# Luna SDK Development Guide

| Version | Date | Revised By | Description of Changes | Remarks |
| --- | --- | --- | --- | --- |
| V1.0 | 2026-08-26 | Eleven | Initial Release | |

# 1. SDK Introduction

## 1.1 High-Level Application Protocol Interface

### 1.1.1 Overview

The robot receives user request commands through the WebSocket communication port `5000`, such as making the robot stand up, squat down, walk, etc. WebSocket is a real-time communication protocol that establishes a persistent connection between the robot and the user端 for fast and efficient transmission of control information and data.

### 1.1.2 Communication Protocol Format

When the robot receives commands from the client via WebSocket, data is transmitted using the JSON communication protocol. This approach offers significant advantages: WebSocket is a full-duplex communication protocol that establishes a real-time, low-latency connection between the client and server, particularly suitable for frequently interacting application scenarios. The JSON data protocol, with its concise and highly readable structure, ensures intuitive and clear data transmission, with cross-platform and cross-language compatibility. The combination of WebSocket and JSON is not only programming language-agnostic and applicable to various devices and systems, but also improves development flexibility and maintenance convenience.

- The request data format contains the following fields:
  - `accid`: The robot's unique serial number, identifying its unique identity;
  - `title`: The command name, prefixed with "`request_`";
  - `timestamp`: The timestamp when the command is sent, in milliseconds;

- `guid`: A unique command identifier used to distinguish different request commands. For synchronous interfaces, the `guid` value must be returned to the client in the "response_xxx" response message. After receiving the response message, the client can determine whether the command has been executed by comparing whether the `guid` value matches the value in the request command;
- `data`: Contains the content of the request. Depending on specific requirements, it may include multiple sub-fields to store the data required by the request command, such as parameters for executing actions, text content for sending messages, etc.;
- Example:

```
代码块
{
  "accid": "HU_D02_001", # Robot's unique serial number, identifying its unique identity
  "title": "request_xxx",   # Command name, prefixed with "request_"
  "timestamp": 1672373633989, # Timestamp when the command is sent, in milliseconds
  "guid": "746d937cd8094f6a98c9577aaf213d98", # Unique command identifier, used to distinguish different request commands
  "data": {}  # Contains the content of the request command
}
```

- The response data format contains the following fields:
  - `accid`: The robot's unique serial number, identifying its unique identity;
  - `title`: The command name, prefixed with "`response_`";
  - `timestamp`: The timestamp when the command is sent, in milliseconds;
  - `guid`: Same as the `guid` value of the corresponding request command;
  - `data`: Should contain at least one "result" sub-field, used to store the execution result data of the request command. If needed, it may also include other sub-fields, such as error codes, error messages, and other information describing the operation result;
  - Example:

```
代码块
{
  "accid": "HU_D02_001",   # Robot's unique serial number, identifying its unique identity
  "title": "response_xxx",  # Command name, prefixed with "response_"
  "timestamp": 1672373633989, # Timestamp when the command is sent, in milliseconds
```

```
  "guid": "746d937cd8094f6a98c9577aaf213d98", # Same as the guid value of the corresponding request command
  "data": { # Contains the specific data content of the response command
    "result": "success"  # "result" stores whether the request command was processed successfully, its value is: "success or fail_xxx"
  }
}
```

- Message Push: This is the process by which the robot proactively sends information to the client. This information can include the robot's serial number, current running status, executed operations, and other data. By promptly sending this information to the client, the robot can help the client better understand its working status, thereby better utilizing the services it provides. Its data format contains the following fields:
  - `accid` : The robot's unique serial number, identifying its unique identity;
  - `title` : The command name, prefixed with " `notify_` ";
  - `timestamp` : The timestamp when the message is sent, in milliseconds;
  - `guid` : The guid value of the message, uniquely identifying this message;
  - `data` : Contains the message data content. Depending on specific requirements, it may include multiple sub-fields to store the data required by the request command;
  - Example:

代码块

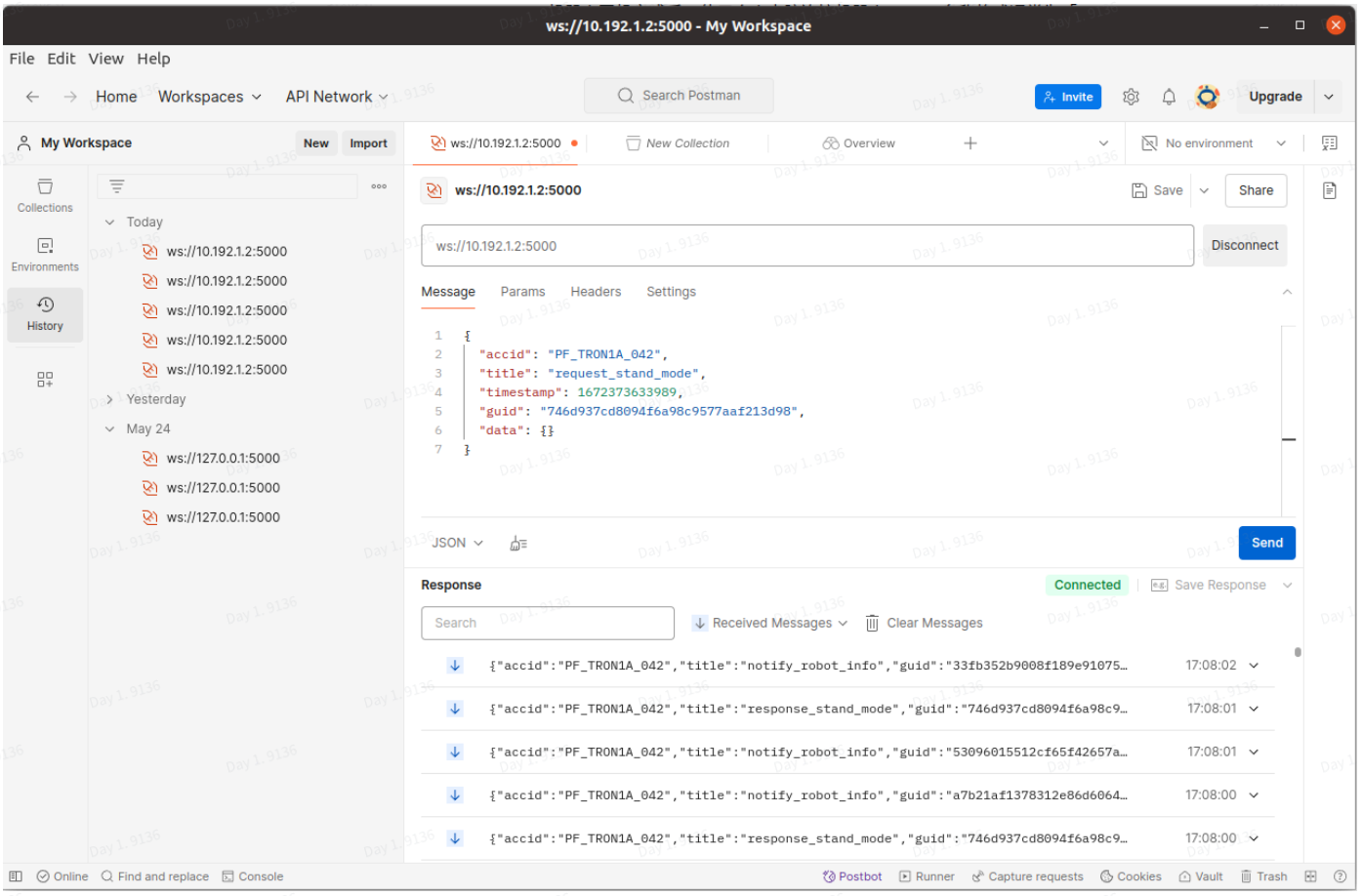
```
{
  "accid": "HU_D02_001",   # Robot's unique serial number, identifying its unique identity
  "title": "notify_xxx",  # Message name, prefixed with "notify_"
  "timestamp": 1672373633989, # Timestamp when the message is sent, in milliseconds
  "guid": "746d937cd8094f6a98c9577aaf213d98", # The guid value of the message, uniquely identifying this message
  "data": { } # Contains the message data content
}
```

### 1.1.3 Communication Testing Method

Postman is a popular API development environment that can be used to test WebSocket interfaces. To test WebSocket interfaces using Postman, follow these steps:

- Install Postman, download address: https://www.postman.com/downloads/?utm_source=postman-home;

- Open Postman and create a WebSocket request;
- Connect to the robot's wireless network
  - After the robot is powered on, use your personal computer to connect to the robot's Wi-Fi, the name format is usually "HU_D02_xxx"
  - Enter the Wi-Fi password: `12345678`
- Enter the WebSocket interface address in the request URL, for example, "ws://10.192.1.2:5000";
- In "Message", enter the command request to be sent;
- Click the "Send" button to send the request command;
- After sending the command, you can receive the response message from the server. Use Postman's response window to view the data returned by the server and check whether it meets the expected result.




# 1.2 Basic Function Protocol Interfaces

## 1.2.1 Set Audio Prompt Language

### 1.2.1.1 Request: request_set_audio_prompts_language

This protocol is used to set the robot's audio prompt language. After a successful request, the robot's subsequent voice prompts will use the specified language, and a switching confirmation

prompt will be played to help the user confirm that the setting has taken effect. Request parameter, language values:

| language | Meaning | User Perception |
| --- | --- | --- |
| cn | Chinese audio prompt | Subsequent prompts use Chinese, and a Chinese switching confirmation prompt is played |
| en | English audio prompt | Subsequent prompts use English, and an English switching confirmation prompt is played |

代码块

```
{
  "accid": "HU_D04_01_001",
  "title": "request_set_audio_prompts_language",
  "timestamp": 1672373633989,
  "guid": "746d937cd8094f6a98c9577aaf213d98",
  "data": {
    "language": "cn"
  }
}
```

### 1.2.1.2 Response: response_set_audio_prompts_language

代码块

```
{
  "accid": "HU_D04_01_001",
  "title": "response_set_audio_prompts_language",
  "timestamp": 1672373633989,
  "guid": "746d937cd8094f6a98c9577aaf213d98",
  "data": {
    "result": "success"
  }
}
```

#### result Values

| result | Meaning | User Prompt Suggestion |
| --- | --- | --- |
| success | Setting successful | Audio prompt language has been switched |

| | | |
|---|---|---|
| fail_no_language | Target language not provided | Please select an audio prompt language |
| fail_invalid_language | Language value not supported | Currently only Chinese and English are supported |
| fail_write_robot_info | Failed to save setting | Setting failed, please try again later |
| fail_play_audio_prompt_service_not_ready | Audio function not ready | Audio function not ready, please try again later |
| fail_play_audio_prompt_call | Switching confirmation prompt request failed | Switching confirmation failed, please retry |
| fail_play_audio_prompt | Switching confirmation prompt playback failed | Confirmation prompt not heard, please retry |

#### 1.2.1.3 Message Push: none

### 1.2.2 Connect to Wi-Fi Hotspot

#### 1.2.2.1 Request: request_connect_wifi

This protocol is used to send a request to the robot router, instructing the router to connect to a Wi-Fi hotspot with the specified SSID and return the connection result.

代码块

```
{
  "accid": "HU_D04_01_001",
  "title": "request_connect_wifi",
  "timestamp": 1672373633989,
  "guid": "746d937cd8094f6a98c9577aaf213d98",
  "data": {
      "wifi_band": 0,  # Wi-Fi band: 0=5GHz, 1=2.4GHz
      "wifi_ssid": "Limx-Guests",  # Target Wi-Fi SSID (Wi-Fi name), case-sensitive, must match the actual hotspot
      "wifi_password": "LimX2024",  # Target Wi-Fi password, password for WPA2-PSK encrypted network
      "router_admin_password": "12345678"  # Robot router administrator password
  }
}
```

#### 1.2.2.2 Response: response_connect_wifi

```
代码块
  "accid": "HU_D04_01_001",
  "title": "response_connect_wifi",
  "timestamp": 1672373633989,
  "guid": "746d937cd8094f6a98c9577aaf213d98",
  "data": {
      "result": "success"  # success
                           # fail_no_wifi_band: Wi-Fi band not specified
                           # fail_no_wifi_ssid: SSID not specified
                           # fail_no_wifi_password: password not specified
                           # fail_no_router_admin_password: administrator
password not specified
  }
}
```

### 1.2.3 Query Wi-Fi Connection Status

#### 1.2.3.1 Request: request_wifi_connection_status

This protocol is used by the client to send a Wi-Fi connection status query request to the robot router. After receiving the request, the router feeds back the core status information of the currently connected Wi-Fi, including the associated SSID, signal strength, and connection result, supporting the client in real-time perception of the device's network connection status. The client initiating a Wi-Fi connection status query must carry the robot router administrator password to complete identity verification.

```
代码块
{
  "accid": "HU_D04_01_001",
  "title": "request_wifi_connection_status",
  "timestamp": 1672373633989,
  "guid": "746d937cd8094f6a98c9577aaf213d98",
  "data": {
      "router_admin_password": "12345678"  # Robot router administrator
password
  }
}
```

#### 1.2.3.2 Response: response_wifi_connection_status

After receiving the query request, the robot router returns the current actual Wi-Fi connection status for client-side parsing, display, or subsequent business processing.

```
代码块
```

```
{
  "accid": "HU_D04_01_001",
  "title": "response_wifi_connection_status",
  "timestamp": 1672373633989,
  "guid": "746d937cd8094f6a98c9577aaf213d98",
  "data": {
      "ssid": "Limx-Guests",
      "signal": -56,        # Unit: dBm
      "result": "success"   # success
                            # fail_disconnected
  }
}
```

### 1.2.4 Enter Ready State

The robot slowly assumes a ready posture.

#### 1.2.4.1 Request: request_prepare

Controls the robot to enter a standing state.

代码块

```
{
  "accid": "HU_D04_01_001",
  "title": "request_prepare",
  "timestamp": 1672373633989,
  "guid": "746d937cd8094f6a98c9577aaf213d98",
  "data": { }
}
```

#### 1.2.4.2 Response: response_prepare

代码块

```
{
  "accid": "HU_D04_01_001",
  "title": "response_prepare",
  "timestamp": 1672373633989,
  "guid": "746d937cd8094f6a98c9577aaf213d98",
  "data": {
      "result": "success"  # success, fail_motor: motor error
  }
}
```

### 1.2.5 Control Robot Walking

#### 1.2.5.1 Enter Walking Mode

The robot enters walking mode and can receive velocity commands.

