LLM Module API

Table of Contents

Version	Update Date	Remarks
v1.0.0	2024.10.24	/

Overview

The LLM Module integrates functional units such as KWS (Keyword Spotting), ASR (Speech Recognition), LLM (Large Language Model), and TTS (Text-to-Speech). Each unit can operate independently as a standalone module or support configuration for data workflow integration, enabling more intelligent interactive applications. The module supports interaction with a host device via `UART` communication, and it uses `JSON`-formatted data packets, making it very easy to use.

Built-in Functional Units

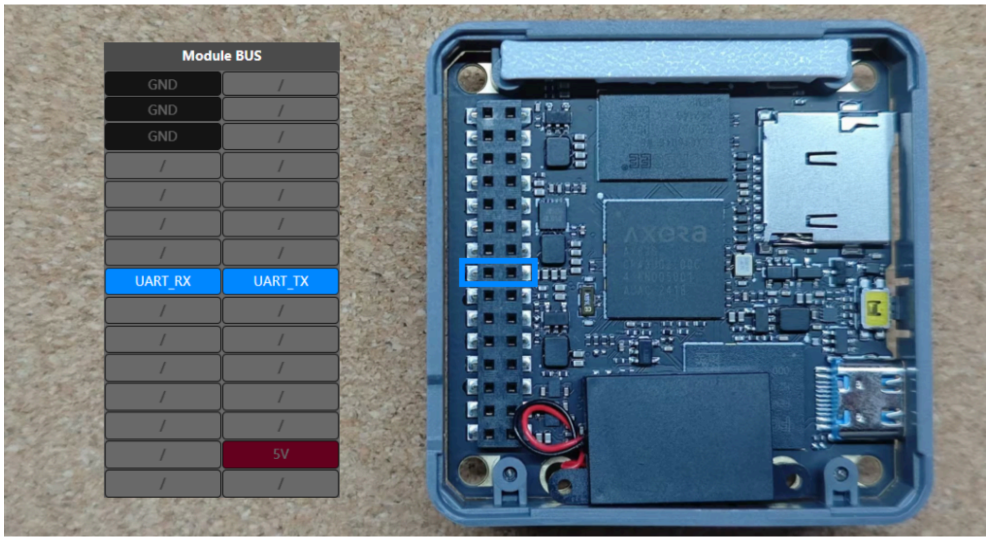
Unit	Unit Name	Unit Capability
sys	System	Set module parameters, retrieve module status
kws	Keyword Detection	Detect the presence of keywords in audio
asr	Speech-to-Text	Convert audio to text
llm	Generative Model	Generate new text based on input text
tts	Text-to-Speech	Convert text to audio
audio	System Audio Interface	Access microphone audio and playback audio

Usage Process

- 1. Stack the module with an M5Stack main controller (Basic/M5Core2/M5Core3, etc.) or connect it directly to TX/RX and power supply using a USB-TTL converter. The module will light up green upon successful startup.
- 2. Initialize the `UART` interface in the program (pin configuration based on the actual connected device, interface configuration as `115200bps 8N1`).
- 3. Refer to the usage examples below, send an initialization frame to activate the desired unit service.

Communication Interface

- The LLM Module's UART interface is configured by default as `115200bps 8N1`.



Data Packet Format

Basic Structure of Sending Frame

```
{
    "request_id": "001",
    "work_id": "llm.1001",
    "action": "taskinfo",
    "object": "None",
    "data":"None"
}
```

- `request_id`:
  - The session ID used to distinguish context, corresponding to the service invocation and response.
- `work_id`:
  - When calling the service unit, enter keyword + ID, e.g., llm.xxxx(id).
  - When initializing the service unit in setup, enter the unit name keyword without the ID, e.g., llm.
- `action`:
  - The method being called, corresponding to the unit method. Please refer to the unit list below.
- `object`:
  - Sets the structure of the parameters passed in `data`. Refer to the parameter structure list for all parameter structures. If there are no parameters, this can be omitted.
- `data`:
  - Parameters to be transmitted; can be omitted if there are no parameters.

Basic Structure of Response Frame

```
{
  "request_id": "002",
  "work_id": "kws.1002",
  "created": 30952,
  "object": "None",
  "data":"None",
  "error":{"code":0, "message":""}
}
```

- `created`:
  - The time when the operation was completed, in Unix timestamp format (seconds).
- `error`:
  - Status information for determining whether the service call succeeded or failed. For more error code information, please see the list below.

