protocol::CpuUsage	
memory_usage	aimdk::protocol::MemoryUsage	
disk_usage	aimdk::protocol::DiskUsage	
network_usage	aimdk::protocol::NetworkUsage	
battery_usage	aimdk::protocol::BatteryUsage	
process_list	aimdk::protocol::ProcessList	
heartbeat_list	aimdk::protocol::AppHeartBeatList	
gpu_usage	aimdk::protocol::GpuUsage	
gist_msg_type	aimdk::protocol::SystemMsgType	Each plugin reports its own message fragment via SystemStatusChannel; the aggregated system message has no gist
gist_seq	uint64	

Exception Code Blocking Request Body

Field	Type	Description
header	aimdk::protocol::RequestHeader	
code	string	Exception code

Field	Type	Description
is_permanent	bool	Whether to block permanently (`true` : permanent, `false` : temporary; expires after program restart). Permanent blocking by default.
is_blocked	bool	Whether to block

`aimdk::protocol::GetBlockedExceptionCodeListRsp`
Response Body for Retrieving Blocked Exception Code List

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Response header
code_list	string[]	List of blocked exception codes

`aimdk::protocol::SetDebugModeReq`
Debug Mode Setting Request Body

Field	Type	Description
header	aimdk::protocol::RequestHeader	
is_debug	bool	Whether in debug mode

`aimdk::protocol::SetShieldAlertLevelReq`
Alert Level Shielding Request Body

Field	Type	Description
header	aimdk::protocol::RequestHeader	
level	aimdk::protocol::AlertLevel	

`aimdk::protocol::GetAlertCountRsp`
Response Body for Querying Active Alert Events in the System

Field	Type	Description
header	aimdk::protocol::ResponseHeader	
total_count	int32	Total alert count
active_count	int32	Active alert count
cleared_count	int32	Cleared alert count

`aimdk::protocol::ClearAlertReq`
Alert Clearing Request Body

Field	Type	Description
header	aimdk::protocol::RequestHeader	
id	string	ID of the alert to clear

Field	Type	Description
data	string	Processing result

`aimdk::protocol::ClearAlertRsp`

Clear alert response body

Field	Type	Description
header	aimdk::protocol::ResponseHeader	
data	aimdk::protocol::ClearAlertRspData	

`aimdk::protocol::GetAlertListReq`

Get current alert information request body

Field	Type	Description
header	aimdk::protocol::RequestHeader	

`aimdk::protocol::GetAlertListRspData`

Field	Type	Description
alerts	aimdk::protocol::Alert []	

`aimdk::protocol::GetAlertListRsp`

Get current alert information response body

Field	Type	Description
header	aimdk::protocol::ResponseHeader	
data	aimdk::protocol::GetAlertListRspData	List of all current alerts

`aimdk::protocol::GetTotalAlertListReq`

Get all alert information request body

Field	Type	Description
header	aimdk::protocol::RequestHeader	
start_timestamp	uint64	Start timestamp for querying alerts
limit_size	int32	Maximum number of alerts to return

`aimdk::protocol::GetTotalAlertListRspData`

Field	Type	Description
alerts	aimdk::protocol::Alert []	

`aimdk::protocol::GetTotalAlertListRsp`

Get all alert information response body

Field	Type	Description
header	aimdk::protocol::ResponseHeader	
data	aimdk::protocol::GetTotalAlertListRspData	

Request body for querying specified exception events

Field	Type	Description
header	aimdk::protocol::RequestHeader	
start_timestamp	uint64	Start timestamp for querying alerts
limit_size	int32	Maximum number of results to return
query	string	Query condition

`aimdk::protocol::ExceptionEvent`

Exception information displayed to aim-master

Field	Type	Description
timestamp	aimdk::protocol::Timestamp	Timestamp when the exception occurred
type	aimdk::protocol::ExceptionEvent::Type	
code	string	Exception code
module_id	string	Module ID corresponding to the exception code
sub_code	string	Sub-code of the module corresponding to the exception
module_name	string	Name of the module reporting the exception
id	string	Exception ID
enum_name	string	Exception description (may be empty)
info	string	Exception details (may be empty)
alert_list	string	Alerts generated by this exception code
is_blocked	bool	Whether blocked (true: blocked, false: unblocked)

`aimdk::protocol::ExceptionEvent::Type`

Name	Number	Description
UNKNOWN	0	
Appear	1	
Disappear	2	

`aimdk::protocol::GetExceptionEventRspData`

Field	Type	Description
event_list	aimdk::protocol::ExceptionEvent []	

`aimdk::protocol::GetExceptionEventRsp`

Response body for querying specified exception events

Field	Type	Description
header	aimdk::protocol::ResponseHeader	
data	aimdk::protocol::GetExceptionEventRspData	

Alert topic data (`/aima/hds/alert`) is generated and published by the alert system.

This topic is sent when an alert is triggered or cleared.

Field	Type	Description
alert	aimdk::protocol::Alert	
alert_count	uint32	Number of active alerts in the system

`aimdk::protocol::AlertState`

Alert state

There are five alert states: undefined, active, cleared, exception changed, and recovering.

Name	Number	Description
AlertState_UNDEFINED	0	Undefined (default)
AlertState_ACTIVE	1	Active
AlertState_CLEARED	2	Cleared
AlertState_ExceptionChanged	3	Exception changed
AlertState_Recovering	4	Recovering

`aimdk::protocol::AlertLevel`

Alert level (health status)

Name	Number	Description
AlertLevel_UNDEFINED	0	No alert
AlertLevel_FATAL	1	Fatal alert
AlertLevel_SERIOUS	2	Serious alert
AlertLevel_WARNING	3	Warning alert
AlertLevel_HIDDEN_DANGERS	4	Hidden risks exist in the system
AlertLevel_STATUS	5	Status-type alerts exist in system

`aimdk::protocol::AlertSolution`

Alert solution

Field	Type	Description
type	string	
content	string	

`aimdk::protocol::AlertShowType`

Alert display type

Name	Number	Description
AlertShowType_Normal	0	
AlertShowType_Toast	1	

`aimdk::protocol::Alert`

Alert information

Field	Type	Description
id	string	Alert ID
appeared_timestamp	uint64	Timestamp when the alert appeared
disappeared_timestamp	uint64	Timestamp when the alert disappeared
alert_code	string	Alert code
state	aimdk::protocol::AlertState	Alert state
exception_list	aimdk::protocol::ModuleException []	List of exceptions in this alert
description	string	Alert description
level	aimdk::protocol::AlertLevel	Alert level
manual_clear	bool	Whether manual clearing is supported

Field	Type	Description
show_type	aimdk::protocol::AlertShowType	Alert display type
solution_list	aimdk::protocol::AlertSolution []	List of alert solutions

`aimdk::protocol::ExceptionType`
Exception type

Name	Number	Description
ExceptionType_Normal	0	Normal type; this exception must be reported periodically
ExceptionType_Trigger_Appear	1	Triggered exception (reported once when appearing); set exception state to “appeared”
ExceptionType_Trigger_Disappear	2	Triggered exception (reported once when disappearing); set exception state to “disappeared”

Module exception information, reported by individual modules.

Field	Type	Description
timestamp	aimdk::protocol::Timestamp	Timestamp when the exception occurred
type	aimdk::protocol::ExceptionType	Exception type
module_id	uint32	Module ID
code	uint32	Module-specific exception code
info	string	Exception details
module_name	string	Module name

`aimdk::protocol::HealthStatusChannel`
System health status topic:

@deprecated This interface has been removed after being deprecated in v0.7 for TE and SM.

Field	Type	Description
header	aimdk::protocol::Header	Header containing timestamp
alert_level	aimdk::protocol::AlertLevel	Alert severity level
alert_impact_range	aimdk::protocol::AlertImpactName []	Affected scope of the alert
demotion_id	uint32	Degradation strategy ID

`aimdk::protocol::ModuleExceptionChannel`
Module exception reporting channel topic: (`/aima/hds/exception`). Subscribed by the alert system.

Field	Type	Description
timestamp	aimdk::protocol::Timestamp	Timestamp when this message was published
exception_list	aimdk::protocol::ModuleException []	List of exceptions included in this message

Usage Examples

HTTP Curl examples are provided below. If calling from a computer other than ORIN, replace `192.168.100.110` with the actual robot IP address. No replacement is needed if used directly on ORIN.

Get current alert list

```
curl --location --request POST 'http://192.168.100.110:50587/rpc/aimdk.protocol.HDSService/
GetAlertList' \
--header 'Content-Type: application/json' \
--data-raw '{}'
```

Attempt to clear current alerts
Before clearing alerts, you need to obtain the alert ID via the `GetAlertList` API. This endpoint attempts to clear alerts; however, if the alert condition persists and continues to be reported, the clearing operation will fail.

```
curl --location --request POST 'http://192.168.100.110:50587/rpc/aimdk.protocol.HDSService/ClearAlert' \
--header 'Content-Type: application/json' \
--data-raw '{"id":"d0062056-1d4f-43ec-a36c-18fc149a5781"}'
```

Get all alerts in the system (including historical ones)

```
curl --location --request POST 'http://192.168.100.110:50587/rpc/aimdk.protocol.HDSService/
GetTotalAlertList' \
--header 'Content-Type: application/json' \
--data-raw '{}'
```

Get the total count of currently active alerts in the system

```
curl --location --request POST 'http://192.168.100.110:50587/rpc/aimdk.protocol.HDSService/
GetAlertCount' \
--header 'Content-Type: application/json' \
--data-raw '{}'
```

Get fault events in the system

```
curl --location --request POST 'http://192.168.100.110:50587/rpc/aimdk.protocol.HDSService/
GetExceptionEvent' \
--header 'Content-Type: application/json' \
--data-raw '{}'
```

Block fault codes
If a specific fault code is falsely triggered in the system, you can use this endpoint to block it. Once blocked, the fault code will no longer be reported to upper layers.

Parameters:

- `code` : The fault code, obtainable via the `GetExceptionEvent` API.

- `is_permanent` : Whether to permanently block the fault code (`true` : permanent block; `false` : temporary block, which resets after program restart).
- `is_block` : Whether to block (`true` : block; `false` : unblock).

```
curl --location --request POST 'http://192.168.100.110:50587/rpc/aimdk.protocol.HDSService/
SetShieldExceptionCode' \
--header 'Content-Type: application/json' \
--data-raw '{
    "code": "0x5100000005",
    "is_permanent": true,
    "is_block": true
}'
```

Remote Control Module

This document describes how to control the robot to play motions and emoticons via the Remote Control (RC) module. Note that motion playback through this module only supports simple playback. For operations such as retrieving playback status and resetting, please refer to the Motion Player Module .

The HTTP backend of this module listens on port `59001` .