#### 1.2.5.2 Request: request_set_walk_mode

代码块

```
{
  "accid": "HU_D04_01_001",
  "title": "request_set_walk_mode",
  "timestamp": 1672373633989,
  "guid": "746d937cd8094f6a98c9577aaf213d98",
  "data": {}
}
```

#### 1.2.5.3 Response: response_set_walk_mode

代码块

```
{
  "accid": "HU_D04_01_001",
  "title": "response_set_walk_mode",
  "timestamp": 1672373633989,
  "guid": "746d937cd8094f6a98c9577aaf213d98",
  "data": {
      "result": "success"  # success, fail_motor: motor error
  }
}
```

### 1.2.6 Control Robot Walking

In mobile operation mode, control the robot to walk through this protocol (requires > 10 Hz command issuance). Please note that in whole-body operation mode, this protocol interface is invalid.

#### 1.2.6.1 Request: request_set_walk_vel

代码块

```
{
  "accid": "HU_D04_01_001",
  "title": "request_set_walk_vel",
  "timestamp": 1672373633989,
```

```
    "guid": "746d937cd8094f6a98c9577aaf213d98",
    "data": {
      "x": 0.0,    #  Forward/backward velocity ratio, value range [-1, 1]
      "y": 0.0,    #  Lateral walking velocity ratio, value range [-1, 1]
      "yaw": 0.0   #  Rotational angular velocity ratio, value range [-1, 1]
    }
  }
```

#### 1.2.6.2 Response: response_set_walk_vel

This message is returned when command execution fails; no response is returned on successful execution.

代码块

```
{
  "accid": "HU_D04_01_001",
  "title": "response_set_walk_vel",
  "timestamp": 1672373633989,
  "guid": "746d937cd8094f6a98c9577aaf213d98",
  "data": {
    "result": "fail_motor"  # fail_imu: IMU error, fail_motor: motor error
  }
}
```

#### 1.2.6.3 Message Push: none

### 1.2.7 Enter Damping Mode

All robot motors stop active motion, with noticeable damping when swung.

#### 1.2.7.1 Request: request_damping

代码块

```
{
  "accid": "HU_D04_01_001",
  "title": "request_damping",
  "timestamp": 1672373633989,
  "guid": "746d937cd8094f6a98c9577aaf213d98",
  "data": {}
}
```

#### 1.2.7.2 Response: response_damping

```
代码块
{
    "accid": "HU_D04_01_001",
    "title": "response_damping",
    "timestamp": 1672373633989,
    "guid": "746d937cd8094f6a98c9577aaf213d98",
    "data": {
        "result": "success" # fail_motor: motor error
    }
}
```

### 1.2.8 Enter Zero Torque Mode

All robot motors stop active motion, with no damping when swung.

#### 1.2.8.1 Request: request_zero_torque

```
代码块
{
    "accid": "HU_D04_01_001",
    "title": "request_zero_torque",
    "timestamp": 1672373633989,
    "guid": "746d937cd8094f6a98c9577aaf213d98",
    "data": {}
}
```

#### 1.2.8.2 Response: response_zero_torque

```
代码块
{
    "accid": "HU_D04_01_001",
    "title": "response_zero_torque",
    "timestamp": 1672373633989,
    "guid": "746d937cd8094f6a98c9577aaf213d98",
    "data": {
        "result": "success" # fail_motor: motor error
    }
}
```

### 1.2.9 Enter Standing Command

> 💡 Interface Function: Starts robot operation. After the robot is powered on, calling this interface transitions the robot into the standing state.

Parameter mode: [lying: robot is lying on the ground　hanging: robot is suspended ]

Return: Returns after the robot has successfully stood up.

Requirement Link: 【PRD】启动与回收机器人产品需求文档_V1.0

#### 1.2.9.1 Request: request_standup

代码块

```
{
  "accid": "HU_D04_01_001",
  "title": "request_standup",
  "timestamp": 1672373633989,
  "guid": "746d937cd8094f6a98c9577aaf213d98",
  "data": {
      "mode": "lying" // "lying": robot is currently lying/sitting  or
                      // "hanging": robot is currently suspended
                      // If there is no "mode" field, the default robot state is "sitting"
  }
}
```

#### 1.2.9.2 Response: response_standup

代码块

```
{
  "accid": "HU_D04_01_001",
  "title": "response_standup",
  "timestamp": 1672373633989,
  "guid": "746d937cd8094f6a98c9577aaf213d98",
  "data": {
      "result": "success" # fail_motor: motor error
                          # fail_invalid_cmd: parameter error
                          # fail_invalid_mode: robot state error
                          # fail_timeout: execution timeout error
  }
}
```

### 1.2.10 Enter Lying Down Command

#### 1.2.10.1 Request: request_lie_down

💡 This interface can be called in Walk state

代码块

```
{
  "accid": "HU_D04_01_001",
  "title": "request_lie_down",
  "timestamp": 1672373633989,
  "guid": "746d937cd8094f6a98c9577aaf213d98",
  "data": {}
}
```

#### 1.2.10.2 Response: response_lie_down

代码块

```
{
  "accid": "HU_D04_01_001",
  "title": "response_lie_down",
  "timestamp": 1672373633989,
  "guid": "746d937cd8094f6a98c9577aaf213d98",
  "data": {
      "result": "success" # fail_motor: motor error
  }
}
```

### 1.2.11 Robot Dance

#### 1.2.11.1 Switch Robot to Dance Mode

##### 1.2.11.1.1 Request: request_enter_dance_mode

代码块

```
{
  "accid": "HU_D04_01_001",
  "title": "request_enter_dance_mode",
  "timestamp": 1672373633989,
  "guid": "746d937cd8094f6a98c9577aaf213d98",
  "data": {
    # 0: Exit dance mode
    # 1: Enter dance mode
    "mode": 0
  }
```

```
}
```

#### 1.2.11.1.2 Response: response_enter_dance_mode

代码块

```
{
  "accid": "HU_D04_01_001",
  "title": "response_enter_dance_mode",
  "timestamp": 1672373633989,
  "guid": "746d937cd8094f6a98c9577aaf213d98",
  "data": {
      "result": "success" # fail_motor
  }
}
```

### 1.2.11.2 Get Dance List

#### 1.2.11.2.1 Request: request_get_dance_list

代码块

```
{
  "accid": "HU_D04_01_001",
  "title": "request_get_dance_list",
  "timestamp": 1672373633989,
  "guid": "746d937cd8094f6a98c9577aaf213d98",
  "data": {}
}
```

#### 1.2.11.2.2 Response: response_get_dance_list

代码块

```

    "accid": "HU_D04_01_001",
    "title": "response_get_dance_list",
    "guid": "746d937cd8094f6a98c9577aaf213d98",
    "timestamp": 1672373633989,
    "data": {
        "result": "success",
        "code": 0,
        "dances": [
            {
                "id": "DAN-14",
```

```
                "index": 0,
                "name": "\u70ed\u70c8",
                "english_name": "One and Only Dance",
                "rc_mapping": "one_and_only_dance",
                "duration": 10
            },
            {
                "id": "DAN-08",
                "index": 1,
                "name": "\u4f4e\u4fd7\u5c0f\u8bf4",
                "english_name": "Pulp Fiction Dance",
                "rc_mapping": "pulp_fiction_dance",
                "duration": 10
            }
        ]
    }
}
```

### 1.2.11.3 Robot Dancing

#### 1.2.11.3.1 Request: request_dance

> 💡 Execution prerequisite: Currently in action library mode

代码块

```
{
  "accid": "HU_D04_01_001",
  "title": "request_dance",
  "timestamp": 1672373633989,
  "guid": "746d937cd8094f6a98c9577aaf213d98",
  "data": {
    "name": "one_and_only_dance"  # Dance name from the rc_mapping field
  }
}
```

#### 1.2.11.3.2 Response: response_dance

代码块

```
{
  "accid": "HU_D04_01_001",
  "title": "response_dance",
  "timestamp": 1672373633989,
```

```
    "guid": "746d937cd8094f6a98c9577aaf213d98",
    "data": {
        "result": "success" # fail_motor
    }
}
```

#### 1.2.11.3.3 Message Push: notify_dance

This message is pushed after the dance is completed or if execution fails during the process.

代码块

```
{
    "accid": "HU_D04_01_001",
    "title": "notify_dance",
    "timestamp": 1672373633989,
    "guid": "746d937cd8094f6a98c9577aaf213d98",
    "data": {
        "result": "success" # fail_motor
    }
}
```

## 1.2.12 Robot Action Library

### 1.2.12.1 Action Interruption

#### 1.2.12.1.1 Request: request_interrupt_action_joystick

代码块

```
{
    "accid": "HU_D04_01_001",
    "title": "request_interrupt_action_joystick",
    "timestamp": 1779355330784,
    "guid": "32cef03a-5563-4b21-9bbb-3e65a8c9ae9e",
    "data": {}
}
```

#### 1.2.12.1.2 Response:

代码块

```
{
    "accid": "HU_D04_01_001",
    "title": "response_interrupt_action_joystick",
    "guid": "32cef03a-5563-4b21-9bbb-3e65a8c9ae9e",
```

```
    "timestamp": 1779355330784,
    "data": {
      "result": "success"
    }
  }
```

### 1.2.12.2 Get Action Library Status

> Interface Description:
> (1) After the robot enters the action library, whether it is in action library/atomic execution/dance mode, "action_library_mode": "action_library"
> (2) When the robot is executing an atomic action or dancing, "action_library_state": "running"

#### 1.2.12.2.1 Request: request_get_action_library_status

代码块

```
{
  "accid": "HU_D04_01_001",
  "title": "request_get_action_library_status",
  "timestamp": 1672373633989,
  "guid": "746d937cd8094f6a98c9577aaf213d98",
  "data": {}
}
```

#### 1.2.12.2.2 Response: response_get_action_library_status

代码块

```
{
  "accid": "HU_D04_01_001",
  "title": "get_action_library_status",
  "timestamp": 1672373633989,
  "guid": "746d937cd8094f6a98c9577aaf213d98",
  "data": {
      "action_library_mode": "action_library" // or "remote_control"
      "action_library_state": "running"       //or "idle"
      "result": "success" # fail_motor
  }
}
```

### 1.2.12.3 Switch Robot to Action Library Mode

#### 1.2.12.3.1 Request: request_set_motion_engine

代码块

```
{
  "accid": "HU_D04_01_001",
  "title": "request_set_motion_engine",
  "timestamp": 1672373633989,
  "guid": "746d937cd8094f6a98c9577aaf213d98",
  "data": {
    # 0: Exit action library mode
    # 1: Enter action library mode
    "mode": 0
  }
}
```

#### 1.2.12.3.2 Response: response_set_motion_engine

代码块

```
{
  "accid": "HU_D04_01_001",
  "title": "response_set_motion_engine",
  "timestamp": 1672373633989,
  "guid": "746d937cd8094f6a98c9577aaf213d98",
  "data": {
      "result": "success" # fail_motor
  }
}
```

### 1.2.12.4 Execute Action Library

#### 1.2.12.4.1 Request: request_action_sync

代码块

```
{
  "accid": "HU_D04_01_001",
  "title": "request_action_sync",
  "timestamp": 1672373633989,
  "guid": "746d937cd8094f6a98c9577aaf213d98",
  "data": {
      "name": "one_and_only_dance,this_way_please"   ,
                                    # name: multiple dances/multiple
actions/mix of dances and actions, separated by ","
```

```
        "music": "bgm1.wav,bgm2.wav,"   # Optional, only supports .wav format files
    }
}
```

#### 1.2.12.4.2 Response: response_action_sync

代码块

```
{
    "accid": "HU_D04_01_001",
    "title": "response_action_sync",
    "timestamp": 1672373633989,
    "guid": "746d937cd8094f6a98c9577aaf213d98",
    "data": {
        "result": "success" # fail_motor
    }
}
```

### 1.2.12.5 Get Action Library List

#### 1.2.12.5.1 Request: request_get_atomic_motion_list

代码块

```
{
    "accid": "HU_D04_01_001",
    "title": "request_get_atomic_motion_list",
    "timestamp": 1672373633989,
    "guid": "746d937cd8094f6a98c9577aaf213d98",
    "data": {}
}
```

#### 1.2.12.5.2 Response: response_get_atomic_motion_list

代码块

```
{
    "accid": "HU_D04_01_001",
    "title": "response_get_atomic_motion_list",
    "guid": "746d937cd8094f6a98c9577aaf213d98",
    "timestamp": 287883835,
    "data": {
        "result": "success",
        "motion_list": [
```

```
            {
                "id": "MON-01",
                "rc_mapping": "greeting1",
                "duration": 7,
                "motion_index": 0,
                "motion_name_cn": "侧身打招呼",
                "motion_name_en": "Greeting1"
            },
            {
                "id": "MON-02",
                "rc_mapping": "greeting2",
                "duration": 9,
                "motion_index": 1,
                "motion_name_cn": "抬腿打招呼",
                "motion_name_en": "Greeting2"
            }
            ......
        ],
        "count": 2
    }
}
```

### 1.2.13 Global Message Protocol Interface

### 1.2.14 Robot Status Information

Robot status information is periodically reported through this protocol, including the following content:

- `accid` : Robot serial number
- `title` : notify_robot_info
- `timestamp` : The timestamp when the message is sent, in milliseconds
- `guid` : The guid value of the message, uniquely identifying this message
- `data` : Contains the message content, example:

代码块

```
{
  "accid": "HU_D04_01_001",
  "title": "notify_robot_info",
  "timestamp": 1672373633989,
  "guid": "746d937cd8094f6a98c9577aaf213d98",
  "data": {
    "result": []
  }
```

```
9   }
```