Streaming Data Sending Frame Structure

```
{
    "request_id": "4",
    "work_id": "llm.1003",
    "action": "inference",
    "object": "llm.utf-8.stream",
    "data": {
        "delta": "What's ur name?",
        "index": 0,
        "finish": true
    }
}
```

Streaming Data Response Frame Structure

```
{
    "created": 1692664605,
    "data": {
        "delta": "I'm not a person, but I'm here to help with any questions you may
have. How can I assist you today?\n",
        "finish": true,
        "index": 0
    },
    "error": {
        "code": 0,
        "message": ""
    },
    "object": "llm.utf-8.stream",
    "request_id": "4",
    "work_id": "llm.1003"
}
```

- `index`:

    - Index for identifying the data segment
- `delta`:
    - Data segment
- `finish`:
    - Indicates the final packet if set to true

Error Codes

Error codes are included in the `error` field of the response to determine the result of the response:

Error Code	Description	Message	Notes
0	Operation Successful!	Operation Successful!	
-1	Communication channel receive state machine reset warning!	reace reset	Continuously sending “}” will trigger this error, used to reset the JSON receive state machine.
-2	JSON parsing error	JSON format error	
-3	`sys action` match error	action match false	
-4	Inference data push error	inference data push false	
-5	Model loading failed	Model loading failed.	
-6	Unit does not exist	Unit Does Not Exist	
-7	Unknown operation	Unknown Operation	
-8	Unit resource allocation failed	Unit Resource Allocation Failed	
-9	Unit call failed	unit call false	
-10	Model initialization failed	Model init failed.	
-11	Model run error	Model run failed.	
-12	Module not initialized	Module has not been initialised.	
-13	Module is already working	Module already working.	

Error Code	Description	Message	Notes
-14	Module is not working	Module is not working.	
-19	Unit resource release failed	Unit Resource Free Failed	

SYS

The SYS unit is used to set module working parameters and retrieve module operation information.

Method	Function	Input Type	Output Type
lsmode	Retrieve available models	None	sys.lsmode
hwinfo	Retrieve CPU load, memory load, chip temperature	None	sys.hwinfo
reset	Reset the unit	None	Returns reset completion JSON
reboot	Reboot the system	None	None
ping	Check if the system is available	None	None

lsmode

- Retrieve available models

```
{
    "request_id": "001",
    "work_id": "sys",
    "action": "lsmode"
}
```

- Retrieve available models response

```
{
    "created": 1692652687,
    "data": [
        {
            "capabilities": [
                "Automatic_Speech_Recognition"
            ],
            "input_type": [
                "sys.pcm"
            ],
            "model": "sherpa-ncnn-streaming-zipformer-zh-14M-2023-02-23",
            "output_type": [
                "asr.utf-8"
            ],
            "type": "asr"
        },
        {
            "capabilities": [
                "Automatic_Speech_Recognition"
            ],
            "input_type": [
                "sys.pcm"
            ],
            "model": "sherpa-ncnn-streaming-zipformer-20M-2023-02-17",
            "output_type": [
                "asr.utf-8"
            ],
            "type": "asr"
        },
        {
            "capabilities": [
                "Keyword_spotting"
            ],
            "input_type": [
                "sys.pcm"
            ],
            "model": "sherpa-onnx-kws-zipformer-wenetspeech-3.3M-2024-01-01",
            "output_type": [
                "kws.bool"
            ],
            "type": "kws"
        },
        {
            "capabilities": [
                "Keyword_spotting"
            ],
            "input_type": [
                "sys.pcm"
            ],
            "model": "sherpa-onnx-kws-zipformer-gigaspeech-3.3M-2024-01-01",
            "output_type": [
                "kws.bool"
            ],
            "type": "kws"
        },
        {
            "capabilities": [
                "text_generation",
                "chat"
            ],
            "input_type": "utf-8",
            "model": "qwen2.5-0.5b",
            "output_type": "utf-8",
            "type": "llm"
        },
        {
            "capabilities": [
                "Text_to_speech"
            ],
            "input_type": [
                "sys.utf-8",
                "llm.utf-8"
            ],
            "model": "single_speaker_fast",
            "output_type": [
                "tts.wav"
            ],
            "type": "tts"
        },
        {
            "capabilities": [
                "Text_to_speech"
            ],
            "input_type": [
                "sys.utf-8",
                "llm.utf-8"
            ],
            "model": "single_speaker_english_fast",
            "output_type": [
                "tts.wav"
            ],
            "type": "tts"
        }
    ],
    "error": {
        "code": 0,
        "message": ""
    },
    "object": "sys.lsmode",
    "request_id": "001",
    "work_id": "sys"
}
```

hwinfo

- Retrieve CPU load, memory load, and chip temperature

```
{
    "request_id": "001",
    "work_id": "sys",
    "action": "hwinfo"
}
```

- Response for retrieving CPU load, memory load, and chip temperature (cpu_loadavg(0%), mem(18%), temperature(46°C))

```
{
    "created": 1692652642,
    "data": {
        "cpu_loadavg": 0,
        "mem": 18,
        "temperature": 46350
    },
    "error": {
        "code": 0,
        "message": ""
    },
    "object": "sys.hwinfo",
    "request_id": "001",
    "work_id": "sys"
}
```

reset

- System reset command.

```
{
    "request_id": "001",
    "work_id": "sys",
    "action": "reset"
}
```

- System reset initiated.

```
{
    "created": 1692652712,
    "error": {
        "code": 0,
        "message": "llm server restarting ..."
    },
    "request_id": "001",
    "work_id": "sys"
}
```

- System reset completion response.

```
{
    "request_id": "0",
    "work_id": "sys",
    "created": 1692652723,
    "error": {
        "code": 0,
        "message": "reset over"
    }
}
```

reboot

- Full system reboot command.

```
{
    "request_id": "001",
    "work_id": "sys",
    "action": "reboot"
}
```