RPC Interfaces

Interface Name	Description	Request Message Type	Response Message Type
`pb:/aimdk.protocol.RcMotionPlayerService/GetMotionList`	Get motion list	aimdk::protocol::CommonRequest	aimdk::protocol::RcMotionListRespon
`pb:/aimdk.protocol.RcMotionPlayerService/PlayerMotion`	Play motion	aimdk::protocol::RcPlayerMotionRequest	aimdk::protocol::CommonResponse
`pb:/aimdk.protocol.RcEmoticonPlayerService/GetEmoticonList`	Get emoticon list	aimdk::protocol::CommonRequest	aimdk::protocol::RcEmoticonListResp
`pb:/aimdk.protocol.RcEmoticonPlayerService/PlayerEmoticon`	Play emoticon	aimdk::protocol::RcPlayerEmoticonRequest	aimdk::protocol::RcPlayerEmoticonR

For corresponding preview images, please refer to the AimMaster software.

emoticon_id	emoticon_name	Description
1	emoticon_prompt_words	Wake-up word prompt
2	emoticon_say_hi	Typically used when greeting the user.
3	emoticon_dance	Typically used when the robot is dancing.
4	emoticon_red_heart	Expressing a heart gesture or gratitude to the user.
5	emoticon_bye	Typically used when saying goodbye to the user.
6	emoticon_sending_a_heart	Typically used when showing care or sending love to the user.
7	emoticon_happy	Happy, joyful

emoticon_id	emoticon_name	Description
8	emoticon_love_u	Typically used when boldly expressing love.
9	emoticon_yeah	Typically used to express good luck or convey positivity.
10	emoticon_ok	Typically used to indicate “no problem” or “completed”.
11	emoticon_normal	General-purpose expression: used during exercise or task execution.
12	emoticon_sad	Feeling down or pretending to be in a bad mood.
13	emoticon_welcome	Typically used to warmly welcome someone.
14	emoticon_like	Typically used to express approval or agreement.
15	emoticon_disable_voice	Voice input disabled
16	emoticon_language_change	Expression shown when switching between Chinese and English.
17	emoticon_working_mode	Expression used during task execution mode.
18	emoticon_continuous_shot_1	First expression in a three-shot sequence
19	emoticon_continuous_shot_2	Second expression in a three-shot sequence
20	emoticon_continuous_shot_3	Third expression in a three-shot sequence
21	emoticon_thinking	General-purpose expression: used during thinking states (e.g., reasoning, multimodal tasks requiring extended thought)
22	emoticon_voice_rejection	User’s spoken input was rejected; voice-specific usage
23	emoticon_human_voice_input	Currently listening to the user’s speech; voice-specific usage
24	emoticon_countdown	Tic-tac-toe game expression: game entering countdown state
25	emoticon_draw	Tic-tac-toe game expression: indicates a draw
26	emoticon_board	Tic-tac-toe game expression: indicates currently playing tic-tac-toe
27	emoticon_lost	Tic-tac-toe game expression: indicates A2 lost this round
28	emoticon_won	Tic-tac-toe game expression: indicates A2 won this round
29	emoticon_identification_error	Tic-tac-toe game expression: indicates an error occurred while recognizing the current game state
30	emoticon_guiding	User has initiated the guiding process

Protobuf Message Types

`aimdk::protocol::RcMotionListResponse`

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Response header

Field	Type	Description
motion_list	aimdk::protocol::RcMotionInfo []	Motion list

Field	Type	Description
motion_id	uint32	Motion ID
motion_name	string	Motion name
time_stamp	uint64	Timestamp
is_change	bool	Whether modification is supported; wheeled motions cannot be modified, while legged motions can
description	string	Description
display_name_zh	string	Display name in Chinese
display_name_en	string	Display name in English
url	string	Motion download URL
thumbnail_url	string	Thumbnail download URL
md5	string	File MD5 checksum
duration	uint32	Motion duration
motion_play_path	string	Motion playback file path
type	aimdk::protocol::MotionType	Motion type

`aimdk::protocol::MotionType`

Motion type

Name	Number	Description
Untyped	0	Uncategorized
Speech	1	Timed speech
RandomInBroadcast	2	Random during broadcast
SkillMotion	3	Skill motion
Show	4	Performance

`aimdk::protocol::RcPlayerMotionRequest`

Field	Type	Description
header	aimdk::protocol::RequestHeader	Request header
motion_id	uint32	Motion ID

`aimdk::protocol::RcEmoticonListResponse`

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Response header
emoticon_list	aimdk::protocol::RcEmoticonInfo []	Emoticon list

Emoticon-related information

Field	Type	Description
emoticon_id	uint32	Emoticon ID
emoticon_name	string	Emoticon name
expand_name	string	Extension name, e.g., “svg” (without the leading dot; the dot will be added automatically during resource composition)
time_stamp	uint64	Timestamp
is_change	bool	Indicates whether modification is supported. For example, wheeled robots cannot modify emoticons, while legged robots can.
display_name_zh	string	Display name in Chinese
display_name_en	string	Display name in English
description	string	Description
bqb_url	string	Download URL for the emoticon package
emoticon_url	string	Download URL for the emoticon file
thumbnail_url	string	Download URL for the preview file
cover_url	string	Download URL for the cover file
bqb_local_path	string	Local path of the emoticon package
emoticon_local_path	string	Local path of the emoticon file
thumbnail_local_path	string	Local path of the preview file
cover_local_path	string	Local path of the cover file
duration	uint32	Duration of the emoticon

`aimdk::protocol::RcPlayerEmoticonRequest`

Field	Type	Description
header	aimdk::protocol::RequestHeader	Request header
emoticon_id	uint32	Emoticon ID
is_need_data	bool	Indicates whether emoticon content is needed. If true, the `emticon_data` field in the response will contain data.

`aimdk::protocol::RcPlayerEmoticonResponse`

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Response header
emoticon_id	uint32	Emoticon ID
emoticon_name	string	Emoticon name
expand_name	string	Extension name
emoticon_path	string	Emoticon file path
emticon_data	bytes	Emoticon data

Usage Example

Action Playback Related

```
#!/bin/bash

# 播放某个动作
# ./a2_PlayerMotion.sh motion_id

if [ $# -eq 0 ]; then
    echo "arg error, need motion_id, ./a2_PlayerMotion.sh motion_id, example: "
    echo ./a2_PlayerMotion.sh 1
    exit 0
fi

curl -i \
    -H 'content-type:application/json' \
    -H 'timeout: 1000' \
    -X POST http://192.168.100.100:59001/rpc/aimdk.protocol.RcMotionPlayerService/PlayerMotion \
    -d '{"motion_id":"'$1'"}'
```

```
#!/bin/bash

# Get the list of supported actions for playback
# ./a2_GetMotionList.sh

curl -i \
    -H 'content-type:application/json' \
    -H 'timeout: 1000' \
    -X POST http://192.168.100.100:59001/rpc/aimdk.protocol.RcMotionPlayerService/GetMotionList \
    -d '{}'
```

表情播放相关

```
#!/bin/bash

# 播放某个表情
# ./a2_PlayerEmoticon.sh emoticon_id

if [ $# -eq 0 ]; then
    echo "arg error, need emoticon_id, ./a2_PlayerEmoticon.sh emoticon_id, example: "
    echo ./a2_PlayerEmoticon.sh 1
    exit 0
fi

curl -i \
    -H 'content-type:application/json' \
    -H 'timeout: 1000' \
```

```
    -X POST http://192.168.100.100:59001/rpc/aimdk.protocol.RcEmoticonPlayerService/PlayerEmoticon \
    -d '{"emoticon_id":"'$1'","is_need_data":false}'
```

```
#!/bin/bash

# 获取表情列表
# ./a2_GetEmoticonList.sh

curl -i \
    -H 'content-type:application/json' \
    -H 'timeout: 1000' \
    -X POST http://192.168.100.100:59001/rpc/aimdk.protocol.RcEmoticonPlayerService/GetEmoticonList \
    -d '{}'
```

LED Strip Control Module

This module resides on the ORIN development board, with the module name `skill pilot` and HTTP port number 52893.

RPC Interfaces

Interface Name	Description	Request Message Type	Response Message Type	Notes
`pb:/aimdk.protocol.HalRgbLightService/SetRgbLightCommand`	Set LED Strip	aimdk::protocol::RgbLightCommandReq	aimdk::protocol::CommonResponse	Note: 192.168.1 port is 5
`pb:/aimdk.protocol.HalRgbLightService/GetRgbLightState`	Get LED Strip State	aimdk::protocol::CommonRequest	aimdk::protocol::RgbLightStateRsp	Note: 192.168.1 port is 5

LED control logic is as follows:

1. All LED beads on the strip are controlled uniformly. You can set preset effects such as solid color, flowing light, breathing light, etc. Individual LEDs or strips cannot be controlled separately.
2. The robot's default lighting is a blue breathing effect, which can be overridden by user-defined commands.
3. Red flashing alerts for low battery, joint overheating, etc., have higher priority than all user-defined commands and cannot be overridden.

Protobuf Protocol

`aimdk::protocol::RgbLightCommandReq`
RGB LED control request

Field	Type	Description
header	aimdk::protocol::RequestHeader	Message header
cmd	aimdk::protocol::RgbLightCommand	RGB LED command

`aimdk::protocol::RgbLightStateRsp`
Response for getting RGB LED state

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Message header
data	aimdk::protocol::RgbLightState	RGB LED state data

`aimdk::protocol::RgbLightCommand`

RGB LED control command

Field	Type	Description
red	uint32	Range: 0~255
green	uint32	Range: 0~255
blue	uint32	Range: 0~255
effect	aimdk::protocol::RgbEffectStatus	RGB lighting effect
control	uint32	Lighting control authority: 0 for embedded takeover, 1 for client-side takeover

`aimdk::protocol::RgbEffectStatus`

Name	Number	Description
RgbEffectStatus_IDLE	0	Off
RgbEffectStatus_MONOCHROME	1	Solid color
RgbEffectStatus_BREATHING	2	Breathing light
RgbEffectStatus_FLASHING	3	Flashing light
RgbEffectStatus_FLOWING	4	Flowing light

`aimdk::protocol::RgbLightState`

RGB LED state

Field	Type	Description
red	uint32	Range: 0~255
green	uint32	Range: 0~255
blue	uint32	Range: 0~255
effect	aimdk::protocol::RgbEffectStatus	RGB lighting effect
control	uint32	Lighting control authority: 0 for embedded takeover, 1 for client-side takeover

Usage Example

```
# 设置灯带http调用
curl -i -H "content-type:application/json" \
        -H "timeout: 60000" \
```

```
        -X POST 'http://192.168.100.110:52893/rpc/aimdk.protocol.HalRgbLightService/SetRgbLightCommand'
\
        -d '{"cmd":{"red":255, "green":0, "blue":0, "effect":2, "control":1}}'

# 获取灯带http调用
curl -i -H "content-type:application/json" \
        -H "timeout: 60000" \
        -X POST 'http://192.168.100.100:56421/rpc/aimdk.protocol.HalRgbLightService/GetRgbLightState' \
        -d '{}'
```

Planning and Control Module

【This module is exclusively for Flagship model users. Base model users do not need to refer to this section.】

This page contains the documentation and protocols for topics and services included in the Planning and Control (PNC) module, supporting the system's planning and control functionalities.

This module resides on the Orin development board, with the HTTP backend listening on port 53176.

Channel Interfaces

Topic Name	Description	Subscribe or Publish	Message Type	Notes	Communication Backend
`/path`	Robot trajectory	Publish	nav_msgs::msg::Path	For visualization and debugging purposes only; for reference only	ros2
`/pnc/estimate_odom`	PNC-estimated velocity	Publish	nav_msgs::msg::Odometry	Estimated based on real-time localization results; published only during navigation tasks. The velocity data is for reference only and may not be accurate. For continuous and reliable robot localization information, please obtain it from the `/tf` transform (from `map` to `base_link`).	ros2

General instructions for task dispatching interfaces:
1. Before executing a task, the robot must successfully complete relocalization, and the map ID specified in the dispatched task must match the map used for relocalization.
2. Before executing a task, the MC state must be switched to RL_LOCOMOTION_DEFAULT (reinforced straight-leg) mode; otherwise, the robot may not properly respond to PNC velocity commands.
3. When dispatching a task, a `task_id` must be set. If not set (default is 0), PNC will automatically generate a `task_id` and return it in the response. The client must store this `task_id` for subsequent control commands or task status queries.