### 1.2.14.1 Battery Data

代码块

```
{
  "accid": "HU_D04_01_001",
  "title": "notify_robot_info",
  "timestamp": 1672373633989,
  "guid": "746d937cd8094f6a98c9577aaf213d98",
  "data": {
    "result": [
        ......
        {
                "level": 0,
                "name": "peripheral",
                "message": "OK",
                "hardware_id": "peripheral",
                "values": [
                    {
                        "key": "bmsconn",
                        "value": "ON"
                    },
                    {
                        "key": "bat_chg",
                        "value": "OFF"
                    },
                    {
                        "key": "bat_off",
                        "value": "OFF"
                    },
                    {
                        "key": "bat_prt",
                        "value": "0"
                    },
                    {
                        "key": "bat_vol",
                        "value": "48830"
                    },
                    {
                        "key": "bat_cur",
                        "value": "2870"
                    },
                    {
                        "key": "battery",
```

```
                    "value": "29"
                },
                {
                    "key": "bat_temp0",
                    "value": "430"
                },
                {
                    "key": "bat_temp2",
                    "value": "430"
                },
                {
                    "key": "bat_temp4",
                    "value": "400"
                },
                {
                    "key": "battery_capacity",
                    "value": "9000mAh"
                }
            ]
        },
    ]
  }
}
```


| Field | Meaning |
| --- | --- |
| bmsconn | Battery connection status: [OFF: Not connected, ON: Connected] |
| bat_chg | Battery charger status: [OFF: Not connected, ON: Connected] |
| bat_off | Battery pre-shutdown status: [OFF: Power off after 1s, ON: Normal] |
| bat_prt | Battery fault code: [0: Normal, non-zero: Abnormal] |
| bat_vol | Battery real-time voltage, unit: mV |
| bat_cur | Battery real-time current, unit: mA |
| battery | Battery charge percentage 0~100 |
| bat_temp0 | Battery temperature 0~100, unit: x10°C |
| bat_temp2 | Battery temperature 0~100, unit: x10°C |
| bat_temp4 | Battery temperature 0~100, unit: x10°C |

### 1.2.14.2 Joystick Information Data (Only included in Lite models)

代码块

```
{
  "accid": "HU_D04_01_001",
  "title": "notify_robot_info",
  "timestamp": 1672373633989,
  "guid": "746d937cd8094f6a98c9577aaf213d98",
  "data": {
    "result": [
        ......
        {
                "level": 0,
                "name": "peripheral",
                "message": "OK",
                "hardware_id": "peripheral",
                "values": [
                    {
                        "key": "joystickconn",
                        "value": "ON"
                    },
                    {
                        "key": "joysticksignal",
                        "value": "1"
                    },
                    {
                        "key": "joystickbattery",
                        "value": "2"
                    }
                ]
            },
    ]
  }
}
```

| Field | Meaning |
| --- | --- |
| joystickconn | Joystick connection status: [OFF: Not connected, ON: Connected] |
| joysticksignal | Joystick signal strength: [0-4, higher is stronger] |
| joystickbattery | Joystick battery information: [0:0%~20%<br>1:21%~40%<br>2:41%~60%<br>3:61%~80% |

| | |
|---|---|
| | 4:81%~100%] |

### 1.2.14.3 System Information

代码块

```
{
  "accid": "HU_D04_01_001",
  "title": "notify_robot_info",
  "timestamp": 1672373633989,
  "guid": "746d937cd8094f6a98c9577aaf213d98",
  "data": {
      "result": [
          {
              "level": 0,
              "name": "system_info",
              "message": "system info",
              "hardware_id": "system_info",
              "values": [
                #### Fields common to both Oli and Luna
                {
                  "key": "ability_running",
                  "value": "ZeroTorque"
                },
                {
                  "key": "ecm_version",
                  "value": "1.1.2"
                },
                {
                  "key": "mode",
                  "value": "Remote"
                },
                {
                  "key": "motor_version",
                  "value": "1: 0.0.9; 2: 0.0.9; 3: 0.0.9; 4: 0.0.9; 5: 0.0.9; 6: 0.0.9; 7: 0.0.9; 8: 0.0.9; 9: 0.0.9; 10: 0.0.9; 11: 0.0.9; 12: 0.0.9; 13: 0.0.9; 14: 0.0.9; 15: 0.0.9; 16: 0.0.9; "
                },
                {
                  "key": "pms_version",
                  "value": "2.1.8"
                },
                {
                  "key": "robot_status",
                  "value": "ZeroTorque"
                },
                {
```

```
                "key": "version",
                "value": "robot-hu-d-2.1.0.20251225062343"
              },
              {
                "key": "sn",
                "value": "HU_D04_01_131"
              },
              ########## Luna-specific fields
              { "key": "sdk_lite_led_enable", "value": "1" },
              { "key": "sdk_lite_led_has_params", "value": "1" },
              { "key": "sdk_lite_led_mode", "value": "0" },
              { "key": "sdk_lite_led_state", "value": "1" },
              { "key": "sdk_lite_led_color", "value": "4" },
              { "key": "sdk_lite_led_brightness", "value": "3" },
              { "key": "sdk_lite_leds", "value": "" },
              {"key": "walk_gait", "value":"walk"}
            ]
        }
    ]
  }
}
```

## Common to both Oli and Luna

| Field | Meaning |
|---|---|
| version | Main controller version |
| ecm_version | Master station version |
| pms_version | Power distribution board version |
| motor_version | Motor version |
| sn | Robot serial number |
| robot_status | Robot current status |
| ability_running | Robot currently running controller |

## Luna-specific fields

| Field | Meaning |
|---|---|
| sdk_lite_led_enable() | Luna LED effect control switch: `0` =off, `1` =on |

| | |
|---|---|
| sdk_lite_led_has_params | Whether Luna LED effect parameters already exist: `0` =no, `1` =yes |
| sdk_lite_led_mode | Luna LED effect control mode: `0` =overall LED effect control, `1` =individual LED RGB control |
| sdk_lite_led_state | Overall LED effect status, valid when `mode=0` |
| sdk_lite_led_color | Overall LED effect color, valid when `mode=0` |
| sdk_lite_led_brightness | Overall LED effect brightness, valid when `mode=0` , range `0-5` |
| sdk_lite_leds | Individual LED RGB flat array string, valid when `mode=1` , e.g. `255,0,0,0,255,0` |

代码块

```
    If mode = 1 for individual RGB control:
    { "key": "sdk_lite_led_mode", "value": "1" },
    { "key": "sdk_lite_led_state", "value": "" },
    { "key": "sdk_lite_led_color", "value": "" },
    { "key": "sdk_lite_led_brightness", "value": "" },
    { "key": "sdk_lite_leds", "value": "255,0,0,0,255,0,0,0,255" }
    When disabled:
    { "key": "sdk_lite_led_enable", "value": "0" },
    { "key": "sdk_lite_led_has_params", "value": "0" },
    { "key": "sdk_lite_led_mode", "value": "" },
    { "key": "sdk_lite_led_state", "value": "" },
    { "key": "sdk_lite_led_color", "value": "" },
    { "key": "sdk_lite_led_brightness", "value": "" },
    { "key": "sdk_lite_leds", "value": "" }
```

### 1.2.14.4 Motor Status Information

代码块

```
{
  "accid": "HU_D04_01_001",
  "title": "notify_robot_info",
  "timestamp": 1672373633989,
  "guid": "746d937cd8094f6a98c9577aaf213d98",
  "data": {
      "result": [
          {
                "level": 1,
                "name": "ethercatCommunicationExp",
                "message": "WARN",
```

```
            "hardware_id": "ethercat",
            "values": [
                {
                    "key": "ethercatCommunicationExp",
                    "value": "motor 17 MOTOR_LOST triggered HALF_STAND"
                },
                {
                    "key": "ethercatResetNormal",
                    "value": "ok!"
                }
            ]
        }
    ]
  }
}
```

| Field | Meaning |
| --- | --- |
| level | Exception level [0:ok 1:warn 2:error] |
| name | Exception type |
| message | Level string |
| hardware_id | Hardware ID |
| values | All exception sets for this hardware |

### 1.2.15 Remote Controller Data

Robot remote controller data is reported through this protocol:

- `accid` : Robot serial number
- `title` : notify_joy_data
- `timestamp` : The timestamp when the message is sent, in milliseconds
- `guid` : The guid value of the message, uniquely identifying this message
- `data` : Contains the message content, example:

代码块

```
{
    "accid": "HU_D04_01_001",
    "title": "notify_joy_data",
    "timestamp": 1672373633989,
```

```
    "guid": "746d937cd8094f6a98c9577aaf213d98",
    "data": {
      "axes": [],      # Joystick axis data
      "buttons": []    # Button data
    }
  }
```

## 1.2.16 Protocol Interface Call Examples

### 1.2.16.1 Python Example Implementation

- Environment preparation: Taking Ubuntu 20.04 as an example, install the following dependencies

代码块

```
sudo apt install python3-dev python3-pip
sudo pip install websocket-client==1.8.0
```

- Run the script

代码块

```
python humanoid.py
```

- humanoid.py implementation

> 💡
> - ACCID: Replace with the actual software SN
> - ROBOT_IP: Generally, 127.0.0.1 for simulation, 10.192.1.2 for real robot

代码块

```
import json
import uuid
import threading
import time
import websocket
from datetime import datetime

# Replace this ACCID value with your robot's actual serial number (SN)
ACCID = None

# Replace it with the real IP address of the robot.
```

```python
# Usually, for simulation, it is: 127.0.0.1
# for a real machine, it is: 10.192.1.2
ROBOT_IP = "10.192.1.2"

# Atomic flag for graceful exit
should_exit = False

# WebSocket client instance
ws_client = None

# Generate dynamic GUID
def generate_guid():
    return str(uuid.uuid4())

# Send WebSocket request with title and data
def send_request(title, data=None):
    global ACCID
    if data is None:
        data = {}

    # Create message structure with necessary fields
    message = {
        "accid": ACCID,
        "title": title,
        "timestamp": int(time.time() * 1000),  # Current timestamp in milliseconds
        "guid": generate_guid(),
        "data": data
    }

    message_str = json.dumps(message)

    # Send the message through WebSocket if client is connected
    if ws_client:
        ws_client.send(message_str)

# Handle user commands
def handle_commands():
    global should_exit
    while not should_exit:
        command = input("Enter command ('prepare', 'damping', 'zero') or 'exit' to quit:\n")

        if command == "exit":
            should_exit = True  # Set exit flag to stop the loop
            break
        elif command == "prepare":
```

```
            send_request("request_prepare")  # request_prepare
        elif command == "damping":
            send_request("request_damping")  # request_damping
        elif command == "zero":
            send_request("request_zero_torque")  # request_zero_torque

# WebSocket on_open callback
def on_open(ws):
    print("Connected!")
    # Start handling commands in a separate thread
    threading.Thread(target=handle_commands, daemon=True).start()

# WebSocket on_message callback
def on_message(ws, message):
    global ACCID
    root = json.loads(message)
    title = root.get("title", "")
    ACCID = root.get("accid", None)

    if title != "notify_robot_info":
        print(f"Received message: {message}")  # Print the received message

# WebSocket on_close callback
def on_close(ws, close_status_code, close_msg):
    print("Connection closed.")

# Close WebSocket connection
def close_connection(ws):
    ws.close()

def main():
    global ws_client

    # Create WebSocket client instance
    ws_client = websocket.WebSocketApp(
        f"ws://{ROBOT_IP}:5000",  # WebSocket server URI
        on_open=on_open,
        on_message=on_message,
        on_close=on_close
    )

    # Configure socket send and receive buffer sizes
    # Increase send buffer size to 2MB (default is typically much smaller)
    # This helps prevent data loss when sending large messages or high-frequency data
    ws_client.sock_opt = [("socket", "SO_SNDBUF", 2 * 1024 * 1024)]

```

```
    # Increase receive buffer size to 2MB
    # This allows handling larger incoming messages without truncation
    ws_client.sock_opt.append(("socket", "SO_RCVBUF", 2 * 1024 * 1024))

    # Run WebSocket client loop
    print("Press Ctrl+C to exit.")
    ws_client.run_forever()

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

### 1.2.16.2 Linux C++ Example

- Install dependencies: Taking Ubuntu 20.04 as an example, install websocketpp, nlohmann/json, and boost dependencies:

代码块

```
sudo apt-get install libboost-all-dev libwebsocketpp-dev nlohmann-json3-dev
```

- Compile the code

代码块

```
g++ -std=c++11 humanoid humanoid.cpp -o humanoid humanoid -lssl -lcrypto -lboost_system -lpthread
```

- Run the program

代码块

```
./humanoid
```

- humanoid.cpp implementation

Code Block

```
#include <iostream>
#include <atomic>
#include <string>
#include <thread>
#include <chrono>
#include <websocketpp/client.hpp>
```

```cpp
#include <websocketpp/config/asio.hpp>
#include <nlohmann/json.hpp>
#include <boost/uuid/uuid.hpp>
#include <boost/uuid/uuid_generators.hpp>
#include <boost/uuid/uuid_io.hpp>

using json = nlohmann::json;
using websocketpp::client;
using websocketpp::connection_hdl;

// Replace this value with the actual serial number (SN) of the robot.
static std::string ACCID = "";

// Replace it with the real IP address of the robot.
// Usually, for simulation, it is: 127.0.0.1
// for a real machine, it is: 10.192.1.2
const std::string ROBOT_IP = "10.192.1.2";

// WebSocket client instance
static client<websocketpp::config::asio> ws_client;

// Atomic flag for graceful exit
static std::atomic<bool> should_exit(false);

// Connection handle for sending messages
static connection_hdl current_hdl;

// Generate dynamic GUID
static std::string generate_guid() {
  boost::uuids::random_generator gen;
  boost::uuids::uuid u = gen();
  return boost::uuids::to_string(u);
}

// Send WebSocket request with title and data
static void send_request(const std::string& title, const json& data =
json::object()) {
  json message;

  // Adding necessary fields to the message
  message["accid"] = ACCID;
  message["title"] = title;
  message["timestamp"] =
std::chrono::duration_cast<std::chrono::milliseconds>(

std::chrono::system_clock::now().time_since_epoch()).count();
  message["guid"] = generate_guid();
```

```
    message["data"] = data;

    std::string message_str = message.dump();

    // Send the message through WebSocket
    ws_client.send(current_hdl, message_str,
websocketpp::frame::opcode::text);
}

// Handle user commands
void handle_commands() {
    std::cout << "Enter command ('prepare', 'damping', 'zero') or 'exit' to
quit:\n";
    while (!should_exit) {
        std::string command;
        std::cin >> command;

        if (command == "exit") {
            should_exit = true;
            return;
        } else if (command == "prepare") {
            send_request("request_prepare");
        } else if (command == "damping") {
            send_request("request_damping");
        } else if (command == "zero") {
            send_request("request_zero_torque");
        }

        sleep(1);

        std::cout << "\nEnter command ('prepare', 'damping', 'zero') or
'exit' to quit:\n";
    }
}

// WebSocket open callback
static void on_open(connection_hdl hdl) {
    std::cout << "Connected!" << std::endl;