- Full system reboot command.

```
{
    "created": 1692652822,
    "error": {
        "code": 0,
        "message": "rebooting ..."
    },
    "request_id": "001",
    "work_id": "sys"
}
```

- Note: After the response message, the system will reboot. During the reboot, a string `V0EUEURS` will be sent, which is the system startup string and can be ignored.

ping

- System service communication test, useful for checking communication status after module power-up.

```
{
    "request_id": "001",
    "work_id": "sys",
    "action": "ping"
}
```

- System service communication test response

```
{
    "created": 1692652310,
    "error": {
        "code": 0,
        "message": ""
    },
    "request_id": "001",
    "work_id": "sys"
}
```

AUDIO

The AUDIO unit is used to control the system sound card, access microphone audio, and playback sound. It provides system audio input and output, supplying audio input for the Keyword Spotting (KWS) and Automatic Speech Recognition (ASR) units and audio output for the Text-to-Speech (TTS) module. The AUDIO unit must be initialized before using the `KWS` and `ASR` units.

Method	**Function**	**Input Type**	**Output Type**
setup	Configure audio unit	audio.setup	None (the returned result includes the successful `work_id`)
exit	End the work of `work_id`	None	None
pause	Pause task operation	None	None
work	Resume task operation	None	None
taskinfo	Retrieve all task instance information		audio.taskinfo

setup

- Initialize the Audio unit and configure playback volume and sound card slot number (capcard, playcard use defaults)

Parameter Description

Parameter	Description	Input Value
capcard	Microphone sound card index	Default system sound card: 0
capdevice	Microphone device index	Onboard silicon mic: 0
capVolume	Input volume	0.0 ~ 10.0 (volume > 1 will amplify, default is 0.5)
playcard	Speaker sound card index	Default system sound card: 0
playdevice	Speaker device index	Onboard speaker: 1
playVolume	Output volume	0.0 ~ 10.0 (volume > 1 will amplify, default is 0.5)

```
{
    "request_id": "1",
    "work_id": "audio",
    "action": "setup",
    "object": "audio.setup",
    "data": {
        "capcard": 0,
        "capdevice": 0,
        "capVolume": 0.5,
        "playcard": 0,
        "playdevice": 1,
        "playVolume": 0.5
    }
}
```

- Response for Audio unit initialization

```
{
    "created": 1692659008,
    "error": {
        "code": 0,
        "message": "audio setup successful"
    },
    "request_id": "1",
    "work_id": "audio.1000"
}
```

pause

- Pause Audio unit command

```
{
    "request_id": "1",
    "work_id": "audio.1000",
    "action": "pause"
}
```

- Response for Audio unit pause command

```
{
    "created": 1692659049,
    "error": {
        "code": 0,
        "message": "audio pause"
    },
    "request_id": "1",
    "work_id": "audio.1000"
}
```

work

- Start Audio unit command

```
{
    "request_id": "1",
    "work_id": "audio.1000",
    "action": "work",
    "object": "audio.setup",
    "data": {
        "capcard": 0,
        "capdevice": 0,
        "capVolume": 0.5,
        "playcard": 0,
        "playdevice": 1,
        "playVolume": 0.25
    }
}
```

- Response for Audio unit start command

```
{
    "created": 1692659297,
    "error": {
        "code": 0,
        "message": "audio work start"
    },
    "request_id": "1",
    "work_id": "audio.1000"
}
```

exit

- End and release the Audio unit

```
{
    "request_id": "1",
    "work_id": "audio.1000",
    "action": "exit"
}
```

- Response for ending and releasing the Audio unit

```
{
    "created": 1692659370,
    "error": {
        "code": 0,
        "message": "audio exit"
    },
    "request_id": "1",
    "work_id": "audio.1000"
}
```

taskinfo

- Query Audio unit status

```
// Sending data
{
    "request_id": "1",
    "work_id": "audio.1000",
    "action": "taskinfo"
}
```

- Response when the Audio unit is running

```
{
    "created": 1692659454,
    "data": "running",
    "error": {
        "code": 0,
        "message": ""
    },
    "object": "audio.state",
    "request_id": "1",
    "work_id": "audio.1000"
}
```

- Response when the Audio unit is stopped

```
{
    "created": 1692659499,
    "data": "stopped",
    "error": {
        "code": 0,
        "message": ""
    },
    "object": "audio.state",
    "request_id": "1",
    "work_id": "audio.1000"
}
```

- Response when the Audio unit is released

```
{
    "created": 1692659403,
    "data": "deinit",
    "error": {
        "code": 0,
        "message": ""
    },
    "object": "audio.state",
    "request_id": "1",
    "work_id": "audio.1000"
}
```

KWS

The KWS unit is used for keyword detection.