Interface Name	Description	Request Message Type	Response Message Type
`pb:/aimdk.protocol.PncService/PlanningNaviToGoal`	Dispatch a planning-based navigation task to a specified goal point ID	aimdk::protocol::PncPlanningNaviToGoalRequest	aimdk::protocol::CommonTaskRespons
`pb:/aimdk.protocol.PncService/PlanningNaviToPose2D`	Dispatch a planning-based navigation	aimdk::protocol::PncPlanningNaviToPose2DRequest	aimdk::protocol::CommonTaskRespons

Interface Name	Description	Request Message Type	Response Message Type
	task to a specified 2D pose		
`pb:/aimdk.protocol.PncService/LinearNaviToRelative`	Dispatch a straight-line navigation task to a specified relative 2D pose	aimdk::protocol::PncLinearNaviToRelativeRequest	aimdk::protocol::CommonTaskRespons
`pb:/aimdk.protocol.PncService/LinearNaviToGoal`	Dispatch a straight-line navigation task to a specified goal point ID	aimdk::protocol::PncLinearNaviToGoalRequest	aimdk::protocol::CommonTaskRespons
`pb:/aimdk.protocol.PncService/LinearNaviToPose2D`	Dispatch a straight-line navigation task to a specified 2D pose	aimdk::protocol::PncLinearNaviToPose2DRequest	aimdk::protocol::CommonTaskRespons
`pb:/aimdk.protocol.PncService/DirectNaviToRelativeGoal`	Dispatch a direct movement task to a specified goal point ID	aimdk::protocol::PncDirectNaviToRelativeGoalRequest	aimdk::protocol::CommonTaskRespons
`pb:/aimdk.protocol.PncService/DirectNaviToRelative`	Dispatch a direct movement task to a specified relative 2D pose	aimdk::protocol::PncDirectNaviToRelativeRequest	aimdk::protocol::CommonTaskRespons
`pb:/aimdk.protocol.PncService/SpinTurnAndMoveForward`	Dispatch an in-place rotation followed by linear forward movement task	aimdk::protocol::PncSpinTurnAndMoveForwardRequest	aimdk::protocol::CommonTaskRespons

Interface Name	Description	Request Message Type	Response Message Type
`pb:/aimdk.protocol.PncService/SpinTurn`	Dispatch an in-place rotation task	aimdk::protocol::PncSpinTurnRequest	aimdk::protocol::CommonTaskRespons
`pb:/aimdk.protocol.PncService/MoveForward`	Dispatch a linear translation task	aimdk::protocol::PncMoveForwardRequest	aimdk::protocol::CommonTaskRespons
`pb:/aimdk.protocol.PncService/PreciseNaviToGoal`	Dispatch a high-precision navigation task to a specified goal point ID	aimdk::protocol::PncPreciseNaviToGoalRequest	aimdk::protocol::CommonTaskRespons
`pb:/aimdk.protocol.PncService/ActionCancel`	Dispatch a task cancellation command	aimdk::protocol::CommonTaskRequest	aimdk::protocol::CommonTaskRespons
`pb:/aimdk.protocol.PncService/ActionResume`	Send a command to resume a paused task	aimdk::protocol::CommonTaskRequest	aimdk::protocol::CommonTaskRespons
`pb:/aimdk.protocol.PncService/ActionGetState`	Get the state of a navigation task	aimdk::protocol::CommonTaskRequest	aimdk::protocol::PncServiceStateRespo

`aimdk::protocol::PncPathChannel`

Field	Type	Description
header	aimdk::protocol::Header	Custom message header specifying the coordinate frame of the path
pose	aimdk::protocol::SE3Pose []	Array of poses along the path

`aimdk::protocol::PncPlanningNaviToGoalRequest`

Field	Type	Description
header	aimdk::protocol::RequestHeader	Header for the RPC request
task_id	uint64	Task ID
map_id	uint64	Map ID
target_id	int64	Target point ID
guide_line_id	int64	Guide path ID (reserved field, unused)
ackerman_mode	bool	Ackermann mode (reserved field, unused)

`aimdk::protocol::PncPlanningNaviToPose2DRequest`

Field	Type	Description
header	aimdk::protocol::RequestHeader	Header for the RPC request
task_id	uint64	Task ID
map_id	uint64	Map ID
pose	aimdk::protocol::SE2Pose []	Target pose(s)
ackerman_mode	bool	Ackermann mode (reserved field, unused)

`aimdk::protocol::PncLinearNaviToRelativeRequest`

Field	Type	Description
header	aimdk::protocol::RequestHeader	Header for the RPC request
task_id	uint64	Task ID
map_id	uint64	Map ID
pose	aimdk::protocol::SE2Pose []	Target pose(s) relative to the robot's body frame

`aimdk::protocol::PncLinearNaviToGoalRequest`

Field	Type	Description
header	aimdk::protocol::RequestHeader	Header for the RPC request
task_id	uint64	Task ID
map_id	uint64	Map ID
target_id	int64	Target point ID

`aimdk::protocol::PncLinearNaviToPose2DRequest`

Field	Type	Description
header	aimdk::protocol::RequestHeader	Header for the RPC request
task_id	uint64	Task ID
map_id	uint64	Map ID
pose	aimdk::protocol::SE2Pose []	Target pose(s)

`aimdk::protocol::PncDirectNaviToRelativeGoalRequest`

Field	Type	Description
header	aimdk::protocol::RequestHeader	Header for the RPC request
task_id	uint64	Task ID
map_id	uint64	Map ID
target_id	int64	Target point ID

`aimdk::protocol::PncDirectNaviToRelativeRequest`

Field	Type	Description
header	aimdk::protocol::RequestHeader	Header for the RPC request
task_id	uint64	Task ID
map_id	uint64	Map ID
pose	aimdk::protocol::SE2Pose []	Target pose(s) relative to the robot's body frame

`aimdk::protocol::PncSpinTurnAndMoveForwardRequest`

Field	Type	Description
header	aimdk::protocol::RequestHeader	Header for the RPC request
task_id	uint64	Task ID
map_id	uint64	Map ID
angle	double	Rotation angle relative to the robot's current heading (unit: radians)
distance	double	Forward distance relative to the robot's current position (unit: meters)

`aimdk::protocol::PncSpinTurnRequest`

Field	Type	Description
header	aimdk::protocol::RequestHeader	Header for the RPC request
task_id	uint64	Task ID

Field	Type	Description
map_id	uint64	Map ID
angle	double	Rotation angle relative to the robot's current heading (unit: radians)

`aimdk::protocol::PncMoveForwardRequest`

Field	Type	Description
header	aimdk::protocol::RequestHeader	Header for the RPC request
task_id	uint64	Task ID
map_id	uint64	Map ID
angle	double	Translation direction relative to the robot's current heading (unit: radians)
distance	double	Translation distance relative to the robot's current position (unit: meters)

Field	Type	Description
header	aimdk::protocol::RequestHeader	Header of the RPC request
task_id	uint64	Task ID
map_id	uint64	Map ID
target_id	int64	Target point ID
pose_offset	aimdk::protocol::SE2Pose []	Relative pose offset from the target point

`aimdk::protocol::PncServiceStateResponse`

Field	Type	Description
header	aimdk::protocol::ResponseHeader	Header of the RPC response
task_id	uint64	Current task ID
state	aimdk::protocol::PncServiceState []	PNC execution states

`aimdk::protocol::PncServiceState`

Name	Number	Description
PncServiceState_UNDEFINED	0	Unknown state
PncServiceState_IDLE	1	Task is idle
PncServiceState_RUNNING	2	Task is running
PncServiceState_PAUSED	3	Task is paused
PncServiceState_SUCCESS	4	Task succeeded

Name	Number	Description
PncServiceState_FAILED	5	Task failed

Usage Examples

1. Issue a PlanningNaviToGoal task

```
#!/bin/bash
#curl使用示例(以 ORIN ip: 192.168.100.110 为例):
curl -i -H 'content-type:application/json' \
        -H 'timeout: 60000' \
        -X POST 'http://192.168.100.110:53176/rpc/aimdk.protocol.PncService/PlanningNaviToGoal' \
        -d '{
              "header": {
                "timestamp": {
                  "seconds": 0,
                  "nanos": 0,
                  "ms_since_epoch": 0
                },
                "control_source": 0
              },
              "task_id": 0,
              "map_id": 1,
              "target_id": 0,
              "guide_line_id": 0,
              "ackerman_mode": false
            }'
```

2. Issue a PlanningNaviToPose2D task

```
#!/bin/bash
#curl使用示例(以 ORIN ip: 192.168.100.110 为例):
curl -i -H 'content-type:application/json' \
        -H 'timeout: 60000' \
        -X POST 'http://192.168.100.110:53176/rpc/aimdk.protocol.PncService/PlanningNaviToPose2D' \
        -d '{
              "header": {
                "timestamp": {
                  "seconds": 0,
                  "nanos": 0,
                  "ms_since_epoch": 0
                },
                "control_source": 0
              },
              "task_id": 0,
              "map_id": 1,
              "pose": {
                "position": {
                  "x": 10.0,
                  "y": 5.0
                },
                "angle": 3.14159
              },
              "ackerman_mode": false
            }'
```

3. Issue LinearNaviToRelative Task

```
#!/bin/bash
#curl使用示例(以 ORIN ip: 192.168.100.110 为例):
curl -i -H 'content-type:application/json' \
        -H 'timeout: 60000' \
        -X POST 'http://192.168.100.110:53176/rpc/aimdk.protocol.PncService/LinearNaviToRelative' \
        -d '{
```

```
              "header": {
                "timestamp": {
                  "seconds": 0,
                  "nanos": 0,
                  "ms_since_epoch": 0
                },
                "control_source": 0
              },
              "task_id": 0,
              "map_id": 1,
              "pose": {
                "position": {
                  "x": 1.0,
                  "y": 1.0
                },
                "angle": 1.57
              }
            }'
```

4. Issue LinearNaviToGoal Task

```
#!/bin/bash
#curl使用示例(以 ORIN ip: 192.168.100.110 为例):
curl -i -H 'content-type:application/json' \
        -H 'timeout: 60000' \
        -X POST 'http://192.168.100.110:53176/rpc/aimdk.protocol.PncService/LinearNaviToGoal' \
        -d '{
              "header": {
                "timestamp": {
                  "seconds": 0,
                  "nanos": 0,
                  "ms_since_epoch": 0
                },
                "control_source": 0
              },
              "task_id": 0,
              "map_id": 1,
              "target_id": 0
            }'
```

5. Issue LinearNaviToPose2D Task

```
#!/bin/bash
#curl使用示例(以 ORIN ip: 192.168.100.110 为例):
curl -i -H 'content-type:application/json' \
        -H 'timeout: 60000' \
        -X POST 'http://192.168.100.110:53176/rpc/aimdk.protocol.PncService/LinearNaviToPose2D' \
        -d '{
              "header": {
                "timestamp": {
                  "seconds": 0,
                  "nanos": 0,
                  "ms_since_epoch": 0
                },
                "control_source": 0
              },
              "task_id": 0,
              "map_id": 1,
              "pose": {
                "position": {
                  "x": 1.0,
                  "y": 1.0
                },
                "angle": 1.57
              }
            }'
```

6. Issue DirectNaviToRelativeGoal Task

```
#!/bin/bash
#curl使用示例(以 ORIN ip: 192.168.100.110 为例):
curl -i -H 'content-type:application/json' \
        -H 'timeout: 60000' \
        -X POST 'http://192.168.100.110:53176/rpc/aimdk.protocol.PncService/DirectNaviToRelativeGoal' \
        -d '{
              "header": {
                "timestamp": {
                  "seconds": 0,
                  "nanos": 0,
                  "ms_since_epoch": 0
                },
                "control_source": 0
              },
              "task_id": 0,
              "map_id": 1,
              "target_id": 0
            }'
```

7. Issue DirectNaviToRelative Task

```
#!/bin/bash
#curl使用示例(以 ORIN ip: 192.168.100.110 为例):
curl -i -H 'content-type:application/json' \
        -H 'timeout: 60000' \
        -X POST 'http://192.168.100.110:53176/rpc/aimdk.protocol.PncService/DirectNaviToRelative' \
        -d '{
              "header": {
                "timestamp": {
                  "seconds": 0,
                  "nanos": 0,
                  "ms_since_epoch": 0
                },
                "control_source": 0
              },
              "task_id": 0,
              "map_id": 1,
              "pose": {
                "position": {
                  "x": 1.0,
                  "y": 1.0
                },
                "angle": 0.0
              }
            }'
```