    // Save connection handle for sending messages later
    current_hdl = hdl;

    // Start handling commands in a separate thread
    std::thread(handle_commands).detach();
}

// WebSocket TCP initialization handler
```

```
95  static void on_tcp_init(connection_hdl hdl)
96  {
97  auto con = ws_client.get_con_from_hdl(hdl);
98
99  // Obtain the underlying TCP socket
100 auto& socket = con->get_socket().lowest_layer();
101
102 // Configure socket options
103 try {
104   boost::system::error_code ec;
105
106   // Set send buffer size (e.g., 2MB)
107   const size_t sendBufferSize = 2 * 1024 * 1024;
108
    socket.set_option(websocketpp::lib::asio::socket_base::send_buffer_size(send
    BufferSize), ec);
109
110   if (ec)
111   {
112     printf("Failed to set send buffer size: %s", ec.message().c_str());
113   }
114
115   // Set receive buffer size (e.g., 2MB)
116   const size_t recvBufferSize = 2 * 1024 * 1024;
117
    socket.set_option(websocketpp::lib::asio::socket_base::receive_buffer_size(r
    ecvBufferSize), ec);
118
119   if (ec)
120   {
121     printf("Failed to set receive buffer size: %s", ec.message().c_str());
122   }
123
124   // Disable Nagle's algorithm to reduce latency
125   socket.set_option(websocketpp::lib::asio::ip::tcp::no_delay(true), ec);
126
127   if (ec)
128   {
129     printf("Failed to disable Nagle's algorithm: %s", ec.message().c_str());
130   }
131 } catch (const std::exception& e) {
132   printf("Socket configuration exception: %s", e.what());
133 }
134 }
135
136 // WebSocket message callback
```

```
137  static void on_message(connection_hdl hdl,
     client<websocketpp::config::asio>::message_ptr msg) {
138  // Parse JSON data from message payload
139  json data = json::parse(msg->get_payload());
140
141  // Extract 'accid' field if present
142  if (data.contains("accid") && data["accid"].is_string() && ACCID.empty()) {
143      ACCID = data["accid"].get<std::string>();
144  }
145
146  if (msg->get_payload().find("notify_robot_info") == std::string::npos) {
147      std::cout << "Received message: " << msg->get_payload() << std::endl;
148  }
149  }
150
151  // WebSocket close callback
152  static void on_close(connection_hdl hdl) {
153    std::cout << "Connection closed." << std::endl;
154  }
155
156  // Close WebSocket connection
157  static void close_connection(connection_hdl hdl) {
158    ws_client.close(hdl, websocketpp::close::status::normal, "Normal
     closure");  // Close connection normally
159  }
160
161  int main() {
162    ws_client.init_asio();  // Initialize ASIO for WebSocket client
163
164    ws_client.set_access_channels(websocketpp::log::alevel::none);
165
166    // Set WebSocket event handlers
167    ws_client.set_open_handler(&on_open);  // Set open handler
168    ws_client.set_message_handler(&on_message);  // Set message handler
169    ws_client.set_close_handler(&on_close);  // Set close handler
170    ws_client.set_tcp_init_handler(&on_tcp_init); // Set tcp init handler
171
172    std::string server_uri = "ws://" + ROBOT_IP + ":5000";  // WebSocket
     server URI
173
174    websocketpp::lib::error_code ec;
175    client<websocketpp::config::asio>::connection_ptr con =
     ws_client.get_connection(server_uri, ec);  // Get connection pointer
176
177    if (ec) {
178        std::cout << "Error: " << ec.message() << std::endl;
179        return 1;  // Exit if connection error occurs
```

```
    }

    connection_hdl hdl = con->get_handle();  // Get connection handle
    ws_client.connect(con);  // Connect to server
    std::cout << "Press Ctrl+C to exit." << std::endl;

    // Run the WebSocket client loop
    ws_client.run();

    return 0;
}
```

| **Function Name** | **subscribeImuData** |
| --- | --- |
| Function Prototype | void subscribeImuData(std::function<void(const ImuDataConstPtr&)> cb); |
| Description | Subscribe to the robot's **IMU data** and call the specified callback function when new IMU data is received. |
| Parameters | cb: Callback function for processing new IMU data. |
| Return Value | None |
| Remarks | The ImuData data structure prototype is as follows: |

代码块

```
/**
 * @struct ImuData
 *
 * @brief Struct representing robot IMU data based on sensor feedback.
 *
 * This struct encapsulates IMU data, including accelerometer, gyroscope, and quaternion.
 */
struct ImuData {
  uint64_t stamp; // Timestamp in nanoseconds, typically indicating when this data was recorded or generated.
  float acc[3];   // Stores IMU accelerometer data to track linear acceleration along three axes (X, Y, Z).
```

```
  float gyro[3];  // Stores IMU gyroscope data to track angular velocity or rotation rate along three axes (X, Y, Z).
  float quat[4];  // Stores IMU quaternion data representing orientation in 3D space (w, x, y, z).
};

// Smart pointer type aliases
typedef std::shared_ptr<ImuData> ImuDataPtr;
typedef std::shared_ptr<ImuData const> ImuDataConstPtr;
```

Code Example

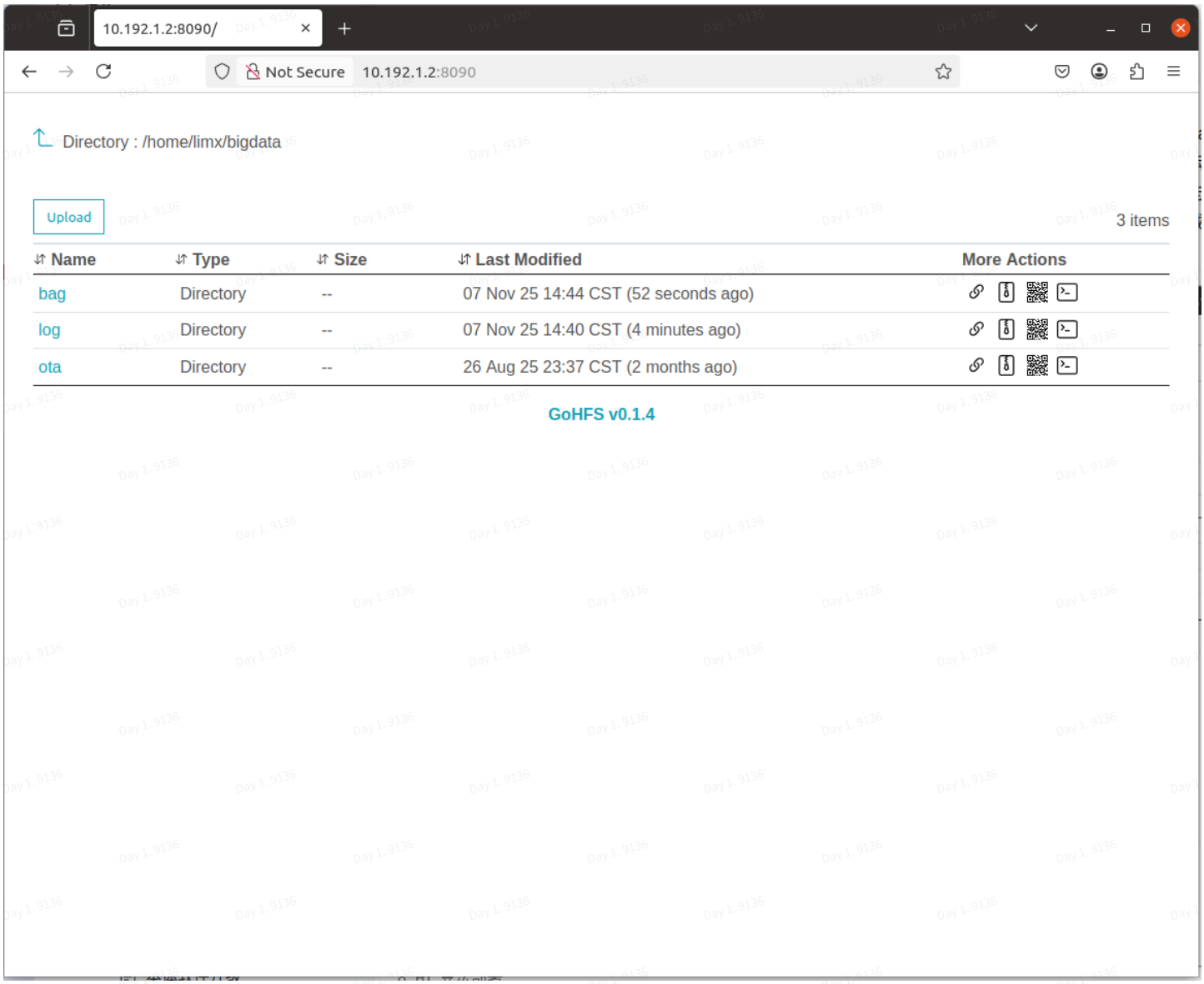
Code Block

```
#include <thread>

// Include limxsdk::Humanoid header to import the Humanoid class
#include "limxsdk/humanoid.h"

// Use limxsdk namespace to simplify references to the Humanoid class
using namespace limxsdk;

int main(int argc, char *argv[]){
  // Get the singleton instance of the Humanoid class
  Humanoid* robot = Humanoid::getInstance();

  // Default robot IP address
  std::string robot_ip = "127.0.0.1";
  if (argc > 1)
  {
    // If command-line arguments are provided, use them as the robot IP address
    robot_ip = argv[1];
  }

  // Initialize the communication runtime environment for the motion control algorithm
  if (!robot->init(robot_ip))
  {
    // Exit if initialization fails
    exit(1);
  }

  // Subscribe to robot status updates with a callback function
  robot->subscribeImuData([&](const ImuDataConstPtr& msg) {
    // Process received ImuData data here
```