Method	Function	Input Type	Output Type
setup	Configure KWS unit	kws.setup	None (the returned result includes the successful `work_id`)
pause	Pause task operation	None	None
work	Resume task operation	None	None
exit	End the work of `work_id`	None	None
taskinfo	Retrieve all task instance information		kws.taskinfo

setup

- Initialize the KWS unit and configure for Chinese/English recognition model. (Note: KWS keyword field does not allow a mix of Chinese/English)

Parameter Description

Parameter	Description	Input Value
model	Conversion model	English model: "sherpa-onnx-kws-zipformer-gigaspeech-3.3M-2024-01-01" Chinese model: "sherpa-onnx-kws-zipformer-wenetspeech-3.3M-2024-01-01"
kws	KWS keyword text setup	Mixing Chinese/English is not allowed, English should be in all uppercase
enoutput	Enable UART output	Enable: true Disable: false

KWS Setup

- Initialize the KWS unit and configure for the English recognition model.

```
{
    "request_id": "2",
    "work_id": "kws",
    "action": "setup",
    "object": "kws.setup",
    "data": {
        "model": "sherpa-onnx-kws-zipformer-gigaspeech-3.3M-2024-01-01",
        "response_format": "kws.bool",
        "input": "sys.pcm",
        "enoutput": true,
        "kws": "HELLO"
    }
}
```

- KWS unit initialization response (Note: the setup process takes approximately 9 seconds)

```
{
    "created": 1692660576,
    "error": {
        "code": 0,
        "message": "kws setup successful"
    },
    "request_id": "2",
    "work_id": "kws.1001"
}
```

- KWS response after keyword trigger

```
{
    "created": 1692660576,
    "error": {
        "code": 0,
        "message": "kws setup successful"
    },
    "request_id": "2",
    "work_id": "kws.1001"
}
```

pause

- Pause KWS unit command

```
{
    "request_id": "2",
    "work_id": "kws.1001",
    "action": "pause"
}
```

- Response for pausing KWS unit command

```
{
    "created": 1692660626,
    "error": {
        "code": 0,
        "message": "kws pause"
    },

    "request_id": "2",
    "work_id": "kws.1001"
}
```

work

- Start KWS unit command

```
{
    "request_id": "2",
    "work_id": "kws.1001",
    "action": "work"
}
```

- Response for starting KWS unit command

```
{
    "created": 1692660651,
    "error": {
        "code": 0,
        "message": "kws work"
    },
    "request_id": "2",
    "work_id": "kws.1001"
}
```

exit

- End and release the KWS unit

```
{
  "request_id": "2",
  "work_id": "kws.1001",
  "action": "exit"
}
```

- Response for ending and releasing the KWS unit

```
{
  "created": 1692654383,
  "error": {
    "code": 0,
    "message": "kws exit"
  },
  "request_id": "2",
  "work_id": "kws.1001"
}
```

taskinfo

- Query KWS unit status

```
{
  "created": 1692654383,
  "error": {
    "code": 0,
    "message": "kws exit"
  },
  "request_id": "2",
  "work_id": "kws.1001"
}
```

- Response when the KWS unit is running

```
{
  "created": 1692654305,
  "error": {
    "code": 0,
    "message": ""
  },
  "object": "kws.state",
  "data": "running",
  "request_id": "2",
  "work_id": "kws.1001"
}
```

- Response when the KWS unit is stopped

```
{
  "created": 1692654535,
  "error": {
    "code": 0,
    "message": ""
  },
  "object": "kws.state",
  "data": "stopped",
  "request_id": "2",
  "work_id": "kws.1001"
}
```

- KWS unit release response

```
{
  "created": 1692654452,
  "error": {
    "code": 0,
    "message": ""
  },
  "object": "kws.state",
  "data": "deinit",
  "request_id": "2",
  "work_id": "kws.0"
}
```

ASR

The ASR unit is used for converting speech to text.

Method	Function	Input Type	Output Type
setup	Configure ASR unit	asr.setup	None (the returned result includes the successful `work_id`)
pause	Pause task operation	None	None
work	Resume task operation	None	None
exit	End the work of `work_id`	None	None
taskinfo	Retrieve all task instance information		asr.taskinfo

setup

- Initialize the ASR unit and configure for Chinese/English model.

Parameter Description

Parameter	Description	Input Value
model	Conversion model	English model: "sherpa-ncnn-streaming-zipformer-20M-2023-02-17" Chinese model: "sherpa-ncnn-streaming-zipformer-zh-14M-2023-02-23"
response_for mat	Output format	Standard output: "asr.utf-8" Streaming output: "asr.utf-8.stream"
input	Input	LLM input: "llm.xxx" (input work_id of the llm unit) UART input: "tts.utf-8" UART streaming input: "tts.utf-8.stream"
enkws	Enable KWS-based activation	Activation via KWS, followed by ASR: true No KWS activation, ASR will operate continuously: false
rule1	Timeout from activation to unrecognized content	Unit: seconds
rule2	Maximum interval time for recognition	Unit: seconds
rule3	Maximum recognition timeout	Unit: seconds
enoutput	Enable UART output	Enable: true Disable: false

ASR Setup

- Initialize the ASR unit and configure for the English model.

```
{
    "request_id": "3",
    "work_id": "asr",
    "action": "setup",
    "object": "asr.setup",
    "data": {
        "model": "sherpa-ncnn-streaming-zipformer-20M-2023-02-17",
        "response_format": "asr.utf-8",
        "input": "sys.pcm",
        "enoutput": true,
        "enkws": true,
        "rule1": 2.4,
        "rule2": 1.2,
        "rule3": 30
    }
}
```