8. Issue SpinTurnAndMoveForward Task

```
#!/bin/bash

# curl usage example (assuming ORIN IP: 192.168.100.110):
curl -i -H 'content-type:application/json' \
        -H 'timeout: 60000' \
        -X POST 'http://192.168.100.110:53176/rpc/aimdk.protocol.PncService/SpinTurnAndMoveForward' \
        -d '{
              "header": {
                "timestamp": {
                  "seconds": 0,
                  "nanos": 0,
                  "ms_since_epoch": 0
                },
                   "control_source": 0
              },
              "task_id": 0,
              "map_id": 1,
```

```
              "angle": 3.14159,
              "distance": 1.0
            }'
```

9. 下发SpinTurn任务

```
#!/bin/bash
#curl使用示例(以 ORIN ip: 192.168.100.110 为例):
curl -i -H 'content-type:application/json' \
        -H 'timeout: 60000' \
        -X POST 'http://192.168.100.110:53176/rpc/aimdk.protocol.PncService/SpinTurn' \
        -d '{
              "header": {
                "timestamp": {
                  "seconds": 0,
                  "nanos": 0,
                  "ms_since_epoch": 0
                },
                "control_source": 0
              },
              "task_id": 0,
              "map_id": 1,
              "angle": 3.14159
            }'
```

10. Issue MoveForward Task

```
#!/bin/bash
#curl使用示例(以 ORIN ip: 192.168.100.110 为例):
curl -i -H 'content-type:application/json' \
        -H 'timeout: 60000' \
        -X POST 'http://192.168.100.110:53176/rpc/aimdk.protocol.PncService/MoveForward' \
        -d '{
              "header": {
                "timestamp": {
                  "seconds": 0,
                  "nanos": 0,
                  "ms_since_epoch": 0
                },
                "control_source": 0
              },
              "task_id": 0,
              "map_id": 1,
              "angle": 0.0,
              "distance": 1.0
            }'
```

11. Issue PreciseNaviToGoal Task

```
#!/bin/bash
#curl使用示例(以 ORIN ip: 192.168.100.110 为例):
curl -i -H 'content-type:application/json' \
        -H 'timeout: 60000' \
        -X POST 'http://192.168.100.110:53176/rpc/aimdk.protocol.PncService/PreciseNaviToGoal' \
        -d '{
              "header": {
                "timestamp": {
                  "seconds": 0,
                  "nanos": 0,
                  "ms_since_epoch": 0
                },
                "control_source": 0
              },
              "task_id": 0,
              "map_id": 1,
```

```
              "target_id": 0,
              "pose_offset": {
                "position": {
                  "x": 1.0,
                  "y": 1.0
                },
                "angle": 0.0
              }
            }'
```

12. Cancel Task

```
#!/bin/bash
#curl使用示例(以 ORIN ip: 192.168.100.110 为例):
curl -i -H 'content-type:application/json' \
        -H 'timeout: 60000' \
        -X POST 'http://192.168.100.110:53176/rpc/aimdk.protocol.PncService/ActionCancel' \
        -d '{
              "header": {
                "timestamp": {
                  "seconds": 0,
                  "nanos": 0,
                  "ms_since_epoch": 0
                },
                "control_source": 0
              },
              "task_id": 6874745825616545102
            }'
```

13. Query Task Status

```
#!/bin/bash
#curl使用示例(以 ORIN ip: 192.168.100.110 为例):
curl -i -H 'content-type:application/json' \
        -H 'timeout: 60000' \
        -X POST 'http://192.168.100.110:53176/rpc/aimdk.protocol.PncService/ActionGetState' \
        -d '{
              "header": {
                "timestamp": {
                  "seconds": 0,
                  "nanos": 0,
                  "ms_since_epoch": 0
                },
                "control_source": 0
              },
              "task_id": 6874745825616545102
            }'
```

Task Execution Engine Module

【This module is exclusively for Flagship model users; Base model users do not need to refer to this documentation.】

This page contains the documentation and protocol specifications for topics and services included in the Task Execution Engine module (TaskEngine), supporting users in launching and controlling tasks.

Note that this module only handles task launching and control, and does **not** include task management operations such as creation, modification, or deletion. For management operations, please refer to AimMaster.
The primary purpose of this module's interfaces is to provide a programmatic interface for lightweight secondary development, enabling task start/stop operations originally available on AimMaster.

Additionally, please avoid mixing controls from AimMaster and the TaskEngine API. Concurrent usage of both control methods may lead to undefined behavior due to potential interface conflicts.

On the Flagship model, this module resides on the Orin development board, with its HTTP backend listening on port **57881** .

Note : Before invoking the execution engine, ensure the robot is in **Auto Mode** , which can be configured via AimMaster.

RPC Interfaces

Interface Name	Description	Request Message Type	Response Message Type	No
`pb:/aimdk.protocol.TaskEngineService/GetTask`	Retrieve execution information for a single task	aimdk::protocol::GetTaskRequest	aimdk::protocol::GetTaskResponse	
`pb:/aimdk.protocol.TaskEngineService/GetAllTasks`	Retrieve execution information for all tasks	aimdk::protocol::GetAllTasksRequest	aimdk::protocol::GetAllTasksResponse	
`pb:/aimdk.protocol.TaskEngineService/SetCurrentTask`	Set the current task by providing a task ID	aimdk::protocol::SetCurrentTaskRequest	aimdk::protocol::SetCurrentTaskResponse	Ca wh tas
`pb:/aimdk.protocol.TaskEngineService/LaunchTask`	Start executing the current task	aimdk::protocol::LaunchTaskRequest	aimdk::protocol::LaunchTaskResponse	1. `Se` be lau 2. sa pa dif `ta` in be
`pb:/aimdk.protocol.TaskEngineService/CtrlTaskState`	Control task state (pause, resume, terminate, etc.)	aimdk::protocol::CtrlTaskStateRequest	aimdk::protocol::CtrlTaskStateResponse	

Protobuf Message Types

`aimdk::protocol::Task`

Task metadata

Field	Type	Description
task_id	int64	Task ID, globally unique identifier for a task. When creating/modifying tasks via management interfaces: `task_id > 0` indicates task modification; `task_id < 0` indicates new task creation. Clients can obtain all valid `task_id` s by calling `GetAllTasks` .
name	string	Task name
fsm_name	string	Task FSM (Finite State Machine) name

Field	Type	Description
is_valid	bool	Task validity status
state	aimdk::protocol::Task::StateType	Current task state
duration	int64	Task execution duration (optional)
setting	string	Original YAML configuration string passed from the system task settings to the task engine (optional)
task_data	bytes	Internal task-associated data (opaque binary payload)

State Type

Name	Number	Description
StateType_UNDEFINED	0	
StateType_IDLE	1	Idle state
StateType_RUNNING	2	Running
StateType_PAUSED	3	Paused
StateType_STOPPED	4	Stopped
StateType_PAUSING	5	Pausing
StateType_STOPPING	6	Stopping
StateType_RESUMING	7	Resuming
StateType_FAILPAUSE	8	Failed to pause
StateType_FAILSTOP	9	Failed to stop
StateType_FAILRESUME	10	Failed to resume
StateType_EXCEPTION	11	Runtime exception
StateType_LAZY_PAUSING	12	Pausing until beat ends

`aimdk::protocol::TaskEngineHeartBeat`

Task engine status heartbeat message

Field	Type	Description
header	aimdk::protocol::RequestHeader	
data	aimdk::protocol::Task	Task information

`aimdk::protocol::GetTaskRequest`

Task information request

Field	Type	Description
header	aimdk::protocol::RequestHeader	
task_id	int64	Task ID

`aimdk::protocol::GetTaskResponse`
Task information response

Field	Type	Description
header	aimdk::protocol::ResponseHeader	
data	aimdk::protocol::Task	Task information

`aimdk::protocol::GetAllTasksRequest`
Request for all task information

Field	Type	Description
header	aimdk::protocol::RequestHeader	

`aimdk::protocol::GetAllTasksResponse`
Response with all task information

Field	Type	Description
header	aimdk::protocol::ResponseHeader	
data	aimdk::protocol::Task []	List of task information

`aimdk::protocol::GetAllFSMsRequest`
Request for all task FSM information

Field	Type	Description
header	aimdk::protocol::RequestHeader	

`aimdk::protocol::GetAllFSMsResponse`
Response with all task FSM information

Field	Type	Description
header	aimdk::protocol::ResponseHeader	
data	string	List of task FSM information

`aimdk::protocol::SetTaskRequest`
Task configuration setting request

Field	Type	Description
header	aimdk::protocol::RequestHeader	
data	string	Raw YAML configuration generated by the client

If creating a new task, `task_id` in the `data` field should be `-1` ; if modifying an existing task, `task_id` should be the ID of the existing task.

`aimdk::protocol::SetTaskResponse`
Task configuration setting response

Field	Type	Description
header	aimdk::protocol::ResponseHeader	
task_id	int64	ID of the successfully created or modified task
is_success	bool	Whether the configuration was set successfully

Request to set the current task (not necessarily a running task; cannot be set while another task is running).

Field	Type	Description
header	aimdk::protocol::RequestHeader	
task_id	int64	Task ID to be set as the current task

`aimdk::protocol::SetCurrentTaskResponse`
Response to set the current task.

Field	Type	Description
header	aimdk::protocol::ResponseHeader	
task_id	int64	Task ID that was set
is_success	bool	Whether the setting succeeded

`aimdk::protocol::DeleteTaskRequest`
Task deletion request.

Field	Type	Description
header	aimdk::protocol::RequestHeader	
task_id	int64	Task ID

`aimdk::protocol::DeleteTaskResponse`
Response to the task deletion request.

Field	Type	Description

Field	Type	Description
header	aimdk::protocol::ResponseHeader	
task_id	int64	Task ID
is_success	bool	Whether deletion succeeded

`aimdk::protocol::DeleteTaskMapRequest`
Map deletion request.

Field	Type	Description
header	aimdk::protocol::RequestHeader	
map_id	string	Map ID

`aimdk::protocol::DeleteTaskMapResponse`
Response to the map deletion request.

Field	Type	Description
header	aimdk::protocol::ResponseHeader	
map_id	string	Map ID
is_success	bool	Whether deletion succeeded

`aimdk::protocol::LaunchTaskRequest`
Task launch request.

Field	Type	Description
header	aimdk::protocol::RequestHeader	
task_id	int64	Task ID

`aimdk::protocol::LaunchTaskResponse`
Result of the task launch request.

Field	Type	Description
header	aimdk::protocol::ResponseHeader	
task_id	int64	Task ID
res	aimdk::protocol::LaunchTaskResponse::ReturnType	Launch request result

`aimdk::protocol::LaunchTaskResponse::ReturnType`
Launch request result type.