```
31      // Note: The callback function is called when ImuData is
    received
32    });
33
34    // Infinite loop to keep the program running
35    while (true)
36    {
37      // Sleep for 1000 milliseconds
38
    std::this_thread::sleep_for(std::chrono::milliseconds(1000));
39    }
40    return 0;
41  }
```

## 1.3 Logs and Data Packets

The robot system automatically records: robot IMU data (ImuData), robot state data (/joint/state), and robot control data (/joint/cmd), and other important data. These data are crucial for robot motion control analysis. In addition, the robot also records runtime log data for troubleshooting and performance optimization when needed. When the computer is connected to the robot's Wi-Fi hotspot, you can access it by entering `http://10.192.1.2:8090` in the browser and download these data. This process provides convenience for robot monitoring, maintenance, and debugging.



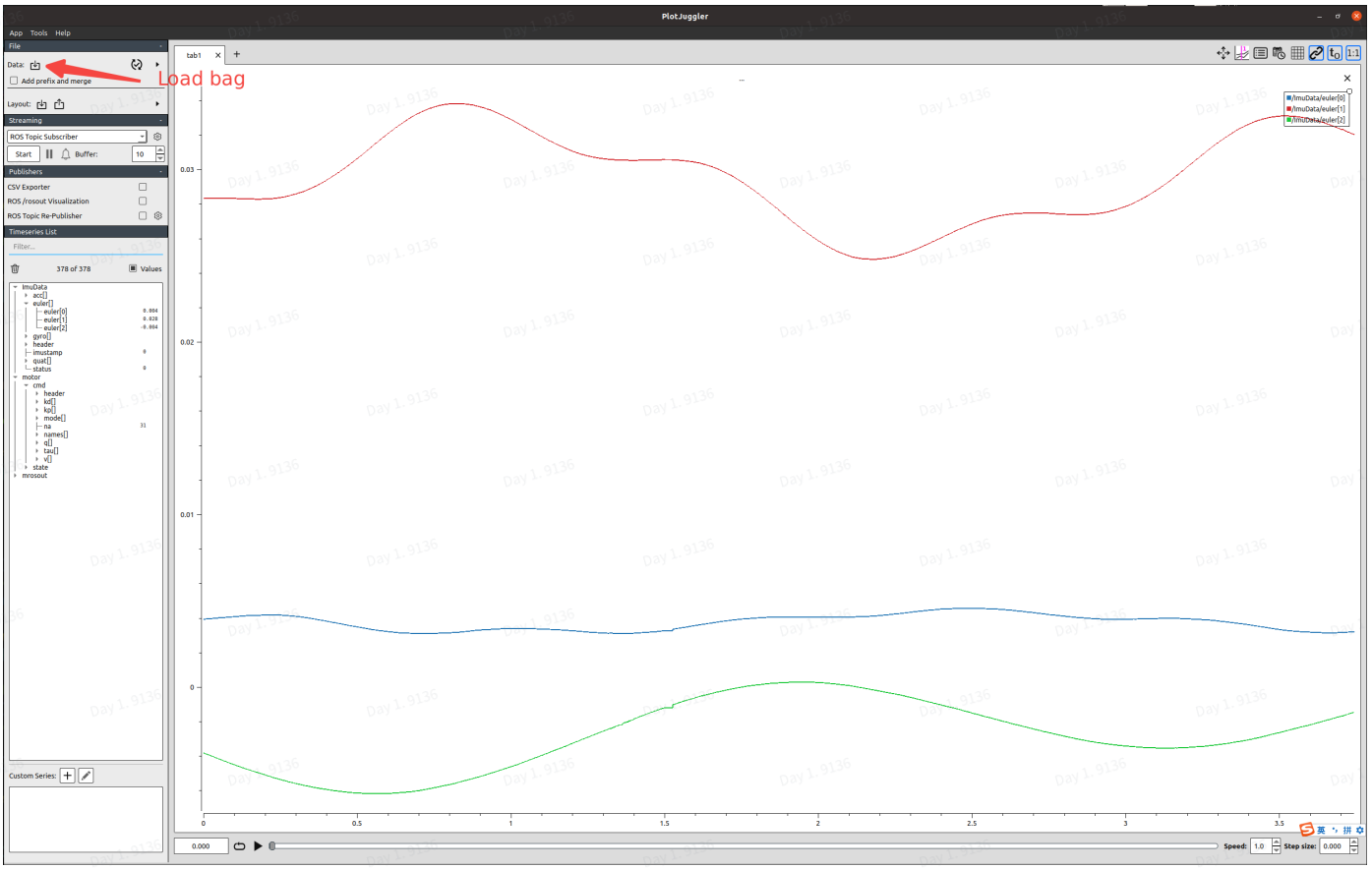

### 1.3.1 Data Packet Visualization Analysis Method

- **Data Packet Download:** After downloading the `.bag` file, you can use the PlotJuggler visualization tool to load and analyze these packet data. It is particularly important to note that if you downloaded a `.bag.active` file, you need to use the following Shell command to reindex the `.bag.active` file and generate a new `.bag` file for PlotJuggler to load.

代码块

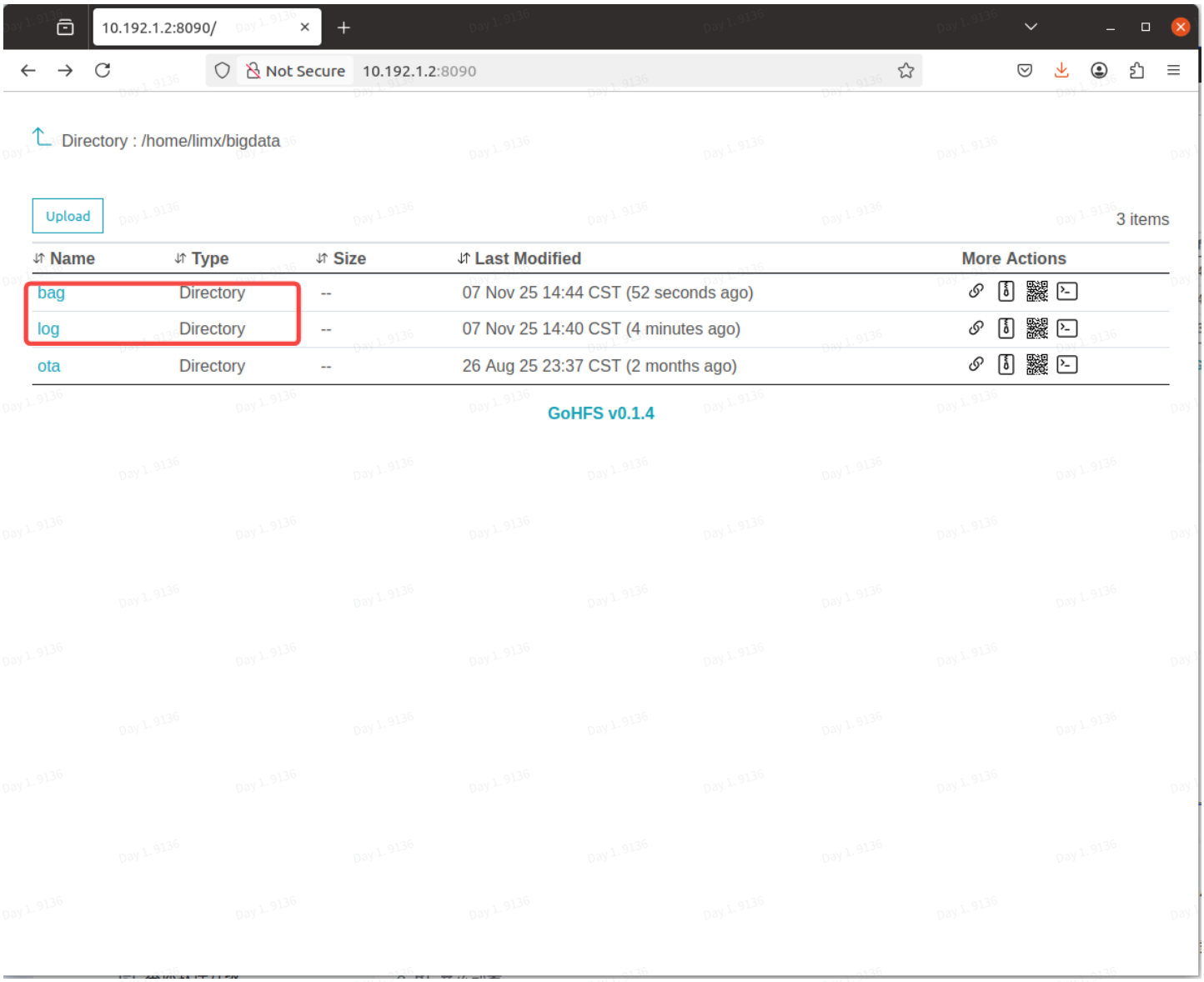
```
rosbag reindex your_file.bag.active
mv your_file.bag.active your_file.bag
```

- **Visualization View:** Start the PlotJuggler visualization tool through the Shell command `rosrun plotjuggler plotjuggler -n`. Load the data packet and analyze the data as shown in the figure below.




## 1.3.2 Log and Diagnostic Tracking Point Data

The following figures show the log and tracking point structured data respectively. They can be used for troubleshooting and performance optimization when needed.




## 1.4 Robot Software Upgrade

We enter the robot management page through the browser and select the locally pre-downloaded robot software version for upgrade. The specific steps are as follows:

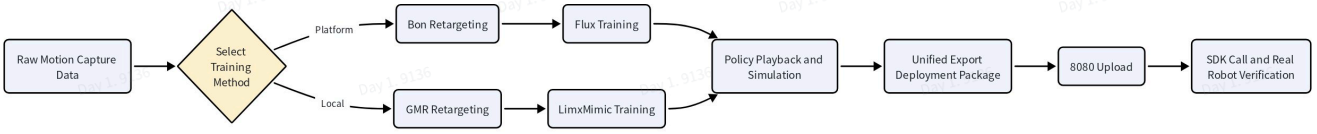
- Please select and connect to your robot's Wi-Fi hotspot, password: `12345678`

1. Access the management page:
   - Enter in the browser address bar: http://10.192.1.2:8080 to enter the robot management page.
2. Select and upgrade software:
   - Select "Version Management -> Browse -> Upgrade" in sequence.
   - After the upgrade is completed, the robot's main control computer will automatically restart.

# 2. Motion Training Method Reference

## 2.1 Training Method Selection

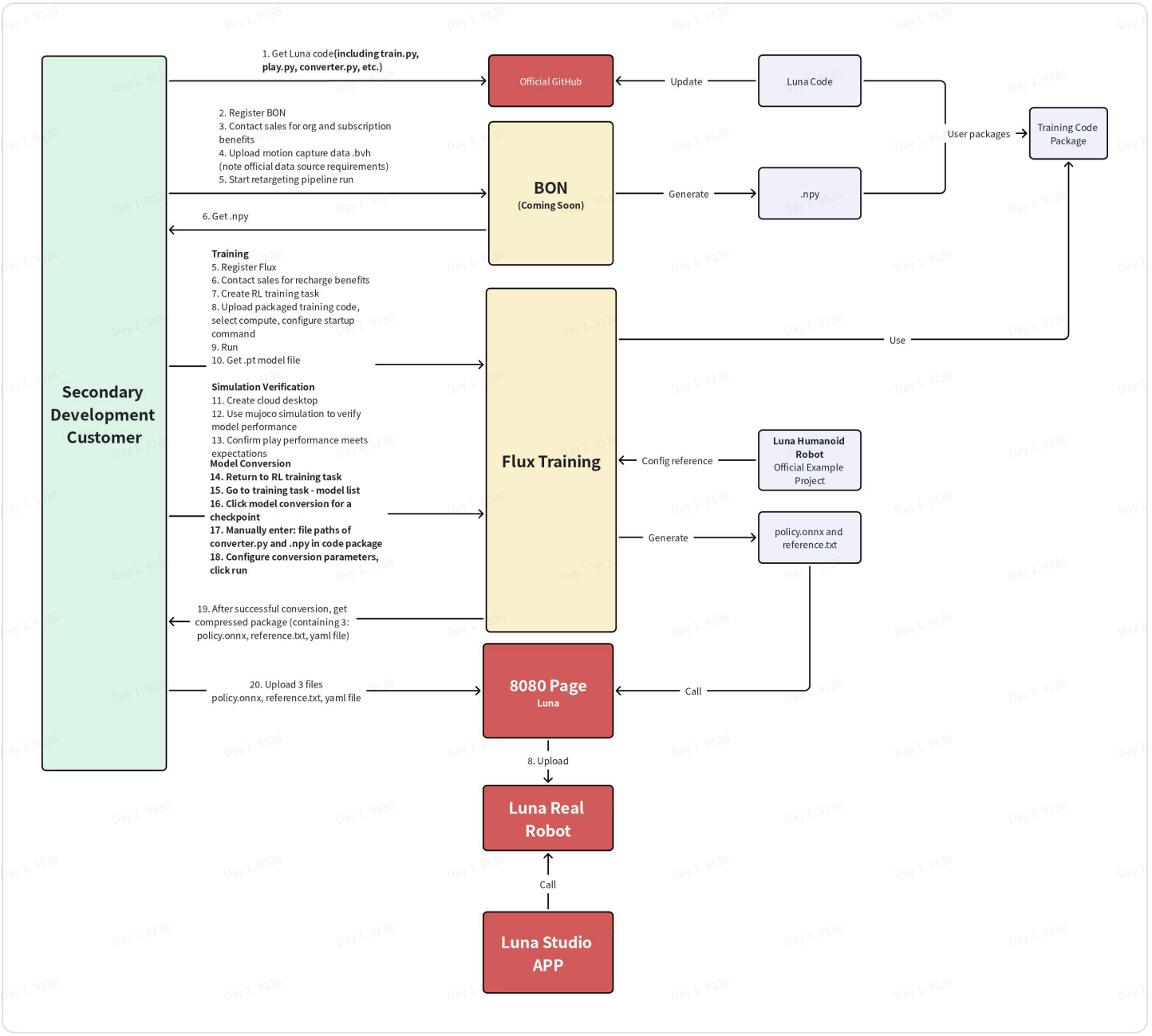
| Method | Applicable Scenarios | Main Process | Prerequisites |
| --- | --- | --- | --- |
| **Bon & Flux** | Platform accessible, wishing to reduce local environment setup work | Bon retargeting → Flux training → cloud desktop/local simulation → export | Bon, Flux accounts and valid computing power quota |
| **Local Offline Training** | Data cannot leave the domain, platform unreachable, or need for high-frequency parameter tuning iterations | GMR local retargeting → LimxMimic local training → local simulation → export | Ubuntu, NVIDIA GPU, official repository access permissions |




> ❗ The final deployment specifications for both routes are the same. Training, playback, and export must use the motion data and checkpoint corresponding to the same task. Do not mix products from different motions, different code versions, or different task configurations.

## 2.2 Platform Training Based on BON & Flux Training

### 2.2.1 User Business Full Process




Note: When Bon is not online, retargeting can be completed through local scripts

## 2.2.2 BON & Flux Training Product User Manual

| Product | Description | Status | Timeline | Official Website | Product User Manual |
|---|---|---|---|---|---|
| LimX BON | BON is a full-process motion data pipeline management platform built for embodied intelligent robots, covering the complete data chain from motion capture upload, Retargeting, to quality assessment and training data export. | In Development | Expected to launch in September | bon.limxdynamics.com | LimX BON \| Product User Manual |

| | | | | | |
|---|---|---|---|---|---|
| LimX Flux Training | Flux Training is a cloud-based algorithm training platform specifically built for embodied intelligence, providing ready-to-use GPU computing power with pre-installed mainstream simulation environments (such as Isaac Gym, Isaac Sim), and engineering adaptations, allowing **developers** to say goodbye to tedious configurations and focus on algorithm training and robot innovation research. | Launched | Full capabilities available, model conversion launched on 9.3 | https://flux.limxdynamics.com | Flux Training \| Product User Manual |

## 2.3 Local Offline Training

### 2.3.1 Computer and Software Requirements

- Recommended Ubuntu 22.04, NVIDIA graphics card and available drivers.
- Minimum recommendation: 16-core CPU, 24 GB VRAM, 32 GB RAM; reserve at least 25 GB of disk space.
- Install Git, Git LFS, Miniconda or Anaconda.
- GitHub account needs access to GMR, Luna robot description repository, and official LimxMimic training repository.

> 💡 Testing confirms that the business training methods for Luna L03 and L04 are consistent; the current public export tool fixedly uses the 141-dimensional observation and 27-dimensional action of HU_L03_01 Parallel Deploy Gravity. The task name, robot configuration, and export tool must come from the same version of the official delivery package. Do not mix L03/L04 configurations on your own.

### 2.3.2 Download Code and Prepare Directory

代码块

```
cd ~
git clone --branch limx --single-branch \
  https://github.com/limx-retarget/GMR.git GMR
git clone https://github.com/limx-luna/luna-beyondmimic.git whole_body_tracking
mkdir -p ~/data
```

Subsequently, use `~/GMR` uniformly for retargeting, and use `~/whole_body_tracking` for motion preparation, training, playback, and export. Place the original motion files in `~/data`, and try to use English for file names and paths without spaces.

### 2.3.3 Install GMR Environment

代码块

```
cd ~/GMR
git lfs install
git lfs pull
git config submodule.assets/luna-description.url \
  https://github.com/limx-luna/luna-description.git
git submodule update --init assets/luna-description

conda create -n gmr python=3.10 -y
conda activate gmr
python -m pip install -e .
conda install -c conda-forge libstdcxx-ng -y
```

> ❗ If the submodule shows `Permission denied` or `Repository not found`, it means the account lacks Luna description repository permissions. You should apply for permissions first. Do not skip the submodule or copy asset files of unknown versions.

### 2.3.4 Determine Original Data Type

| Input | Processing Entry | Key Confirmation Items |
| --- | --- | --- |
| Luna Retargeting NPY | Direct preview and training | Must be generated by the official GMR/Bon process, not any NumPy file |
| SMPL-X NPZ | `smplx_to_robot.py` | pose, root, trans and frame rate fields are complete |
| Ordinary BVH | `bvh_to_robot.py` | Confirm lafan1, nokov, fzmotion, soma or noitom format and actual frame rate |
| Xsens 3ds Max BVH | `xsens_bvh_to_robot.py` | Confirm displacement unit; centimeters use 0.01, millimeters use 0.001 |

| | | |
|---|---|---|
| Video, FBX, CSV, etc. | Convert first | The current official process cannot read directly |

### 2.3.5 Retarget to Luna Motion

The following commands only execute the one corresponding to the data type:

代码块

```
conda activate gmr
cd ~/GMR
mkdir -p output

# Ordinary BVH example: modify format and motion_fps according to actual data
python scripts/bvh_to_robot.py \
  --bvh_file ~/data/walk.bvh \
  --format nokov \
  --motion_fps 50 \
  --robot limx_luna \
  --save_path output/my_motion.npy