- ASR unit initialization response

```
{
    "created": 1692667736,
    "error": {
        "code": 0,
        "message": "asr setup successful"
    },
    "request_id": "3",
    "work_id": "asr.1002"
}
```

- ASR trigger response

```
{
    "created": 1692655176,
    "data": {
        "delta": " hello",
        "index": "0"
    },
    "object": "asr.stream",
    "request_id": "004",
    "work_id": "asr.1003"
}
```

pause

- Pause ASR unit command

```
{
    "request_id": "3",
    "work_id": "asr.1002",
    "action": "pause"
}
```

- Pause ASR unit command response

```
{
    "created": 1692670174,
    "error": {
        "code": 0,
        "message": "asr pause"
    },
    "request_id": "3",
    "work_id": "asr.1002"
}
```

work

- Start ASR unit command

```
{
    "request_id": "3",
    "work_id": "asr.1002",
    "action": "work"
}
```

- Start ASR unit command response

```
{
    "created": 1692670213,
    "error": {
        "code": 0,
        "message": "asr work"
    },
    "request_id": "3",
    "work_id": "asr.1002"
}
```

exit

- End and release the ASR unit

```
{
    "request_id": "3",
    "work_id": "asr.1002",
    "action": "exit"
}
```

- ASR unit release response

```
{
    "created": 1692670254,
    "error": {
        "code": 0,
        "message": "asr exit"
    },
    "request_id": "3",
    "work_id": "asr.1002"
}
```

taskinfo

- Query ASR unit status

```
{
    "request_id": "3",
    "work_id": "asr.1002",
    "action": "taskinfo"
}
```

- Response when ASR unit is running

```
{
    "created": 1692669923,
    "data": "running",
    "error": {
        "code": 0,
        "message": ""
    },
    "object": "asr.state",
    "request_id": "3",
    "work_id": "asr.1002"
}
```

- Response when ASR unit is stopped

```
{
  "created": 1692653792,
  "data": "stopped",
  "error": {
    "code": 0,
    "message": ""
  },
  "object": "asr.state",
  "request_id": "3",
  "work_id": "asr.1002"
}
```

- Response when ASR unit is released

```
{
    "created": 1692669874,
    "data": "deinit",
    "error": {
        "code": 0,
        "message": ""
    },
    "object": "asr.state",
    "request_id": "3",
    "work_id": "asr.0"
}
```

LLM

The LLM (Large Language Model) unit can generate responses based on input text.

Method	Function	Input Type	Output Type
setup	Configure LLM unit	llm.setup	None (the returned result includes the successful `work_id`)
inference	Perform inference	Typical: llm.utf-8 (model difference can be checked via `sys.lsmode`)	None (returns only data submission result; final inference result will depend on configuration)
pause	Pause task operation	None	None
work	Resume task operation	None	None
exit	End the work of `work_id`	None	None
taskinfo	Retrieve all task instance information		llm.taskinfo

setup

- Initialize the LLM unit and configure a specified model. Current pre-installed model:
  - `qwen2.5-0.5b`

Parameter Description

Parameter	Description	Input Value
model	Conversion model	Pre-installed model "qwen2.5-0.5b"
response_format	Output format	Standard output: "llm.utf-8" Streaming output: "llm.utf-8.stream"
input	Input	ASR input: "asr.xxx" (input work_id of the ASR unit) UART input: "llm.utf-8" UART streaming input: "llm.utf-8.stream"
enkws	KWS interruption of ongoing process	Interrupt with KWS: true Do not interrupt with KWS: false
max_length	Configure max output token length	Maximum: 1024, recommended: 127
prompt	Model initialization prompt	
enoutput	Enable UART output	Enable: true Disable: false

LLM Input From ASR

- Initialize LLM unit and configure ASR (speech-to-text) as input data

```
// Input from ASR
{
    "request_id": "4",
    "work_id": "llm",
    "action": "setup",
    "object": "llm.setup",
    "data": {
        "model": "qwen2.5-0.5b",
        "response_format": "llm.utf-8.stream",
        "input": "asr.1001",
        "enoutput": true,
        "enkws": true,
        "max_token_len": 127,
        "prompt": "You are a knowledgeable assistant capable of answering various
questions and providing information."
```

```
    }
}
```

LLM Input From UART

- Initialize LLM unit and configure UART interface as input data

```
// Input from UART
{
    "request_id": "4",
    "work_id": "llm",
    "action": "setup",
    "object": "llm.setup",
    "data": {
        "model": "qwen2.5-0.5b",
        "response_format": "llm.utf-8",
        "input": "llm.utf-8.stream",
        "enoutput": true,
        "enkws": true,
        "max_token_len": 127,
        "prompt": "You are a knowledgeable assistant capable of answering various
questions and providing information."
    }
}
```