Name	Number	Description
ReturnType_UNDEFINED	0	
ReturnType_SUCCEED	1	Request succeeded
ReturnType_FAIL_TASK_CONFLICT	2	Another task is running; task conflict
ReturnType_FAIL_TASK_VALIDATE	3	Task validation failed
ReturnType_FAIL_UNKNOWN	4	Unknown reason

Task state switching request

Field	Type	Description
header	aimdk::protocol::RequestHeader	
task_id	int64	Task ID. If not set, the current task will be paused.
type	aimdk::protocol::CtrlTaskStateRequest::Type	Control type

`aimdk::protocol::CtrlTaskStateRequest::Type`
Task state switching type

Name	Number	Description
Type_UNDEFINED	0	
Type_PAUSE	1	Pause
Type_RESUME	2	Resume
Type_STOP	3	Stop

`aimdk::protocol::CtrlTaskStateResponse`
Task state switching response

Field	Type	Description
header	aimdk::protocol::ResponseHeader	
task_id	int64	Task ID
is_success	bool	Control result

Usage Examples

HTTP Curl examples are provided below. If executing outside the robot's ORIN development board, replace `192.168.100.110` with the actual IP address of the robot's ORIN development board, and replace `task_id` with the desired task ID:

Get a single task

```
curl --location --request POST 'http://192.168.100.110:57881/rpc/aimdk.protocol.TaskEngineService/
GetTask' \
--header 'Content-Type: application/json' \
--data-raw '{
       "task_id": "1"
    }'
# GetTaskResponse
{
"data": [
{
    "task_id": "1",
    "name": "task1",
    "is_valid": true,
    "state": "StateType_IDLE",
    "duration": "0",
    "setting": "task_type: interaction_1 xxx YAML content"
}
]}
```

Get all tasks

```
curl --location --request POST 'http://192.168.100.110:57881/rpc/aimdk.protocol.TaskEngineService/
GetAllTasks' \
--header 'Content-Type: application/json' \
--data-raw '{}'
# GetAllTasksResponse
{
"data": [
{
    "task_id": "1",
    "name": "task1",
    "is_valid": true,
    "state": "StateType_IDLE",
    "duration": "0",
    "setting": "task_type: interaction_1 xxx YAML content"
},
{
    "task_id": "2",
    "name": "task2",
    ...
}
...
]}
```

Set task by task ID

```
curl --location --request POST 'http://192.168.100.110:57881/rpc/aimdk.protocol.TaskEngineService/
SetCurrentTask' \
--header 'Content-Type: application/json' \
--data-raw '{
       "task_id": "15"
    }'

# SetCurrentTaskResponse
{
    "header":{
        "code":"0",
        "msg":"",
        "domain":""
    },
    "task_id":"15",
    "is_success":true
}
```

Start task

```
curl --location --request POST 'http://192.168.100.110:57881/rpc/aimdk.protocol.TaskEngineService/
LaunchTask' \
--header 'Content-Type: application/json' \
--data-raw '{
       "task_id": "15"
    }'

# LaunchTaskResponse
{
    "header":{
        "code":"0",
        "msg":"",
        "domain":""
    },
    "task_id":"14",
    "res":"ReturnType_SUCCEED"
}
```

Pause / Resume / Stop task

```
curl --location --request POST 'http://192.168.100.110:57881/rpc/aimdk.protocol.TaskEngineService/
CtrlTaskState' \
--header 'Content-Type: application/json' \
--data-raw '{
       "task_id": "15",
        "type" : "Type_PAUSE"
    }'

# CtrlTaskStateResponse
{
    "header":{
        "code":"0",
        "msg":"",
        "domain":""
    },
    "task_id":"14",
    "res":"ReturnType_SUCCEED"
}
```

Other APIs

To streamline documentation and reduce user comprehension overhead, certain less frequently used or highly independent APIs have been separated out. Basic invocation examples are provided here without listing them under specific modules.

The following API examples are intended to run by default on the Flagship ORIN development board. For the Base model, replace `192.168.100.110` with the x86 IP address (`192.168.100.100`).

Get Battery Status

```
curl -i     -H 'content-type:application/json' \
            -H 'timeout: 1000' \
            -X POST 'http://192.168.100.100:56421/rpc/aimdk.protocol.HalBmsService/GetBmsState' \
            -d '{}'
```

Get Emergency Stop Status

```
curl -i     -H 'content-type:application/json' \
            -H 'timeout: 1000' \
            -X POST 'http://192.168.100.100:56421/rpc/aimdk.protocol.HalEmergencyService/
```

```
GetEmergencyState' \
            -d '{}'
```

Get System Status

This refers to the software system status. By default, after powering on and setting the emergency stop remote control to ON, the system undergoes a series of state transitions, launching various software and algorithm modules. Once completed, it enters the READY state, indicating normal system operation.

Common system states:

- **Startup** : System is starting up
- **Ready** : Startup completed
- **Manual** : Manual operation mode
- **Auto** : Autonomous execution mode
- **MotionStream** : Teleoperation mode
- **OTA** : Over-the-air update mode

```
curl -i     -H 'content-type:application/json' \
            -H 'timeout: 1000' \
            -X POST 'http://192.168.100.110:51011/rpc/aimdk.protocol.SystemService/GetSystemState' \
            -d '{}'
```

Query Basic OTA Information of the Robot

```
curl -i     -H 'content-type:application/json' \
            -H 'timeout: 1000' \
            -X POST 'http://192.168.100.100:57900/rpc/aimdk.protocol.OTAService/GetRobotOtaInfo' \
            -d '{}'
```

Query Current OTA Update Progress of the Robot

```
curl -i     -H 'content-type:application/json' \
            -H 'timeout: 1000' \
            -X POST 'http://192.168.100.100:57900/rpc/aimdk.protocol.OTAService/GetCurrentOtaProgress' \
            -d '{}'
```

Frequently Asked Questions

【Important】Why are my custom programs on ORIN automatically deleted?

In versions V0.7 and above, the disk monitor module automatically cleans files in the user directory that have not been modified for over 24 hours. To prevent this, place your files under `/agibot/data/home/agi/Desktop` , which is a whitelist directory and will not be cleaned by the disk monitor module.

How can I confirm whether my robot is a T3 or P1 model?

T3 and P1 are hardware models with certain hardware differences, and their corresponding software also differs slightly.

You can run `cat /etc/bsp_version` on ORIN to check:

- If the file does **not exist** , your robot is a **T3** model.
- If the file **exists** , your robot is a **P1** model.

Can I modify the onboard software environment?

No. Modifying system dependencies may cause existing onboard software modules to fail, and no support or warranty will be provided for such modifications.

If system dependencies are altered, the original environment may become unrecoverable, requiring the robot to be returned to the factory for reflashing.

When deploying programs on ORIN that require additional dependencies, you have two options:

1. Use the existing ORIN environment and include all dependencies as source code dependencies in your program.
2. For Python programs, use virtual environments; for other languages, use containers to isolate your program from the host system environment and avoid modifying system dependencies.

Is it supported to develop and replace the onboard motion control program?

Developing your own motion control program requires significant experience in motion control development and is **not recommended** .

If you need to develop a custom motion control program for research purposes, please contact our business team for further discussion. Note that replacing the motion control program will disable all built-in motion-related functionalities of the robot. Please proceed with caution, as any resulting falls or damage will **not be covered under warranty** .

Can I directly control the robot using ROS2?

Yes. High-level control interfaces (RPC) still need to be invoked via HTTP protocol, but low-level motion control and sensor data acquisition can be developed using ROS2.

Why can't I receive ROS2 topics published by the backend on ORIN?

Before running ROS2 examples on ORIN, you need to set the following environment variables:

```
source /opt/ros/humble/setup.bash
export ROS_DOMAIN_ID=232
export ROS_LOCALHOST_ONLY=0
export FASTRTPS_DEFAULT_PROFILES_FILE=/agibot/software/v0/entry/bin/cfg/ros_dds_configuration.xml
```

Generally, **do not modify** the DDS configuration file above. Other configurations have not been tested and cannot guarantee normal communication between onboard modules.

The default QoS settings for ROS2 topics are:

```
history: keep_last
depth: 10
reliability: best_effort
```

Please ensure that the QoS settings are consistent between publishers and subscribers to avoid communication failures due to QoS incompatibility.

Why can't I receive the robot's ROS2 topics on my personal development PC?

To ensure ROS2 communication stability, ROS2 traffic is restricted by default to communication between the robot's ORIN and the x86 development board, and cannot be accessed externally.

If you absolutely need to receive ROS2 topics from your personal PC, follow these steps:

1. Configure your PC to the `192.168.2.0/24` subnet and connect via Ethernet cable to the left network port on the robot's rear panel (the ORIN network debugging port). After connecting, ensure you can ping `192.168.2.50` ; otherwise, communication will not work.

2. Install ROS2 Humble on your PC.

3. Back up the following files on ORIN:
   `/agibot/software/v0/entry/bin/cfg/ros_dds_configuration.xml`
   `/agibot/software/v0/entry/bin/cfg/privileged_ros_dds_configuration.xml`

   ```
   cp /agibot/software/v0/entry/bin/cfg/ros_dds_configuration.xml /agibot/software/v0/entry/bin/cfg/
   ros_dds_configuration.xml.backup
   cp /agibot/software/v0/entry/bin/cfg/privileged_ros_dds_configuration.xml /agibot/software/v0/entry/
   bin/cfg/privileged_ros_dds_configuration.xml.backup
   ```

4. On ORIN, modify both `ros_dds_configuration.xml` and `privileged_ros_dds_configuration.xml` by adding `192.168.2.50` to the `interfaceWhitelist` list in each file. After modification, the relevant code snippet should look like this:

   ```
   <interfaceWhitelist>
      <address>192.168.100.110</address>
      <address>192.168.2.50</address>
   </interfaceWhitelist>
   ```

5. Run the following command on your PC. You should now be able to receive ROS2 topics published by the robot's ORIN:

   ```
   source /opt/ros/humble/setup.bash
   export ROS_DOMAIN_ID=232
   export ROS_LOCALHOST_ONLY=0
   ros2 topic list
   ```

⚠️ This method has not been widely tested and is provided for development and debugging purposes only. The robot must be under protective measures when using this configuration, and it must NOT be used in production environments.

How can a container on ORIN communicate with the ORIN host via ROS2?

1. When creating the container, share IPC and network with the host by adding the following options:

   ```
   --ipc=host --net=host
   ```

2. Copy the file `/agibot/software/v0/entry/bin/cfg/ros_dds_configuration.xml` from ORIN into the container.

3. Before echoing or receiving data inside the container, run the following command (note: replace the DDS config file path with the actual path inside the container):

```
source /opt/ros/humble/setup.bash
export ROS_DOMAIN_ID=232
export ROS_LOCALHOST_ONLY=0
export FASTRTPS_DEFAULT_PROFILES_FILE=/path/to/ros_dds_configuration.xml
```

4. Run the following command inside the container. You should now be able to receive ROS2 topics published by the robot's ORIN:

```
ros2 topic list
```

What should I do if certain APIs are not working?

1. **HTTP API call failures** – troubleshoot based on return values and symptoms:
   - "Connection refused" may indicate incorrect IP, PORT, or URL.
   - If IP/PORT/URL are correct but connection is still refused, the corresponding module might not be running (possibly due to an onboard error or E-stop being OFF).
   - "req deserialization failed" suggests an issue with the request body format.
   - "URL not found" means the URL is likely incorrect.
   - All HTTP APIs have usage examples—please carefully compare your code with the provided examples.
2. **Motion control API issues** :
   - Verify that you are calling the correct action code under the appropriate Action state.
   - If upper-body motion APIs are unresponsive, ensure you are **not** in `JOINT_SERVO` mode (in this mode, you must call the `DisableMotionPlayer` API from the motion playback module to enable motion).
   - If dexterous hand APIs are unresponsive, the torque might be too low—try adjusting `position` and `effort` values and test multiple times.
3. **ROS2-related issues** :
   - Check if required environment variables are loaded (`ROS_DOMAIN_ID` , `ROS_LOCALHOST_ONLY` , `FASTRTPS_DEFAULT_PROFILES_FILE`).
   - Verify that QoS settings are correct.
   - Confirm that the topic name is spelled correctly.

If the issue persists after the above checks, please contact the relevant technical support team.

Does the system support user-defined motions?

Yes, but you need to purchase teleoperation equipment. Motions can be recorded directly in AimMaster.

Recorded motion files must be placed in the `/agibot/data/var/rc/custom` directory on the x86 board following the specified procedure.## What are the joint limits for the robot?