# Xsens 3ds Max BVH example
python scripts/xsens_bvh_to_robot.py \
  --bvh_file ~/data/walk_xsens.bvh \
  --robot limx_luna \
  --bvh_format 3DSM \
  --scale 0.01 \
  --reset_to_zero \
  --save_path output/my_motion.npy
```

When the BVH type is unknown, you should first confirm the acquisition device and export template with the data provider. Incorrect format, frame rate, or unit often manifests as the robot moving sideways, body leaving the ground, or abnormal limb directions.

### 2.3.6 Preview Retargeted Motion

代码块

```
conda activate gmr
cd ~/GMR
python scripts/vis_robot_motion.py \
  --robot limx_luna \
  --robot_motion_path output/my_motion.npy
```

Confirm that the overall direction, limb posture, foot contact, and motion tail have no obvious abnormalities. In a pure SSH environment, you can temporarily skip the preview, but you must complete simulation verification after training.

### 2.3.7 Install LimxMimic Training Environment

代码块

```
conda create -n limxmimic python=3.10.15 -y
conda env config vars set PYTHONNOUSERSITE=1 -n limxmimic
conda env config vars set OMNI_KIT_ACCEPT_EULA=Y -n limxmimic
conda activate limxmimic

python -m pip install --upgrade pip
python -m pip install "setuptools==75.8.0" wheel
python -m pip install torch==2.5.1 torchvision==0.20.1 \
  --index-url https://download.pytorch.org/whl/cu118
python -m pip install --no-cache-dir "isaacsim[all,extscache]==4.5.0" \
  --extra-index-url https://pypi.nvidia.com

cd ~
git clone --branch v2.1.0 --depth 1 \
  https://github.com/isaac-sim/IsaacLab.git IsaacLab-2.1.0
cd ~/IsaacLab-2.1.0
echo "setuptools<81" > ~/isaaclab-build-constraints.txt
export PIP_CONSTRAINT=~/isaaclab-build-constraints.txt
export TERM=xterm
./isaaclab.sh --install

cd ~/whole_body_tracking
git submodule update --init --recursive
python -m pip install -e source/whole_body_tracking
```

GMR and LimxMimic use two independent Conda environments. Retargeting commands run in `gmr`, and training and export commands run in `limxmimic`.

### 2.3.8 Pre-training Check

代码块

```
conda activate limxmimic
cd ~/whole_body_tracking
python scripts/prepare_motion.py \
  --input ~/GMR/output/my_motion.npy \
  --output motions/my_motion_parallel_tail10.npz \
  --robot hu_l03_parallel \
```

```
  --linkage_report motions/my_motion_linkage_report.json \
  --force_tail
```

This step is used to explicitly check parallel linkage solving, tail frames, and kinematics results. The generated NPZ is a diagnostic intermediate file; the current recommended training, playback, and combined export still uniformly pass in the same trusted NPY, and the tool internally executes a consistent motion preparation process.

### 2.3.9 Start Training

代码块

```
conda activate limxmimic
cd ~/whole_body_tracking
python scripts/rsl_rl/train.py \
  --task=Tracking-Flat-HU-L03-Parallel-Deploy-Gravity-v0 \
  --motion_file ~/GMR/output/my_motion.npy \
  --motion_force_tail \
  --num_envs 4096 \
  --headless \
  --max_iterations 15000 \
  --logger tensorboard
```

When the training directory appears in the terminal and iteration information is continuously updated, it means training has started normally. When VRAM is insufficient, reduce `--num_envs` to 2048 or 1024 in sequence. Do not modify the task name and observation/action dimensions.

### 2.3.10 Playback and Export

代码块

```
# Playback the checkpoint from the same training run
python scripts/rsl_rl/play.py \
  --task=Tracking-Flat-HU-L03-Parallel-Deploy-Gravity-v0 \
  --checkpoint /path/to/model_14999.pt \
  --motion_file ~/GMR/output/my_motion.npy \
  --motion_force_tail \
  --num_envs 1 \
  env.commands.motion.debug_vis=false \
  env.scene.contact_forces.debug_vis=false

# Export the complete deployment package
python tools/offline_onnx_exporter/export_policy_and_reference.py \
```

```
    --checkpoint /path/to/model_14999.pt \
    --motion ~/GMR/output/my_motion.npy \
    --output_path exported/my_motion
```

## 2.4 Training Engineering, Adjustable Parameters and Interface Red Lines

### 2.4.1 Repository Directory Structure

The repository is divided into two major parts: `source/` is the training code installed as a Python package, and `scripts/` is the directly executable command-line tools. Daily configuration changes are in `source/`, and daily process runs are in `scripts/`.

代码块

```
whole_body_tracking/
├── scripts/                          Command-line tools, execute directly
│   ├── rsl_rl/
│   │   ├── train.py                  Start training
│   │   ├── play.py                   Playback policy, also exports ONNX
│   │   └── cli_args.py               Training/playback command-line parameters
│   ├── solve_parallel_linkage.py     Solve closed-chain degrees of freedom
│   ├── npy_to_npz.py                 Retargeting npy to training npz
│   ├── append_motion_tail.py         Add tail frames that maintain the terminal posture
│   ├── verify_motion_npz.py          Verify npz with forward kinematics (hard gate)
│   ├── dump_articulation_order.py    Print the canonical joint/rigid body order of the model
│   └── replay_npz.py                 Directly playback reference motions in Isaac Sim
│
├── source/whole_body_tracking/whole_body_tracking/
│   ├── tasks/tracking/
│   │   ├── tracking_env_cfg.py       Total configuration for rewards, terminations, domain randomization, observations
│   │   ├── mdp/
│   │   │   ├── commands.py           Reference motion loading, error calculation, adaptive sampling
│   │   │   ├── rewards.py            Reward function implementation
│   │   │   ├── terminations.py       Termination condition implementation
│   │   │   ├── events.py             Domain randomization implementation
│   │   │   └── observations.py       Observation item implementation
│   │   └── config/hu_l03/
```

```
24  │   │       ├── parallel_env_cfg.py  Environment configuration for closed-
    chain variant
25  │   │       └── agents/
26  │   │           └── rsl_rl_ppo_cfg.py  PPO hyperparameters and network
    structure
27  │   ├── robots/
28  │   │   └── hu_l03_parallel.py       Closed-chain model actuator gains,
    armature, action scaling
29  │   ├── assets/HU_L03_description/   Robot USD / URDF / MJCF
30  │   └── utils/exporter.py            ONNX export and metadata
31  │
32  ├── motions/                         Converted npz reference motions
33  ├── logs/rsl_rl/                     Training products: checkpoint,
    TensorBoard, exported ONNX
34  ├── export_specs/                    Offline ONNX export specification files
35  └── tools/offline_onnx_exporter/     Independent export tool not dependent on
    Isaac Sim
```

### 2.4.2 Configuration File Quick Reference

| Modification Target | Corresponding File |
| --- | --- |
| Rewards, termination thresholds, domain randomization | `tasks/tracking/tracking_env_cfg.py` |
| Curriculum sampling parameters | `MotionCommandCfg` in `tasks/tracking/mdp/commands.py` |
| Add a new reward function | `tasks/tracking/mdp/rewards.py` |
| PPO hyperparameters, network width | `config/hu_l03/agents/rsl_rl_ppo_cfg.py` |
| Actuator stiffness and damping | `robots/hu_l03_parallel.py` |
| Observation items | `config/hu_l03/parallel_env_cfg.py`, **but do not modify** |

### 2.4.3 Interface Red Line: Observation and Action Items Cannot Be Modified

❌ **It is forbidden to add, delete, or adjust observation items, observation order, action dimensions, and action order.** These contents constitute the fixed interface

contract between the policy and the robot's lower-level controller. After modification, the policy cannot be correctly deployed.

| Item | Reason for Not Being Modifiable |
| --- | --- |
| **Observation** | Each item and its order are written into ONNX metadata. The lower-level controller concatenates the input vector according to a fixed layout. Adding or deleting items or changing the order will cause input misalignment. |
| **Action** | The 27-dimensional actions correspond to specific motor slots respectively. Changing the width or order will cause control commands to be sent to the wrong joints. |

The current task `Tracking-Flat-HU-L03-Parallel-Deploy-Gravity-v0` uses **141-dimensional observation, 27-dimensional action**:

代码块

```
command(54) + base_ang_vel(3) + projected_gravity(3)
            + joint_pos(27) + joint_vel(27) + actions(27) = 141
```

**Deploy** means that global position, state-estimated linear velocity, and sensorless link degrees of freedom that the lower-level controller cannot directly obtain have been removed; **Gravity** means that the gravity direction directly provided by the IMU is retained.

### 2.4.4 Reward Parameters

Rewards are a relatively safe adjustment entry, defined in `RewardsCfg` of `tracking_env_cfg.py`.

| Reward Item | Weight | std | Main Impact After Increasing |
| --- | --- | --- | --- |
| motion_global_anchor_pos | 0.5 | 0.3 | Torso world position fits more tightly, but at the cost of local posture degrees of freedom |
| motion_global_anchor_ori | 0.5 | 0.4 | More accurate torso orientation |
| motion_body_pos | 1.0 | 0.3 | More accurate limb positions, the main constraint for motion similarity |
| motion_body_ori | 1.0 | 0.4 | More accurate limb orientations |
| motion_body_lin_vel | 1.0 | 1.0 | |

| | | | |
|---|---|---|---|
| | | | Linear velocity and motion rhythm closer to the reference |
| motion_body_ang_vel | 1.0 | 3.14 | Rotation rhythm closer to the reference |
| action_rate_l2 | −0.1 | — | Smoother actions, less jitter, but slower response |
| joint_limit | −10.0 | — | Stronger avoidance of joint limits, may conflict with reference motions close to the limits |
| undesired_contacts | −0.1 | — | Reduce contact with parts other than feet and hands |

💡 **std is usually more worth prioritizing adjustment than weight.** The tracking reward adopts the `exp(-error² / std²)` form; the smaller the std, the stricter the constraint, and the larger the std, the higher the tolerance range. The six `motion_*` items define the task itself and are not recommended for deletion.

### 2.4.5 Termination Conditions

Termination conditions are defined in `TerminationsCfg`, which directly determines whether the policy has the opportunity to learn the complete motion.

| Termination Item | Default Threshold | Meaning |
|---|---|---|
| time_out | 10 s (500 steps) | Normal end, not a failure |
| anchor_pos | 0.25 m | Maximum allowable deviation of torso anchor height from the reference |
| anchor_ori | 0.8 | Deviation of torso tilt from the reference |
| ee_body_pos | 0.25 m | Maximum allowable deviation of any end height of both ankles and both wrists from the reference |

You can use "threshold ÷ peak vertical velocity of the end" to estimate the fault tolerance window. For example, if the peak velocity is 2 m/s and the threshold is 0.25 m, the policy is only allowed to lag by about 0.125 seconds. When the completion rate of large leg lifts, jumps, or standing-up motions is persistently low, and the failure reason is concentrated in `ee_body_pos`, you can evaluate relaxing the threshold to 0.4–0.5 m.

### 2.4.6 Domain Randomization

Increasing randomization can improve real-robot robustness, but will reduce training speed and simulation tracking accuracy.

| Parameter | Default Range | Effect |
|---|---|---|
| physics_material static friction | 0.3–1.6 | Cover different ground friction conditions |
| physics_material dynamic friction | 0.3–1.2 | Affect sliding characteristics after contact |
| physics_material elasticity | 0.0–0.5 | Control ground contact rebound |
| add_joint_default_pos | ±0.01 rad | Simulate joint zero calibration errors |
| base_com | x ±0.025 m, y/z ±0.05 m | Simulate torso center of mass deviation, has a large impact on balance |
| push_robot interval | 1–3 s | Control external disturbance frequency |
| push_robot velocity | xy ±0.5 m/s, yaw ±0.78 rad/s | Increasing can improve anti-disturbance ability, but will interfere with fine motions |

### 2.4.7 Curriculum Sampling

Curriculum sampling is defined in `MotionCommandCfg`, used to determine from which time point of the motion each reset starts.

| Parameter | Default Value | Impact |
|---|---|---|
| adaptive_kernel_size | 1 | When 1, failures are only recorded at the current time point; when long motions are stuck at a fixed difficulty point, it can be adjusted to 5, allowing sampling to cover the process before entering the difficulty point |
| adaptive_uniform_ratio | 0.1 | Ensure all time periods have the minimum sampling amount, avoid computing power concentrated on a single difficulty point |
| adaptive_alpha | 0.001 | Control the update speed of the failure histogram; increasing makes the response faster but more volatile |
| pose_range / velocity_range | Subject to configuration | Control initial state disturbance during reset, increasing can improve robustness |

### 2.4.8 PPO Hyperparameters and Network

Defined in `config/hu_l03/agents/rsl_rl_ppo_cfg.py`. The default configuration is suitable for most motions, and modification is not recommended without clear evidence.

| Parameter | Default Value | Description |
|---|---|---|
| Network | [512, 256, 128], ELU | Only consider widening when the observation dimension increases significantly; the current observation dimension is prohibited from modification |
| num_steps_per_env | 24 | Number of sampling steps per environment per round |
| learning_rate | 1e-3, adaptive | The actual learning rate is scheduled by desired KL feedback |
| desired_kl | 0.01 | Main update step size control parameter; smaller is more stable but slower training |
| entropy_coef | 0.005 | Increasing enhances exploration, too large will cause action jitter |
| gamma / lam | 0.99 / 0.95 | Discount factor and GAE parameter |
| max_iterations | 30000 | Can be overridden from the command line, in practice 15000 rounds usually enter the convergence interval |

### 2.4.9 Actuator Gains

Actuator gains are defined in `robots/hu_l03_parallel.py`, uniformly calculated from natural frequency, damping ratio, and armature in the manufacturer's MJCF:

代码块