- LLM unit initialization response

```
{
    "created": 1692664107,
    "data": "None",
    "error": {
        "code": 0,
        "message": "llm setup successful"
    },
    "object": "None",
    "request_id": "4",
    "work_id": "llm.1003"
}
```

inference

UART inference

- Submit inference data via UART

```
// Streaming Input
{
    "request_id": "4",
    "work_id": "llm.1003",
    "action": "inference",
    "object": "llm.utf-8.stream",
    "data": {
        "delta": "What's ur name?",
        "index": 0,
        "finish": true
    }
}
// Non-Streaming Input
{
    "request_id": "4",
    "work_id": "llm.1003",
    "action": "inference",
    "object": "llm.utf-8",
    "data": "What's ur name?"
}
```

- Inference response data

```
{
    "created": 1692664605,
    "data": {
        "delta": "I'm not a person, but I'm here to help with any questions you may have. How can I assist you today?\n",
        "finish": true,
        "index": 0
    },
    "error": {
        "code": 0,
        "message": ""
    },
    "object": "llm.utf-8.stream",
    "request_id": "4",
    "work_id": "llm.1003"
}
```

pause

- Pause LLM unit command

```
{
    "request_id": "4",
    "work_id": "llm.1003",
    "action": "pause"
}
```

- LLM unit pause command response

```
{
    "created": 1692664941,
    "error": {
        "code": 0,
        "message": "llm pause"
    },
    "request_id": "4",
    "work_id": "llm.1003"
}
```

work

- Start LLM unit command

```
{
    "request_id": "4",
    "work_id": "llm.1003",
    "action": "work"
}
```

- LLM unit start command response

```
{
    "created": 1692664972,
    "error": {
        "code": 0,
        "message": "llm work"
    },
    "request_id": "4",
    "work_id": "llm.1003"
}
```

exit

- End and release LLM unit

```
{
    "request_id": "4",
    "work_id": "llm.1003",
    "action": "exit"
}
```

- LLM unit release response

```
{
    "created": 1692664858,
    "data": "None",
    "error": {
        "code": 0,
        "message": "llm exit"
    },
    "object": "None",
    "request_id": "4",
    "work_id": "llm.1003"
}
```

taskinfo

- Query LLM unit status

```
{
    "request_id": "4",
    "work_id": "llm.1003",
    "action": "taskinfo"
}
```

- LLM unit running response

```
{
    "created": 1692664730,
    "data": "running",
    "error": {
        "code": 0,
        "message": ""
    },
    "object": "llm.state",
    "request_id": "4",
    "work_id": "llm.1003"
}
```

- LLM unit stopped response

```
{
    "created": 1692664823,
    "data": "stopped",
    "error": {
        "code": 0,
        "message": ""
    },
    "object": "llm.state",
    "request_id": "4",
    "work_id": "llm.1003"
}
```

- LLM unit release response

```
{
    "created": 1692664881,
    "data": "deinit",
    "error": {
        "code": 0,
        "message": ""
    },
    "object": "llm.state",
    "request_id": "4",
    "work_id": "llm.1003"
}
```

TTS

The TTS unit is used for converting text to speech.

Method	Function	Input Type	Output Type
setup	Configure TTS unit	tts.setup	None (the returned result includes the successful `work_id`)
inference	Perform inference	Typical: tts.utf-8 (model difference can be checked via `sys.lsmode`)	None (returns only data submission result; final inference result will depend on configuration)

Method	Function	Input Type	Output Type
pause	Pause task operation	None	None
work	Resume task operation	None	None
exit	End the work of `work_id`	None	None
taskinfo	Retrieve all task instance information		tts.taskinfo

setup

- Initialize the TTS unit and configure for Chinese/English model.

Parameter Description

Parameter	Description	Input Value
model	Conversion model	English model: "single_speaker_english_fast" Chinese model: "single_speaker_fast"
input	Input	LLM input: "llm.xxx" (input work_id of the llm unit) UART input: "tts.utf-8" UART streaming input: "tts.utf-8.stream"
enkws	KWS interruption of process	Interrupt with KWS: true Do not interrupt with KWS: false
enoutput	Enable UART output	Enable: true Disable: false

TTS Input From LLM

- Initialize the TTS unit, configure for English text-to-speech model, and set LLM inference results as input.

```
// Input from LLM
{
    "request_id": "5",
    "work_id": "tts",
    "action": "setup",
    "object": "tts.setup",
    "data": {
        "model": "single_speaker_english_fast",
        "response_format": "tts.base64.wav",
        "input": "llm.1004",
        "enoutput": true,
        "enkws": true
    }
}
```

TTS Input From UART

- Initialize the TTS unit and configure for the English text-to-speech model, with input configured for UART command streaming input.

```
// Input from UART
{
  "request_id": "5",
  "work_id": "tts",
  "action": "setup",
  "object": "tts.setup",
  "data": {
        "model": "single_speaker_english_fast",
        "response_format": "tts.base64.wav",
        "input": "tts.utf-8.stream",
        "enoutput": true,
        "enkws": true
  }
}
```

- TTS unit initialization response

```
{
    "created": 1692668824,
    "error": {
        "code": 0,
        "message": "tts setup successful"
    },
    "request_id": "5",
    "work_id": "tts.1004"
}
```

inference

UART inference

- Submit TTS conversion data content via UART. Each model only supports one language at a time; to convert a different language, please use `exit` to release the TTS unit and reinitialize with `setup`.