The joint limits for the arms, head, and dexterous hands are provided in the Motion Control Module Documentation . They are listed again here for reference:

Arm Joint Limits (Position control only. For velocity and acceleration control, please implement via timing between position commands. The following velocity and acceleration limits are provided as reference):

Joint Name	Min Angle (rad)	Max Angle (rad)	Max Speed (rad/s)	Max Acceleration (rad/s²)
Left Arm J1	-2.91	2.91	3.00	6.28
Left Arm J2	-0.46	1.60	3.00	6.28
Left Arm J3	-2.91	2.91	3.00	6.28
Left Arm J4	-2.00	-0.03	3.00	6.28
Left Arm J5	-2.94	2.94	3.00	6.28
Left Arm J6	-0.45	0.45	3.00	6.28
Left Arm J7	-0.35	0.35	3.00	6.28
Right Arm J1	-2.91	2.91	3.00	6.28
Right Arm J2	-1.60	0.46	3.00	6.28
Right Arm J3	-2.91	2.91	3.00	6.28
Right Arm J4	0.03	2.00	3.00	6.28
Right Arm J5	-2.94	2.94	3.00	6.28
Right Arm J6	-0.45	0.45	3.00	6.28
Right Arm J7	-0.35	0.35	3.00	6.28

Head Joint Limits (Position control only):

Joint Name	Min Angle (rad)	Max Angle (rad)
neck_shake	-0.785	0.785
neck_nod	-0.401	0.401

Dexterous Hand Joint Limits (Supports both position and torque control; limits are identical for left and right hands):

Joint Name	Min Position	Max Position	Min Torque (g)	Max Torque (g)
Thumb Joint 1	0	2000	0	5700
Thumb Joint 2	0	2000	0	5700
Index Finger	0	2000	0	5700
Middle Finger	0	2000	0	5700
Ring Finger	0	2000	0	5700
Little Finger	0	2000	0	5700

Where is it recommended to deploy secondary development programs?

1. **Deployment on the x86 development board is not allowed**
   - The motion control software runs on the x86 board. Deploying additional programs on this board may cause motion control anomalies due to OS scheduling, CPU load, memory usage, etc., potentially leading to robot instability or even falls.
2. **Simple secondary development programs**
   - Programs that call basic RPC interfaces for high-level control, such as those built on Agibot's software and algorithms for functions like exhibition hall navigation or voice interaction
   - Recommended deployment: on a third-party computer (e.g., a laptop) or development board, using HTTP protocol for calls
3. **Complex secondary development programs** (e.g., embodied manipulation, interactive intelligence)
   - Programs requiring Channel interfaces for continuous data acquisition and robot control, such as embodied AI algorithms
   - Recommended deployment: on the Orin development board or a third-party industrial PC
     - If deployed on Orin: the base Orin environment must not be modified. Python programs should use virtual environments for isolation; C++ and other languages should use Docker for environment isolation
   - RPC interfaces still use HTTP protocol, while Channel interfaces use ROS2

Why do motion commands to the head, arms, or dexterous hands have no effect?

When the motion control module is in `JOINT_SERVO` mode, control of the upper body is taken over by the motion playback module.

In software version V0.6 and above, the motion playback module only continuously sends commands to the robot's head, arms, and dexterous hands when the motion control module is in `JOINT_SERVO` mode. Therefore, in other modes, the motion playback module will not interfere with control of these components.

If you still wish to stop the motion playback module from sending commands to the head, arms, and dexterous hands, you can call the `DisableMotionPlayer` interface of the motion playback module.

```
curl -i \
    -H 'content-type:application/json' \
    -H 'timeout: 1000' \
    -X POST http://192.168.100.100:56444/rpc/aimdk.protocol.MotionCommandService/DisableMotionPlayer \
    -d '{}'
```

SE3 pose control for the arms can only be properly invoked when the motion control module is in `PLANNING_MOVE` mode. Calling this interface does not require disabling the motion playback module.## [Not Recommended] How to Modify the Frame Rate and Resolution of the Default Sensor Topics?

Modifying this configuration will disable all modules that depend on camera data, including perception, obstacle avoidance, and navigation. No related support or guarantees will be provided.

1. First, back up the relevant configuration files for recovery purposes.

```
cd /agibot/software/v0/config/hal_sensor
cp a2_t2_sensor.yaml a2_t2_sensor.yaml.backup
```

1. Modify the `width` , `height` , and `fps` parameters of the corresponding camera in `/agibot/software/v0/config/hal_sensor/a2_t2_senor.yaml` . Only these three parameters can be changed.

2. Restart the `hal_sensor` module.

```
aima em stop-app hal_sensor
aima em start-app hal_sensor
```

The camera `width` , `height` , and `fps` only support fixed parameter combinations. To obtain valid parameter combinations, follow the method below:

In Step 2 above, change the `height` of a specific camera to its original value +1, then restart `hal_sensor` .

```
cat /agibot/log/hal_sensor/hal_sensor.log
```

Scroll up in the log to find the corresponding triplet table as shown below:

Why Does the Motion Command Only Raise the Right Hand?

For the `pb:/aimdk.protocol.MotionCommandService/SendMotionCommand` interface, an incorrect `motion_id` parameter will cause this issue. A correct parameter should resemble the following:

```
/agibot/data/resources/default/motion/motion_player/default/演讲10s/演讲10s
```

Are the Timestamps of Various Data Sources Hardware Timestamps or ROS2 Timestamps?

1. The joint state timestamps published by the motion control module use the system timestamp when the embedded message was received.
2. The chin-mounted RealSense D415 uses the timestamp at the moment the frame was captured.
3. The hip-mounted Orbbec uses the system clock timestamp when the `hal_sensor` module publishes the data.
4. Fisheye cameras differ between T3 and P1 models:
    1. T3 models use Xinying cameras, which estimate the exposure center time as system time minus half the exposure duration.
    2. P1 models use Senyun cameras, which use the timestamp when the camera module receives the image from the camera SDK.

How to Obtain the Robot's Position on the Map?

Prerequisite knowledge: Understanding SLAM occupancy grid maps and their corresponding pixel coordinates.

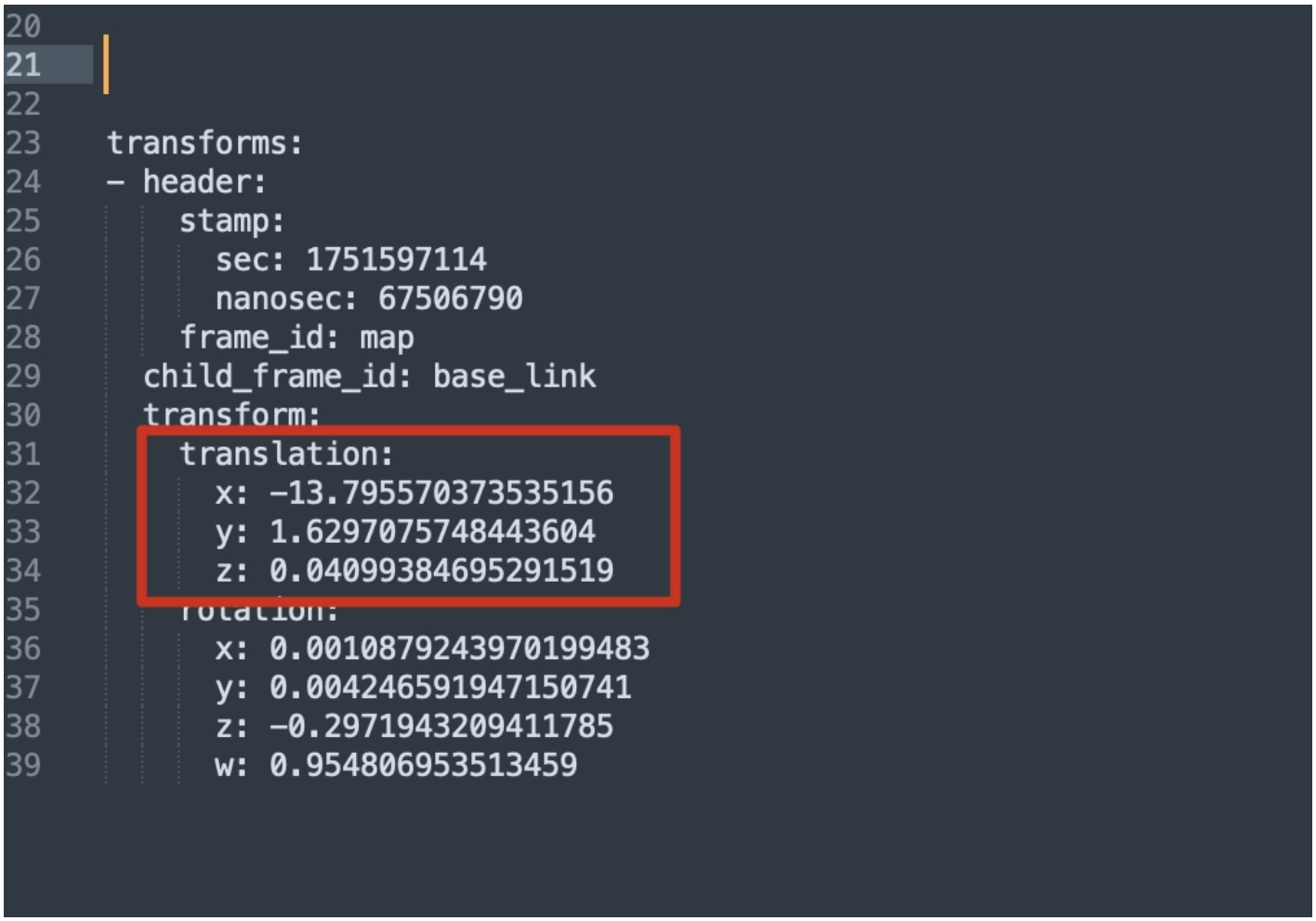
1. Obtain map information from the `pb:/aimdk.protocol.MappingService/Get2DWholeMap` interface, as shown below:

a2-map-data-demo.txt


```
{
        "header": {
                "code": "0",
                "msg": "",
                "timestamp": {
                        "seconds": "0",
                        "nanos": 0,
                        "ms_since_epoch": "1751016216225"
                }
        },
        "data": {
                "map_id": "1750750957172",
                "map_name": "3A展厅2",
                "width": 843,
                "height": 652,
                "resolution": 20,
                "origin_x": 478,
                "origin_y": 288,
                "map_data": "iVBORw0KGgoAAAANSUhEUgAAA0sAAAKMCAIAAAC1vLSZAAAgAElEQVR42u3dS7LVOBdo6/
wzskwDqNAJ6kSQraBKw6jSCoigTieo0AA6kLfgc3V0JNtLtmVbkr8RGRmb/VjLr2UNz6k59b///vvvLwAAAAzE3w4BAAAAw3sQv37/cRAAAEB3/
E+WFgAAYDDE8IoQzAMAAB0hhgcAADAaYngAAAAMDwAAAAwPAAAADK8nvnz9/
uXr9+kLRwMAALSASovKtvf500fHAQAA3IsYHgAAwGiI4QEAAIyGGB4AAADDAwAAAMMDAAAAwwMAAADDAwAAwP/PPw7B9YTeyJrnAQCAM9At5Ta9C/A8AABQF1nae/j3w/
vY+ax4BgAAKiKGdymTycV699dff3378XP64vOnj9Y9AwAADK8/vcsNL5a8oHoOFwAA2I0sbRP8+
+F9krd1TAAAwG7E8K5jNkWbE8fzBPMAAADD617veB4AADiOLG2jKLYFAAC7EcO7gq0BvJikCENIDwAAvEQM7yK9201ShAEAAMDwWuGgpcWeF5K2UrcAAGAWWdpzWemBtw9JWwAAwPCaMLw
z0qyKbQEAAMO7Te9OMjwrYQAAgCXMw7uCPLVaBSthAACAWcTwziL41udPH3/9/nOlRArmAQDA8BjeiXo3Rdpi/
Tq19QnPAwAAsrRnMdlVcKxgdWd3tpttqgIAAJ6GGF59Eq8KMbwrGxcrwgAA4MmI4Y2JIgwAABgeztKs8wppt3qepC0AAM9BlrYySQ+8oHcV17TY91KKMAAAeA5ieD0x6d23Hz93xAWTeJ6
DCQDAwIjh1ee8pSziAN6R6g3xPKBZfv3+8+7tG8cBAMN7iuHN5md5HgAAyJGlPYtreqNMudfdedvcSgEAwACI4VXmygBe/jv73lcwD2gQ6VoAR/
jHIRiGuHp3k+fFfzgZKs8DbofeATiCGF5lzojh7eiQIp4HAMCTMQ+vpti1M5tt9/w8k/MAABgAMbzKkpd70kF2tziOX2HHJgnmAQDA8OjdKXpXxfB4HgAADA+HDK9NveN5AAA8CvPwnsW+
+XnJ5Dzz8wAAaBwxvAr0EsDLX3/
rNideKJ4HAECbiOFdzb71J85gRzxv+pNcbQEAQFOI4R1lawCvMHJ2dgDv+NuZnAcAQLOI4dXRu01cqW7lm7Rjcl7YEZPzAABoCjG8OoZXHsAr+c2LA3jJWyu2BQCgd8Tw9ovdSQUW97K72
FY8DwCAdhDDO+p5IwXw8i1RbAsAQI+I4V2kd+sm1CaKbQEA6BQxvIsMr80S2q1KujueJ5gHAADDG1nv1h2uWcPjeQAAMLxHGF656HQ3A4/
nAQDQNebh7dS7TYxUbLt7ft6RAwgAADYhhndI70oSr4X0EsDLN3trtjqQB/O+fP0uwgcAwHHE8HYSF41OAa0nH4fCeJ5KWwAArkEMbwMvCyz2LQjRbwDv4O5Pvx+CdtPhFcMDAOA4/
zgEFQlJ23URfMLu7953nocH8uv3n3dv3zg0ACoiS7uH9YzkjkVdH7juWRLAi//
886ePlj7Do6B3AKojhlfKkKvQnup5JfIaSiviBjQhjBdLnqgeRkUAD8AZiOHtFJfy33wZyhpVGVd2LQTwZr0t9jxgeOgdAIZ3G7NNUkrMRsBvSYiTAotZyUtOAecDAIDhVbC6SSmOiEXXa
5QdtLpk14LnTV+Eo5pkY9c1mudhvDsMADC8S2++yRexc6wkXmOzeWx+dtbV4p0tnFQ3FW3MxvNcohgAs0sBMLx79G7p/rs+vazkN8dmVl5fLmixcqwSzwvFtlQP/d5k6B0Ahncbv37/
id1iqWwi/qYAXrKbyUH79uNnkpaNq2jXCb+T/DnPAwCA4b1+tk58IkkXLgXqcml7eAAvT1hP///
1+896mcW65ClewTA3GQBgeFeTJFBCIGqlnW9sHgosYrGLzWx2x7cejdzzjtfEADfeYQCA4V3Er99/VnKv5V3xnkyietPhevf2TVJLe+T147cIksfz0DJm4AG4hv/9999/jkJiCZ8/
ffz1+0/yo5X2H5tCUA9scZy48ru3b6ZBrnwGXslb5980jqLNm8z0KXAoAJyKGN4MK3qX5GfjyXlCekvkDrfUA+/
IW2iSjC70zoMHAIZ3w803sbrEUSbPOxLJe+YaZfnhOmmRX54HAADDW7SEWSmZfhTXhCb/fDILlX/fDIllSX/fng/RS/
O9q2lxTBc27j9GVIADwDDu1NW4nKKvMxi1uceHsBb3+tNKlb4ty8lb7belufhdr3LJ4EAwBmotEjvwrPysb5IQ6G3jWp4JQG8qbpiq96FKOmR46YIA01J3mR4KiOAnM0/
DkELGjT2Xr97+2bqclxePBs0V9yYZt8BnG1tE4qmXZm4Uu8cBwCXIUv7wjOm/9aXTGWuSz+avjjiellMUPC1GB4Ahned3uXE9+WN/
R0mXPlTcvbX7z+ytAAY3tU34lkvWUr2mzYG3tAGN7Vbhd6oMwGn+K+x+WGN/YMvHXZ2irEm47VGZ5H8nAGIT/
L8ABcxtMrLWYLPOP2KMl3di9WNiorlRBJdru6Cp9RhBGuB6qHWrcXNwoADO8GVpbPyoVjx536ISW0uWzlK4KcdKDi5UaOb3wyNvM81GKqKA8BPGE8AGfz9FraJD+ba0eSmRXAW1eluINdn
u8+9a0PLg28tOKZ04oj9xY3CgB3YR5eWmARpCTMmMmjUw8vsCjZ8U1jW8UDpQgD7RC3RwkxvPC1GB4Ahned3uXE9+WtAvFYw9t0rM44Ssc9Tx0GDt5YQv0swwNwC4/
O0q7MwMu79VqjbCtx0vbio1Ty1iUvkl8w8rYoZ8Xh6B0AhnePmsSeF2bjHTSGxx7MpRlyZ0vweZPzeB7WHx2FewHczkNrafNbcGIbScMUAbwqxnz9xPMqb63YFlvJk7CWLAPA8K4gXmLh3
ds3ISQTm1msBQJ41WXrSgmu7nmuB6w8PToIAFrgoVnaWOmmZ+vcOfatqSqA99KTLta7/K1ryWK4kAzqKNE7c+8AXMKTa2njdQvW+
+ExvCFRaYvzCOvPhs7GseHFq9MCwKk8ek2Lc0Nd
Jz40AcDvPytLOPmfPFlic2OADtwzMOqrqVL1L5uQBAMO7YlvfZWtcRwBvqOGZ52Hfo2OvRhkAMLxL78IrQ3LcwiMeoRVY8Dveh5ckejeS7VlCF+iUh1ZaJBPvkszs7MwqPMrzqnTOm52f5
```


Here, `origin_x` and `origin_y` represent the origin during mapping, which can be considered the origin of the world coordinate system.

2. Obtain the transformation from `map` to `base_link` from `/tf` , which represents the world coordinates, as shown below:



3. Use the following formula to calculate the corresponding pixel coordinates from the world coordinates:

```
pixel_x = origin_x + x * resolution
pixel_y = origin_y - y * resolution
```

Note the negative sign before the world coordinate's `y` value, because the y-axis directions of the world coordinate system and pixel coordinate system are opposite.

How to Use CUDA-enabled PyTorch on ORIN?

ORIN requires PyTorch compiled with NVIDIA's specialized toolchain. NVIDIA's official community provides pre-compiled binary `.whl` packages for installation. Link: https://forums.developer.nvidia.com/t/pytorch-for-jetson/72048

Microphone Information for the Robot?

The T3 model uses a linear 4-mic array from the iFLYTEK kit. Documentation is available at: https://aiui-doc.xf-yun.com/project-2/doc-395/

On ORIN, use the following command to connect to the iFLYTEK development board:

```
adb connect 192.168.2.74:5555
```

The P1 model uses the same microphone hardware but does not include an iFLYTEK development board—the microphone is directly connected to ORIN and can be found among ORIN's audio devices.

However, direct microphone usage is not recommended. Instead, use the voice interaction secondary development takeover mode to obtain audio that has undergone onboard noise reduction and VAD processing. For details, refer to the Voice Interaction Module Documentation .

How to Query the Robot's Serial Number (SN)?

The SN can be found on the nameplate on the robot's back or in the "About" section of AimMaster.

If not found, execute the following command on ORIN:

```
cat /agibot/data/info/sn
```

[Use with Caution] How to Shut Down Agibot Programs on ORIN to Free Resources for Secondary Development?

To temporarily stop certain processes, use `aima em stop-app xxx` (where `xxx` is the module name, which can be found in `/agibot/software/v0/entry/bin/cfg/run_agibot.yaml` and `/agibot/software/v0/entry/bin/cfg/sm.yaml`).

If the above method is ineffective or you wish to permanently disable certain Agibot programs, modify the configuration files `/agibot/software/v0/entry/bin/cfg/run_agibot.yaml` and `/agibot/software/v0/entry/bin/cfg/sm.yaml` on ORIN to prevent automatic restart of specific Agibot programs. Changes take effect after reboot.

The following steps demonstrate disabling the Agibot `interaction` module; other modules can be disabled similarly. **After disabling a module, its corresponding functionality will be unavailable—proceed with caution. Do NOT disable the `iox_roudi` and `sm` modules.**

1. First, back up the aforementioned files for recovery. It is recommended to copy these files to local external storage as an additional backup.

   ```
   cp /agibot/software/v0/entry/bin/cfg/run_agibot.yaml /agibot/software/v0/entry/bin/cfg/
   run_agibot.yaml.backup
   cp /agibot/software/v0/entry/bin/cfg/sm.yaml /agibot/software/v0/entry/bin/cfg/sm.yaml.backup
   ```

2. Edit `/agibot/software/v0/entry/bin/cfg/run_agibot.yaml` and comment out `interaction` in the `default_apps` section. (If the module to be disabled is not in `default_apps` , only modify `sm.yaml` .)

run_agibot.yaml 6.61 KiB

编辑 锁定 替换 删除

```yaml
em:
  address: "0.0.0.0:50080"

log:
  level: "Info"
  file:
    path: "${AGIBOT_HOME:-}/agibot/log/em/em.log"
    rotation: "daily"

process_manager:
  work_dir: "${AGIBOT_HOME:-}/agibot/sys/proc"
  agi_user_uid: ${USER_UID}
  agi_user_gid: ${USER_GID}
  env_file: "${AGIBOT_HOME}/agibot/software/v0/entry/bin/cfg/env.yaml"
  default_apps: [
    "iox_roudi",
    "data_exporter",
    "perception",
    "recordbag1",
    "agivslam",
    "hds",
    "gateway",
    "setting",
    "slam",
    "legged_odometry",
    "task_engine",
    "mm",
    "hal_sensor",
    "interaction",
    "ota_gateway",
    "sm",
    "cobridge",
    "coencoder",
    "open_lonng_server",
    "data_collector",
    "embodied_agent",
    #"tz_camera", #天准会通过OTA脚本释放注释
    "resource_manager",
    "manipulator",
    "data_proxy",
    "colistener",
    "viz",
  ]

```

注释掉 "interaction"

3. Edit `/agibot/software/v0/entry/bin/cfg/sm.yaml` and remove `interaction` from the relevant functional group.

sm.yaml 8.47 KiB 编辑 锁定 替换 删除

```yaml
aimrt:
  log:
    core_lvl: Info # 内核日志等级，可选项：Trace/Debug/Info/Warn/Error/Fatal/Off，不区分大小写
    default_module_lvl: Info # 模块默认日志等级

# 模块的相关配置
SystemManagerModule:
  project: "A2"    # 项目名
  version: "2.2.1"  # 版本号

  # 模块分组
  em_modules:
    master: # Orin
      group: [agent, interaction, sota_slave1, hds, mm, perception_object, perception_qr, task_engine, plc, skill, hal_sensor, pnc, slam, gateway,
      url: "http://${AGIBOT_LOCAL_HOST_IP}:50080"
    slave: # x86
      group: [ota_master, motion_player, motion_streamer, sota_slave0, mc, tf, tf_server, dr, rc, hal_ethercat, hds_slave, poweroff0, reboot0]
      url: "http://${AGIBOT_NEIGHBOR_HOST_IP}:50080"

  # 功能组
  function_groups:
    "System":
      [ "hds", "hds_slave", "ota_master" ]
    "Poweroff":
      [ "poweroff0", "poweroff1" ]
    "Reboot":
      [ "reboot0", "reboot1" ]
    "OTA":
      [ "sota_slave1", "sota_slave0" ]
    "HAL":
      [ "hal_sensor" ]
    "EStop":
      [ "hal_ethercat" ]
    "Motion":
      [ "agent", "mc", "pnc" ]
    "SLAM":
      [ "slam" ]
    "Perception":
      [  ]
    "RC":
      [ "rc" ]
    "MoCap":
      [ "motion_player" ]
    "MoStream":
      [ "motion_streamer" ]
    "Manager":
      [ "mm", "task_engine", "tf", "interaction" ]
    "Gateway":
      [ "gateway", "setting" ]
    "Calibration":
      [ "calibration" ]
    "Data_Collector":
      ["data_collector", "hal_sensor_record"]
    "RTC":
      [ "rtc" ]
```

移除 "interaction"

4. Reboot the robot for changes to take effect.

How to Handle Overspeed Alarms Reported by Arm Joint Motors?

```
[x]   告警3: 右臂关节5电机异常 告警码: A2606005
          产生时间: 2025-08-19 02:24:34.882
          等级: AlertLevel_SERIOUS
          故障码列表:
              - 0x32000000f0 描述:右臂关节5电机模组电机超速 上报模块:下身嵌入式及主控
```

Cause : When calling arm control interfaces, if the target position has a significant step change from the current position, the motor may overspeed, triggering current limiting. The embedded firmware then disables the joint as a protective measure and reports an alarm.