```
NATURAL_FREQ = 10 * 2 * pi    # 10 Hz
DAMPING_RATIO = 2.0

stiffness = armature * NATURAL_FREQ ** 2
damping = 2 * DAMPING_RATIO * armature * NATURAL_FREQ
```

- **NATURAL_FREQ increase**: Tracking is stiffer and more accurate, but the real robot is more prone to jitter.
- **DAMPING_RATIO increase**: The system is more stable, but the response is more sluggish.
- **soft_joint_pos_limit_factor** defaults to 0.9; it can be carefully increased when reference motions need to be close to the limits.

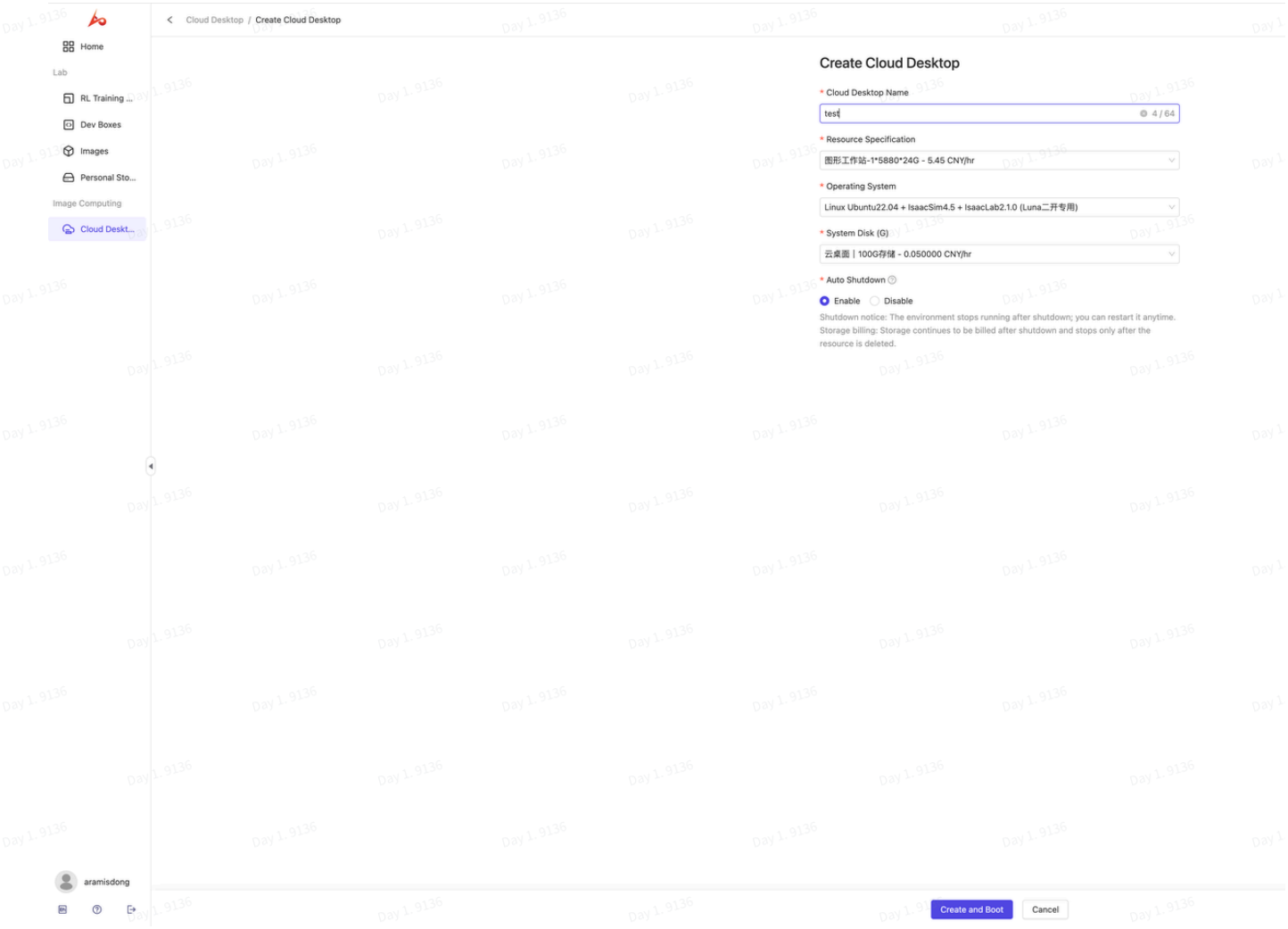
> ❗ Adjusting gains should modify the unified constants. Do not arbitrarily change values joint by joint, so as not todisrupt the relative relationship between individual joints. Any parameter adjustment result must re-complete simulation and real-robot safety verification.

# 3. Dance Motion Simulation Cloud Desktop Verification

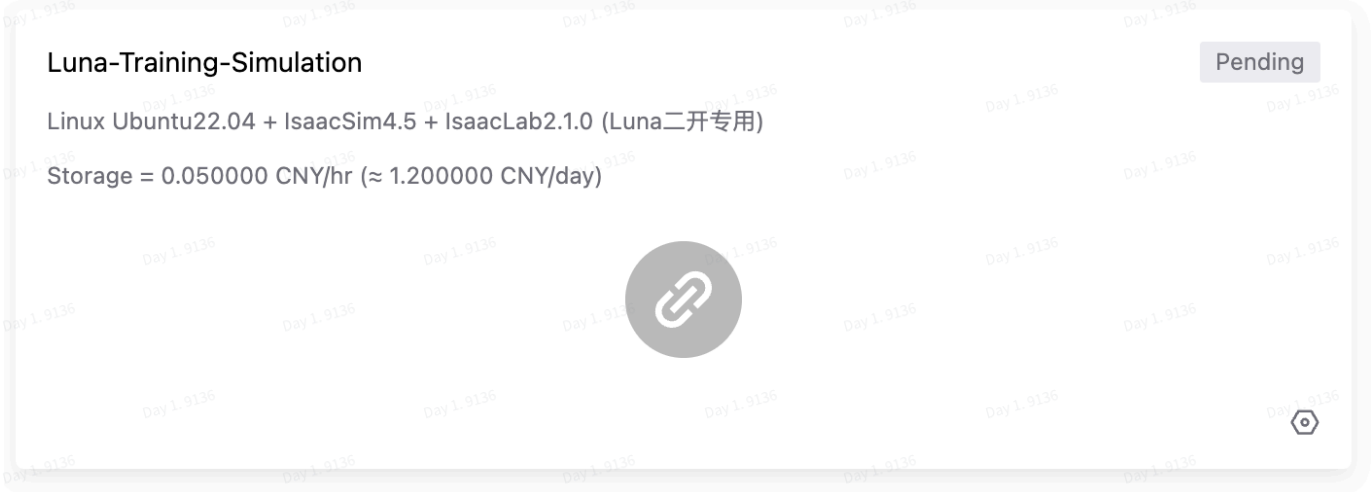
## 3.1 Cloud Desktop Creation

1. Register and log in to Flux Training (https://internal.limxdynamics.com/user/login)
2. Recharge the account (for redemption codes, please contact LimX Dynamics sales colleagues)
3. Select "Cloud Desktop" and click Add

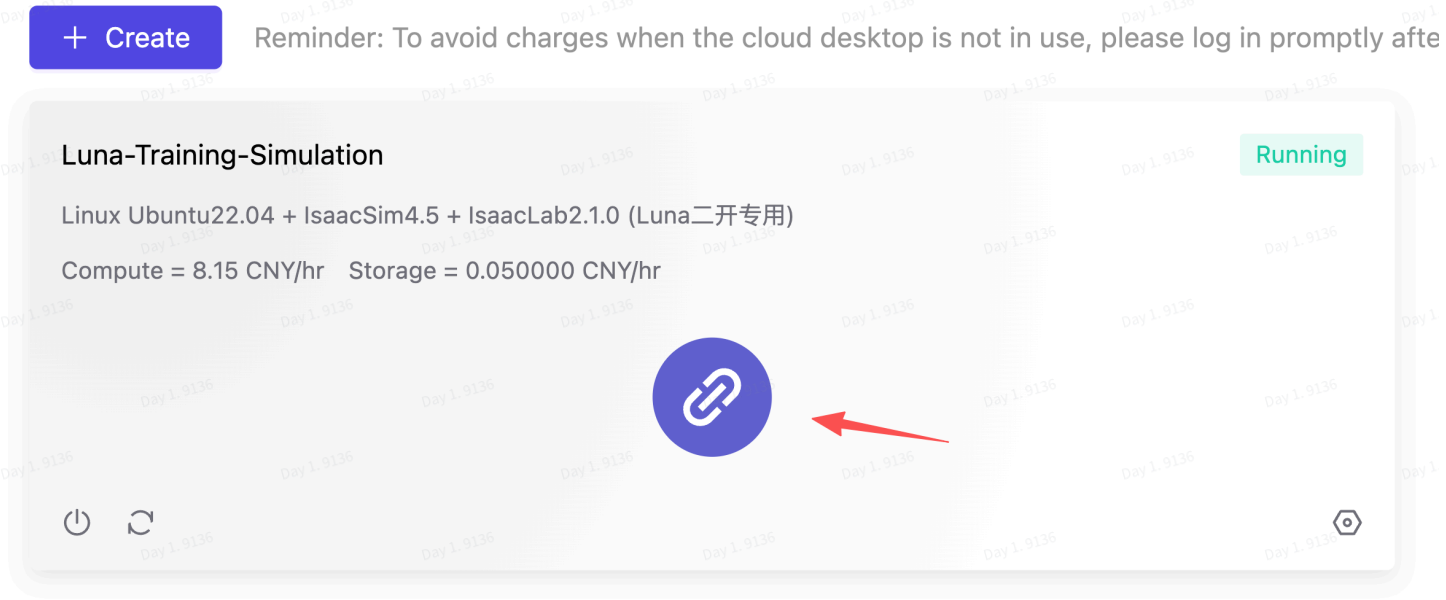

4. Create a new cloud desktop and select configuration parameters
   a. Computing power: Select as needed, 3 tiers available, 5880 16G/24G/48G
   b. Image system: Scroll to the bottom and select the image dedicated to Luna secondary development
   c. System disk: Select as needed, 2 tiers available, 100G and 200G
   d. Auto shutdown: Select as needed, the identification criterion is whether the mouse is moved



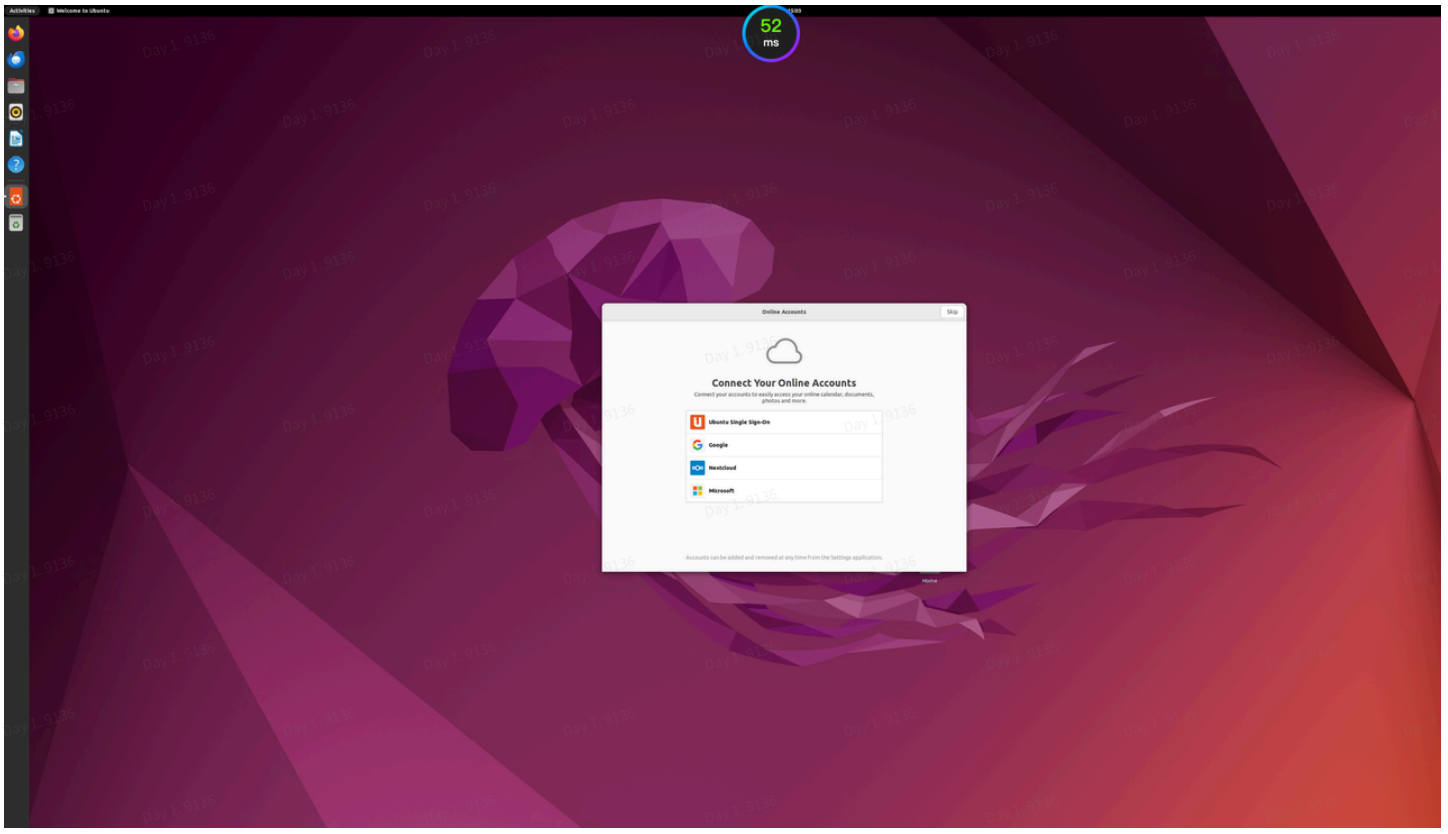

5. Create and power on, wait for boot




6. Click Login




7. Login to use




## 3.2 Simulation Running

In the home directory, there is a LunaSim folder. Enter this folder:

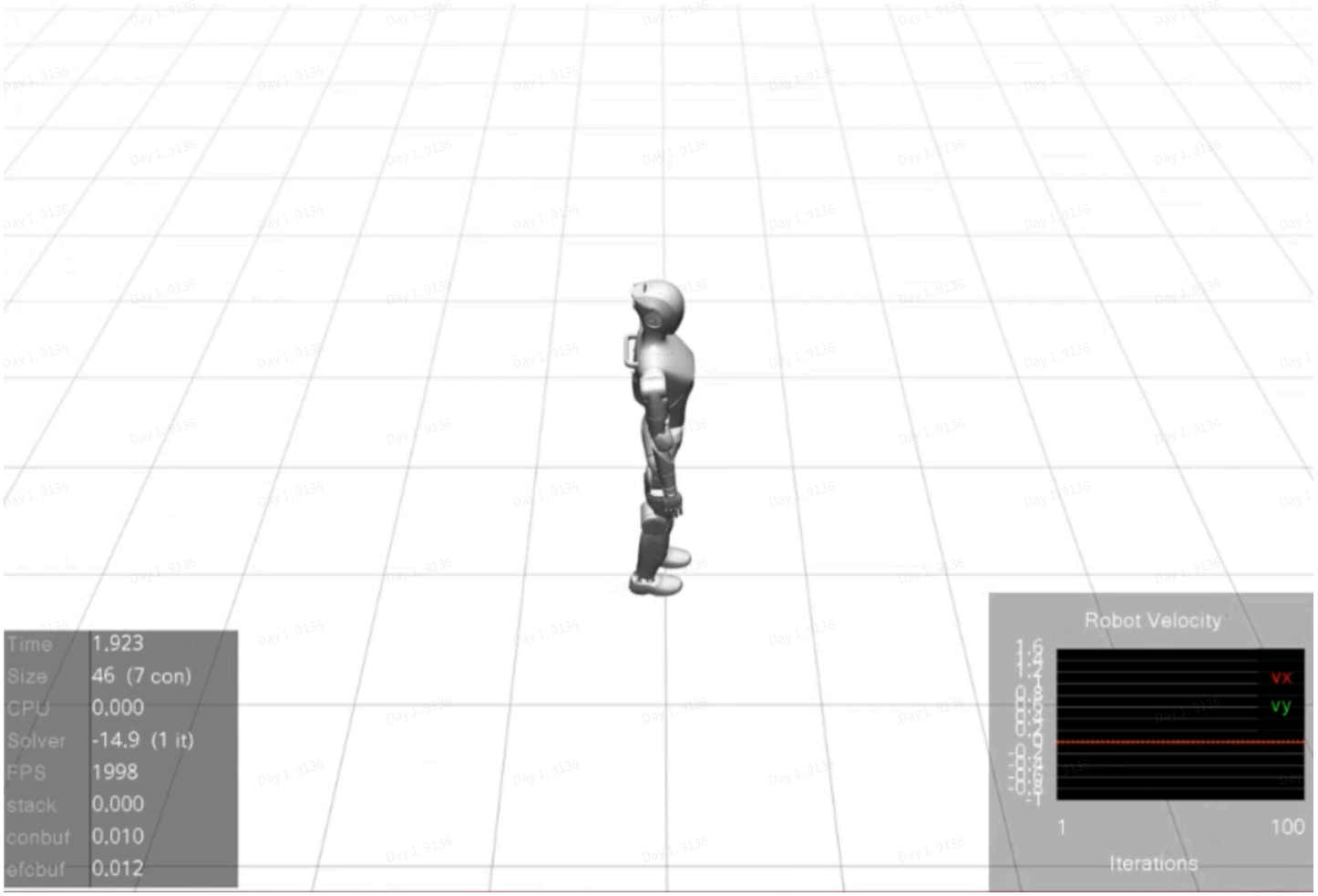
代码块