- Note: Text for conversion must end with a period:
  - For English text, use an English period `.` (half-width symbol)
  - For Chinese text, use a Chinese period `。` (full-width symbol)
  - Sentence delimiters should use `,` (half-width symbol)

```
// Streaming Input
{
    "request_id": "4",
    "work_id": "tts.1004",
    "action": "inference",
    "object": "tts.utf-8.stream",
    "data": {
        "delta": "I don't know what your name.",
        "index": 0,
        "finish": true
    }
}

// Non-Streaming Input
{
    "request_id": "4",
    "work_id": "tts.1004",
    "action": "inference",
    "object": "tts.utf-8",
    "data": "I don't know what your name."
}
```

pause

- Pause TTS unit command

```
{
    "request_id": "5",
    "work_id": "tts.1004",
    "action": "pause"
}
```

- Pause TTS unit command response

```
{
    "created": 1692668916,
    "error": {
        "code": 0,
        "message": "tts pause"
    },
    "request_id": "5",
    "work_id": "tts.1004"
}
```

work

- Start TTS unit command

```
{
    "request_id": "5",
    "work_id": "tts.1004",
    "action": "work"
}
```

- Start TTS unit command response

```
{
    "created": 1692668944,
    "error": {
        "code": 0,
        "message": "tts work"
    },
    "request_id": "5",
    "work_id": "tts.1004"
}
```

exit

- End and release the TTS unit

```
{
    "request_id": "5",
    "work_id": "tts.1004",
    "action": "exit"
}
```

- TTS unit release response

```
{
    "created": 1692669052,
    "error": {
        "code": 0,
        "message": "tts exit"
    },
    "request_id": "5",
    "work_id": "tts.1004"
}
```

taskinfo

- Query TTS unit status

```
{
    "request_id": "5",
    "work_id": "tts.1004",
    "action": "taskinfo"
}
```

- TTS unit running response

```
{
    "created": 1692668878,
    "data": "running",
    "error": {
        "code": 0,
        "message": ""
    },
    "object": "tts.state",
    "request_id": "5",
    "work_id": "tts.1004"
}
```

- TTS unit stopped response

```
{
    "created": 1692668968,
    "data": "stopped",
    "error": {
        "code": 0,
        "message": ""
    },
    "object": "tts.state",
    "request_id": "5",
    "work_id": "tts.1004"
}
```

- TTS unit release response

```
{
    "created": 1692669081,
    "data": "deinit",
    "error": {
        "code": 0,
        "message": ""
    },
    "object": "tts.state",
    "request_id": "5",
    "work_id": "tts.1004"
}
```

Application

Text To Speech

Convert text to speech via the TTS unit. (TTS)

- 1. Initialize Audio unit

```
{
    "request_id": "1",
    "work_id": "audio",
    "action": "setup",
    "object": "audio.setup",
    "data": {
        "capcard": 0,
        "capdevice": 0,
        "capVolume": 0.5,
        "playcard": 0,
        "playdevice": 1,
        "playVolume": 0.5
    }
}
```

- Audio unit initialization response

```
{
    "created": 1692652475,
    "error": {
        "code": 0,
        "message": "audio setup successful"
    },
    "request_id": "1",
    "work_id": "audio.1000"
}
```

- 2. Initialize the TTS unit and configure for English text-to-speech model, with input configured for UART command input.

```
// Input from UART
{
  "request_id": "5",
  "work_id": "tts",
  "action": "setup",
  "object": "tts.setup",
  "data": {
        "model": "single_speaker_english_fast",
        "response_format": "tts.base64.wav",
        "input": "tts.utf-8",
        "enoutput": true,
        "enkws": true
  }
}
```

- TTS unit initialization response

```
{
    "created": 1692652569,
    "error": {
        "code": 0,
        "message": "tts setup successful"
    },
    "request_id": "5",
    "work_id": "tts.1001"
}
```

- 3. Input text to start TTS conversion.

```
{
    "request_id": "4",
    "work_id": "tts.1001",
    "action": "inference",
    "object": "tts.utf-8",
    "data": "Hello My Friend."
}
```

Text Assistant

Input content via text to the LLM model, process inference, and play back as speech. (LLM+TTS)

- 1. Initialize Audio unit

```
{
    "request_id": "1",
    "work_id": "audio",
    "action": "setup",
    "object": "audio.setup",
    "data": {
        "capcard": 0,
        "capdevice": 0,
        "capVolume": 0.5,
        "playcard": 0,
        "playdevice": 1,
        "playVolume": 0.5
    }
}
```

- Audio unit initialization response

```
{
    "created": 1692652330,
    "error": {
        "code": 0,
        "message": "audio setup successful"
    },
    "request_id": "1",
    "work_id": "audio.1000"
}
```

- 2. Initialize the LLM unit and configure UART interface as input data

```
// Input from UART
{
    "request_id": "4",
    "work_id": "llm",
    "action": "setup",
    "object": "llm.setup",
    "data": {
        "model": "qwen2.5-0.5b",
        "response_format": "llm.utf-8",
        "input": "llm.utf-8",
        "enoutput": true,
        "enkws": true,
        "max_token_len": 127,
        "prompt": "You are a knowledgeable assistant capable of answering various
questions and providing information."
    }
}
```