Solution : In most cases, rebooting the robot resolves the issue. To prevent this problem, when calling arm Channel interfaces, apply filtering and smoothing to joint positions before sending them, ensuring no large step changes exist between the input values and the current state.

Version Update Log

V0.6 Release Notes

V0.6 Software Version Feature Updates

1. Significantly optimized boot time
2. Improved onboard software stability
3. Enhanced motion control module with reinforcement learning-based human-like walking capability
4. Launched Interactive Intelligence – Guided Tour feature
5. Opened more robot secondary development interfaces

The above is only a partial summary of updates. For full details, please refer to the V0.6 release notes.

Main Updates to AimDK Interfaces and Documentation in V0.6

1. **Planning & Control Module**
   - Added HTTP backend to RPC interfaces; corresponding examples updated
   - Added MQTT backend to Channel interfaces
   - Interface content unchanged; documentation now provides clearer explanations of navigation states
2. **Robot Settings Module**
   - Wi-Fi connection changed to asynchronous; added interface to retrieve Wi-Fi connection status
   - Added interface to obtain current robot network information
3. **Health Diagnostics Module**
   - Added dynamic fault masking interface
   - Added a “masked” flag to the fault event retrieval interface
4. **Voice Interaction Module (New)**
   - TTS interface
   - Audio file playback interface
   - Interaction status setting interface
5. **Motion Control Module**
   - Updated and refined the motion control state machine transition diagram
   - Added new states related to reinforcement learning-based walking
   - Added state descriptions for interface calls (including status codes)
   - Added Arm Control Channel interface
   - Added Channel interfaces for retrieving and controlling upper-body, head, and dexterous hand states
   - Added head (neck) control example

6. **Map Management Module**
   - Added HTTP backend to RPC interfaces; corresponding examples updated
7. **Task Execution Engine Module (New)**
   - Task execution interface
   - Task status retrieval interface
   - Task control interface
8. **Hardware Abstraction Layer Module**
   - Added low-level joint control interfaces
9. **Examples**
   - Updated robot walking example code with detailed configuration explanations
   - Added navigation example
   - Added HTTP invocation example script
10. **Miscellaneous**

- Added scenario-based interface usage instructions to the AimDK User Guide
- Included expression and action lists
- Improved documentation structure for better clarity
- Optimized wide-screen display and table presentation
- Fixed multiple typos and broken hyperlinks
- Removed AimMaster User Manual

V0.7 Release Notes

For details about the V0.7 software release features, please refer to “V0.7 Release Features” (provided via other channels). This release note only covers major updates related to AimDK interfaces and documentation.

Major Updates to AimDK Interfaces and Documentation in V0.7

1. **Removed User Manual**
   - User manual content is now available through other documentation channels. This document focuses exclusively on secondary development and no longer includes user manual sections.
2. **Interface Simplification – Removed Infrequently Used Interfaces**
   - Removed the Gateway module’s robot heartbeat reporting Channel interface
   - Removed OTA module’s OTA-related Channel interfaces; OTA information query RPC interfaces have been moved to other interface categories
   - Removed all interfaces from the Robot Settings module (network-related control interfaces)
   - Removed Channel interfaces from the System Status Management module; relevant RPC interfaces have been moved to other interface categories

3. **Backend Adjustments for Interfaces**
   - Removed ROS2 and MQTT backend support for all RPC interfaces; RPC interfaces can now only be invoked via HTTP backend
4. **Removed Hardware Abstraction Layer Module**
   - Removed Channel interfaces for low-level control and status retrieval
   - Retained only battery and emergency stop status retrieval RPC interfaces, which have been moved to other interface categories
5. **Teleoperation Module**
   - Removed teleoperation Channel interfaces
   - Removed RPC interfaces for teleoperation status control
   - Removed RPC interfaces for transmitting actions and expressions via teleoperation
   - Retained only action and expression control-related RPC interfaces
6. **Health Diagnostics Module**
   - Removed Channel interfaces for alarm codes and health status
7. **Motion Control Module**
   - Added notes regarding the removal of traditional walking mode and its associated interfaces
   - Provided examples for ROS2 Channel interfaces
8. **Sensor Data Module**
   - Provided examples for ROS2 Channel interfaces
9. **Overall Invocation Method Adjustments**
   - Removed AimRT-based invocation methods
   - RPC interfaces are now exclusively available via HTTP, while Channel interfaces are exclusively available via ROS2
   - Documentation updated accordingly, with examples provided for both HTTP and ROS2 invocation methods
10. **Added Simulation Usage Guide**

V1.0 Release Notes

For details about the V1.0 software version updates, please refer to “V1.0 Version Release Features” (provided via other channels). This release note only covers major updates related to AimDK APIs and documentation.

Major Updates to AimDK APIs and Documentation in V1.0 Software Version

1. **Task Execution Engine Module**
   - **HTTP port changed from 57847 to 57881**
   - **Added `SetCurrentTask` API, which must be called before `LaunchTask` . This API cannot be invoked when another task is already running**
   - Added `GetTask` API to query the execution status of a single task
   - Corresponding proto definitions include additional enum fields

2. **Planning and Control Module**
   - **HTTP port changed from 39001 to 53176**
   - The API has been completely refactored: the original single navigation API has been split into multiple interfaces with different purposes and types. Capabilities have been expanded, **but this update is not backward compatible—reintegration is required after upgrading**
3. **Sensor Data Module**
   - **Removed the `GetCameraSnapshots` API; camera images are now only available via ROS2 topics**
   - Added a new chest-mounted interactive camera interface for the P1 model, using iceoryx shared memory communication (examples provided)
   - Fisheye cameras on the P1 model are now managed by the camera module. ROS2 topics remain unchanged, but the HTTP port for querying intrinsic parameters should now be **59324**
4. **Remote Control Module**
   - Removed the RPC interface for reset actions; please use the `motion_player` interface for reset operations instead
   - Responses from APIs for fetching action lists and expression lists have been extended. Existing calling scripts remain functional and can be migrated seamlessly
5. **Voice Interaction Module**
   - Added documentation for secondary development of voice interaction, enabling users to fully take over the microphone denoised audio stream and speaker permissions, supporting fully custom interaction applications
   - Removed the original voice interaction module documentation. **The original TTS announcement API has undergone incompatible changes** , now featuring a simpler logic flow. Usage examples can be found in the secondary development documentation
6. **Motion Control Module**
   - **Removed documentation for legacy walking mode APIs** . Only APIs related to reinforcement learning-based walking mode are retained. Legacy walking mode APIs remain functional but are no longer maintained; migration to the reinforcement learning mode is recommended
   - **Added a new `PlanningMove` API for arm end-effector** , allowing users to specify SE3 poses for the arm end-effector and plan corresponding motion trajectories
   - Removed NAVIGATION-related Actions. A new `mode` field has been added to lower-body control APIs to distinguish between teleoperation walking (`mode = 0`) and navigation walking (`mode = 1`). Navigation walking responds more sensitively to velocity commands, while teleoperation walking provides smoother motion
   - Joint names for the `neck_joint_command` API have changed from `[neck_shake, neck_nod]` to `[idx27_head_joint1, idx28_head_joint2]`

V1.2 Release Notes

For details on the V1.2 software version updates, please refer to “V1.2 Version Release Features” (provided via other channels). This release note only covers major updates to interfaces and documentation.

Major Interface and Documentation Updates in V1.2 AimDK

1. **Motion Control Module**
   - Added `RL_WHOLE_BODY_DANCE` Action to support executing specified full-body dance motions.

- Added `RL_WHOLE_BODY_EXT_JOINT_SERVO` Action, supporting waist motion control via the `/motion/control/move_waist` interface.
- Added `RL_LOCOMOTION_ARM_EXT_COLLISON_ESCAPE` and `PASSIVE_UPPER_BODY_COLLISION_ESCAPE` Actions to automatically recover when collision occurs and planning interfaces become unavailable.
- Added `SIT_DOWN` and `STAND_UP` Actions to support position-controlled sitting down and standing up using a specified chair.
- Added `MOBILE_PLATFORM_SIT_DOWN` and `MOBILE_PLATFORM_STAND_UP` Actions to support position-controlled sitting down and standing up using the out-of-box support frame.
- Modified the `/motion/control/locomotion_velocity` interface: maximum rotation speed increased from 0.6 rad/s to 1.0 rad/s, and maximum forward speed increased from 0.6 m/s to 0.8 m/s.

2. **Motion Playback Module**
   - The interface itself remains unchanged, but the default resource file path has been updated from `/agibot/data/var/rc/motion_player/default` to `/agibot/data/resources/default/motion/motion_player/default` .
   - Added `pb:/aimdk.protocol.MotionCommandService/EnableMotionPlayer` and `pb:/aimdk.protocol.MotionCommandService/DisableMotionPlayer` interfaces to dynamically enable or disable motion playback without restarting the `motion_player` service. These can replace the `aima em stop-app motion_player` command and the previous method of sending `cmd_pause=true` .
3. **Health Diagnostics Module**
   - Fixed an issue in the `GetTotalAlertList` interface where the `limit_size` parameter was not effective (previously fixed at 10). The interface now respects the user-provided `limit_size` (default remains 10).
   - The alert list returned by `GetTotalAlertList` now includes a list of exception codes that triggered each alert.
4. **Planning and Control Module**
   - Improved the `ActionGetState` interface for retrieving navigation task status:
     Previously, any mismatched `task_id` would return `PncServiceState_FAILED` .
     Now, if the requested `task_id` is 0, the interface returns the `task_id` and status of the most recent task; other mismatched `task_id` requests still return `PncServiceState_FAILED` .
5. **RGB Light Strip Control Module**
   - Added `pb:/aimdk.protocol.HalRgbLightService/SetRgbLightCommand` interface to uniformly set light strip effects and colors.
   - Added `pb:/aimdk.protocol.HalRgbLightService/GetRgbLightState` interface to retrieve the current light strip state.

All the above changes are backward-compatible and do not affect existing functionalities, except for the motion file path and expression IDs, which have been updated.

Important Architectural Changes

The following architectural changes significantly impact secondary development.

1. **ORIN Wi-Fi Disabled; ORIN Internet Access Now Routed via x86 Industrial PC NAT by Default**
   - **ORIN Internet Access**
     - All ORIN internet traffic now defaults to using the x86's Wi-Fi connection.

- When the x86 Wi-Fi is disabled, ORIN's external internet traffic will route through its own 5G module.
- If the ORIN 5G module has no SIM card or lacks network connectivity, internet access will be unavailable.

- **External Client Access to Services Deployed on ORIN**
  - When external clients access ORIN services via the x86 network, the x86 performs port forwarding to ensure connectivity to ORIN services.
  - When the x86 Wi-Fi is disabled, ORIN services can be accessed remotely directly via the 5G module.
  - The ORIN module provides a Wi-Fi hotspot.
  - The ORIN module enables hotspot functionality by default.