```
cd ~/LunaSim
```

Run the one-click simulation startup script:

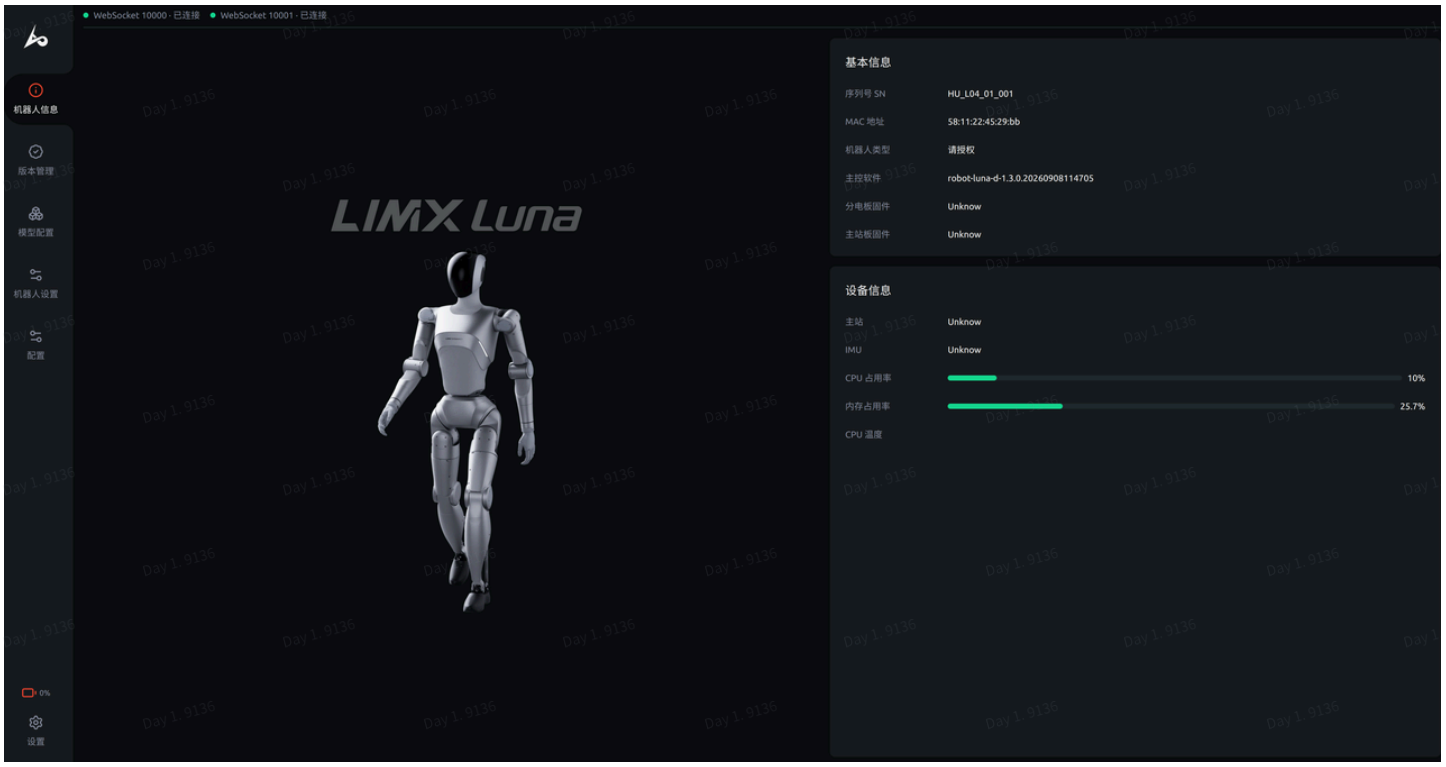
```
代码块
1   ./start_sim.sh
```

Wait for the simulation to start, the robot stands in the simulation environment

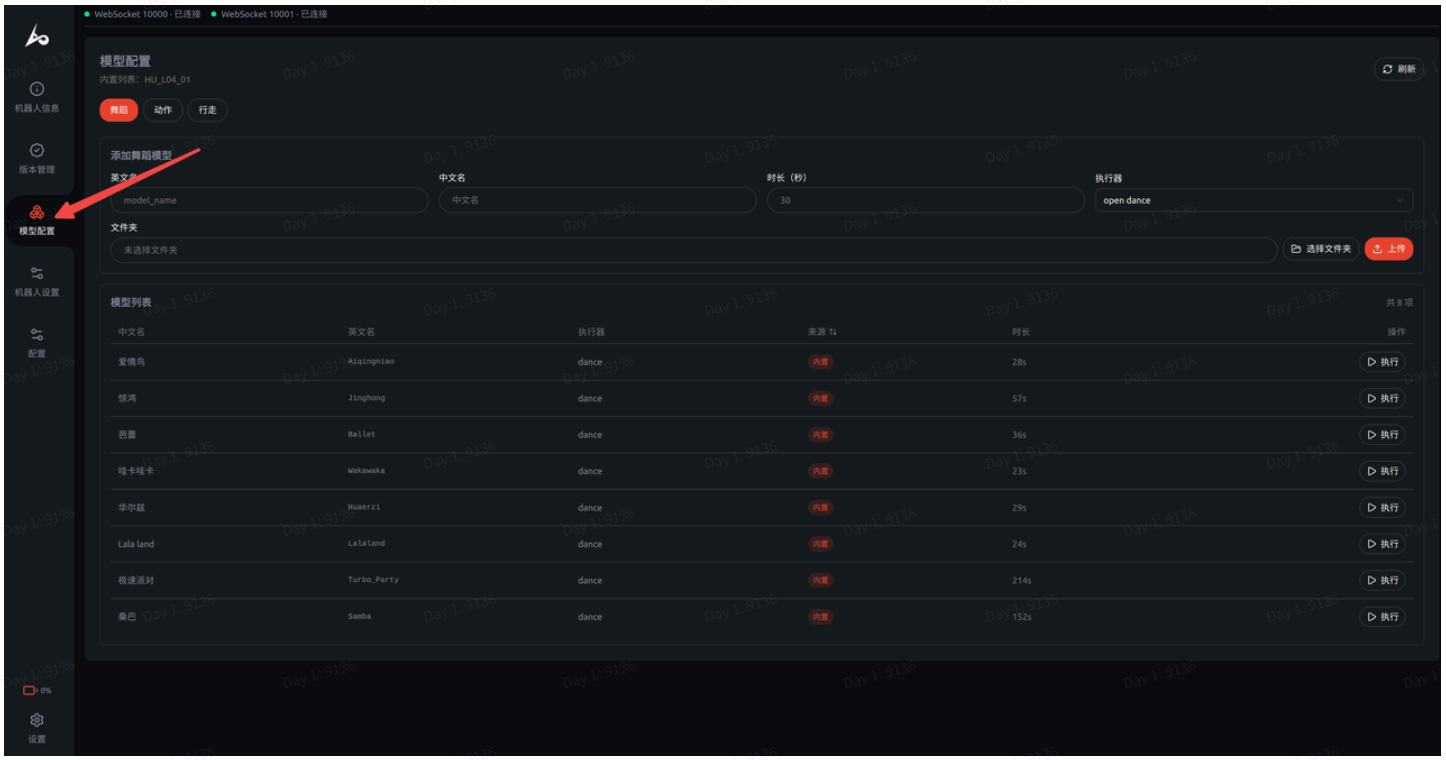



## 3.3 Dance Verification

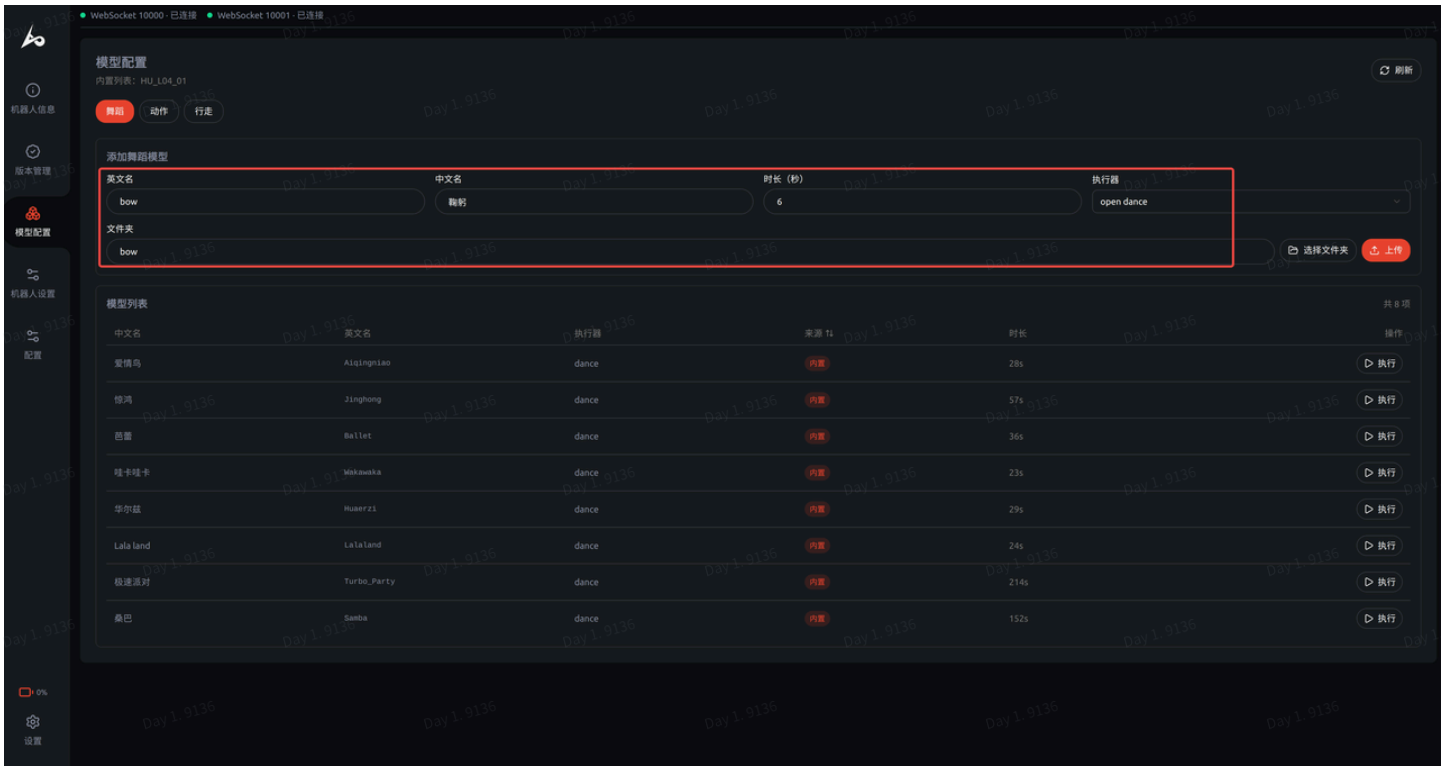
After the simulation is started, open the browser and enter `127.0.0.1:8080` in the browser address bar to enter the robot configuration interface




After observing that the simulation connection is normal, select Model Configuration




Select "Select Folder", select the folder containing the trained policy.onnx, reference trajectory reference.txt, and configuration file param.yaml, name the action you added, and click Upload:




When the model list shows the newly added action as shown in the figure below, you can click Execute, and Luna will execute the action in the simulation


（注：部分内容可能由 AI 生成）