- LLM unit initialization response

```
{
    "created": 1692652323,
    "error": {
        "code": 0,
        "message": "llm setup successful"
    },
    "request_id": "4",
    "work_id": "llm.1001"
}
```

- 3. Initialize the TTS unit and configure for English text-to-speech model, with input configured for LLM inference results.

```
// Input from LLM
{
    "request_id": "5",
    "work_id": "tts",
    "action": "setup",
    "object": "tts.setup",
    "data": {
        "model": "single_speaker_english_fast",
        "response_format": "tts.base64.wav",
        "input": "llm.1001",
        "enoutput": true,
        "enkws": true
    }
}
```

- TTS unit initialization response

```
{
    "created": 1692652354,
    "error": {
        "code": 0,
        "message": "tts setup successful"
    },
    "request_id": "5",
    "work_id": "tts.1002"
}
```

- 4. Submit inference data via UART

```
// Non-Streaming Input
{
    "request_id": "4",
    "work_id": "llm.1001",
    "action": "inference",
    "object": "llm.utf-8",
    "data": "What's ur name?"
}
```

- 5. Inference response data, with audio output.

```
{
    "created": 1692652407,
    "data": "I'm not a person, but I'm here to help with any questions you may have.
How can I assist you today?\n",
    "error": {
        "code": 0,
        "message": ""
    },
    "object": "llm.utf-8",
    "request_id": "4",
    "work_id": "llm.1001"
}
```

Voice Assistant

Use KWS for activation -> trigger ASR for speech-to-text -> use converted content as LLM input for inference -> finally output the inference result as speech via TTS. (KWS+ASR+LLM+TTS)

- 1. Initialize Audio unit

```
{
    "request_id": "1",
    "work_id": "audio",
    "action": "setup",
    "object": "audio.setup",
    "data": {
        "capcard": 0,
        "capdevice": 0,
        "capVolume": 0.5,
        "playcard": 0,
        "playdevice": 1,
        "playVolume": 0.5
    }
}
```

- Audio unit initialization response

```
{
    "created": 1692652330,
    "error": {
        "code": 0,
        "message": "audio setup successful"
    },
    "request_id": "1",
    "work_id": "audio.1000"
}
```

- 2. Initialize KWS unit and configure for English recognition model with wake word "HELLO."

```
{
    "request_id": "2",
    "work_id": "kws",
    "action": "setup",
    "object": "kws.setup",
    "data": {
        "model": "sherpa-onnx-kws-zipformer-gigaspeech-3.3M-2024-01-01",
        "response_format": "kws.bool",
        "input": "sys.pcm",
        "enoutput": true,
        "kws": "HELLO"
    }
}
```

- KWS initialization response (Note: the setup process takes approximately 9 seconds)

```
{
    "created": 1692652559,
    "error": {
        "code": 0,
        "message": "kws setup successful"
    },
    "request_id": "2",
    "work_id": "kws.1001"
}
```

- 3. Initialize the ASR unit, configure for English speech recognition model, and set KWS to trigger ASR.

```
{
    "request_id": "3",
    "work_id": "asr",
    "action": "setup",
    "object": "asr.setup",
    "data": {
        "model": "sherpa-ncnn-streaming-zipformer-20M-2023-02-17",
        "response_format": "asr.utf-8",
        "input": "sys.pcm",
        "enoutput": true,
        "enkws": true,
        "rule1": 2.4,
        "rule2": 1.2,
        "rule3": 30
    }
}
```

- ASR initialization response

```
{
    "created": 1692652705,
    "error": {
        "code": 0,
        "message": "asr setup successful"
    },
    "request_id": "3",
    "work_id": "asr.1002"
}
```

- 4. Initialize the LLM unit and configure ASR (speech-to-text) as input data

```
// Input from ASR
{
    "request_id": "4",
    "work_id": "llm",
    "action": "setup",
    "object": "llm.setup",
    "data": {
        "model": "qwen2.5-0.5b",
        "response_format": "llm.utf-8.stream",
        "input": "asr.1002",
        "enoutput": true,
        "enkws": true,
        "max_token_len": 127,
        "prompt": "You are a knowledgeable assistant capable of answering various
questions and providing information."
    }
}
```

- LLM initialization response

```
{
    "created": 1692653061,
    "error": {
        "code": 0,
        "message": "llm setup successful"
    },
    "request_id": "4",
    "work_id": "llm.1003"
}
```

- 5. Initialize the TTS unit, configure for English text-to-speech model, and set LLM inference results as input.

```
// Input from LLM
{
    "request_id": "5",
    "work_id": "tts",
    "action": "setup",
    "object": "tts.setup",
    "data": {
        "model": "single_speaker_english_fast",
        "response_format": "tts.base64.wav",
        "input": "llm.1003",
        "enoutput": true,
        "enkws": true
    }
}
```

- TTS unit initialization response

```
{
    "created": 1692653109,
    "error": {
        "code": 0,
        "message": "tts setup successful"
    },
    "request_id": "5",
    "work_id": "tts.1004"
}
```

- 6. Wake up using the keyword "HELLO," then proceed with voice interaction.