Atlas 300I 推理卡
6.0.0

DSMI API 参考（型号 3000,3010）

文档版本	01
发布日期	2023-02-01

华为技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： https://e.huawei.com

目 录

1 简介

DSMI（Device System Manage Interface）是管理昇腾AI处理器的接口，包括设备管理接口、MAC地址管理接口、风扇管理接口、配置管理接口、软件升级接口、昇腾AI处理器复位启动接口。通过这些接口，用户可以获取昇腾AI处理器的数量、健康状态、温度、传感器、风扇等信息，便于用户监控昇腾AI处理器的状态。

说明

- 昇腾310 AI处理器场景下，由于DSMI session限制，普通用户的并发场景下，同一个device最多支持8个线程或者进程同时调用DSMI接口。
- 主进程和其子进程不能同时调用DSMI接口。
- 调用者需要校验本文档中接口的返回值和出参，确保在合法范围内才能使用。
- DSMI接口从1.0.10版本开始废弃，计划于2023年版本不再提供，请使用对应的DCMI接口进行替代。

2 使用约定

- 本文中接口的所有参数都是必选参数。
- 昇腾310 AI处理器场景下，约定如下：
  - PCIe标卡，具体的产品形态例如Atlas 300。
  - mini模块，具体的产品形态例如Atlas 200、Atlas 200 DK开发者板。
  - 如果PCIe标卡或mini模块作为EP的场景，DSMI接口都运行在Host侧；mini模块作为RC的场景，DSMI接口都运行在mini侧（例如，Atlas 200 DK开发者板）。
- 如果具体的DSMI接口中无特殊说明，则DSMI接口权限在如下场景相同。
  - 物理机中的特权容器root用户等同于物理机root用户。

3 设备管理接口

3.1 dsmi_get_device_count

函数原型

int dsmi_get_device_count(int *device_count)

功能说明

查询加载成功的昇腾AI处理器设备个数。

参数说明

参数名	输入/输出	描述
device_count	输出	查询到的设备个数。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

异常处理

无。

约束说明

无。

调用示例

```
int ret;
int device_count;
ret = dsmi_get_device_count(&device_count);
if(ret != 0) {
    //todo
    return ret;
}
```

3.2 dsmi_get_all_device_count

函数原型

int dsmi_get_all_device_count(int *all_device_count)

功能说明

查询host启动后，与Host的PCIe建链成功的昇腾AI处理器设备个数。

昇腾310 AI处理器场景下，该接口只支持PCIe标卡。

参数说明

参数名	输入/输出	描述
all_device_count	输出	查询到的设备个数。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

无。

调用示例

```
int ret;
int device_count;
ret = dsmi_get_all_device_count(&device_count);
```

```
if(ret != 0) {
    // todo
    return ret;
}
```

3.3 dsmi_list_device

函数原型

int dsmi_list_device(int device_id_list[], int count)

功能说明

列举所有Device的设备号。

参数说明

参数名	输入/输出	描述
device_id_list[]	输出	输出所有Device的设备号列表。
count	输入	设备个数；count通过调用**dsmi_get_device_count**接口获取。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

无。

调用示例

```
int ret;
int device_count;
int device_list[64] = {0};

ret = dsmi_get_device_count(&device_count);
if(ret != 0){
    // todo
    return ERROR;
}

if (device_count == 0) {
    // todo
    return ERROR;
}

ret = dsmi_list_device(&device_list[0], device_count);
```

```
if(ret != 0) {
    // todo
    return ret;
}
```

3.4 dsmi_get_device_health

函数原型

int dsmi_get_device_health(int device_id, unsigned int *phealth)

功能说明

获取设备总体健康状态，如果有多个告警，以最严重的告警作为设备的告警。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。
phealth	输出	设备总体健康状态的指针，只代表本部件，不包括与本部件存在逻辑关系的其它部件。 例如，**phealth**的值以十六进制形式显示时，健康状态值为： • 0：正常 • 1：一般告警 • 2：重要告警 • 3：紧急告警 • 0xffffffff：该设备不存在或者未启动 **说明** 如果有多个告警，以最严重的告警作为设备的告警。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

无

调用示例

```
int ret;
unsigned int health;
ret = dsmi_get_device_health(0, &health);
if (ret != 0) {
    // todo
    return ERROR;
}
```

3.5 dsmi_get_device_errorcode

函数原型

int dsmi_get_device_errorcode(int device_id, int* errorcount, unsigned int *perrorcode)

功能说明

查询设备故障码。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。
errorcount	输出	异常错误码数量，取值范围：0~128。
perrorcode	输出	错误码， perrorcode 长度至少为128 unsigned int类型大小。 详细的错误码描述请参见《黑匣子错误码信息列表》。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

无。

调用示例

```
int ret;
int device_count;
```

```
int device_list[64] = {0};
int errorcount;
unsigned int perrorcode[128] = {0};
unsigned int health;

ret = dsmi_get_device_count(&device_count);
if(ret != 0){
    // todo
    return ERROR;
}

if (device_count == 0) {
    // todo
    return ERROR;
}

ret = dsmi_list_device(device_list, device_count);
if(ret != 0) {
    // todo
    return ret;
}

for (int i = 0; i < device_count; i++) {
    ret = dsmi_get_device_health(device_list[i], &health);
    if (ret != 0) {
        // todo
        return ERROR;
    }

    // 只有非健康下，才会有错误码信息
    if (health != 0) {
        ret = dsmi_get_device_errorcode(device_list[i], &errorcount, perrorcode);
        if(ret != 0) {
           // todo
            return ret;
        }
    }
}
```

3.6 dsmi_query_errorstring

函数原型

int dsmi_query_errorstring(int device_id, unsigned int errorcode, unsigned char *perrorinfo, int buffsize)

功能说明

查询设备故障描述信息。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。 由于查询错误字符描述时，跟设备无关，因此该参数值可以是有效device_id中的一个随意值。
errorcode	输入	要查询的异常错误码。该异常错误码通过**dsmi_get_device_errorcode**接口获取，详细的错误码描述请参见《黑匣子错误码信息列表》。
perrorinfo	输出	对应的错误字符描述。
buffsize	输入	传入perrorinfo的大小，确保长度为48字节。若设置的perrorinfo大小大于48字节，则默认只使用48字节。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

无。

调用示例

```
#define ERROR_CODE_MAX_NUM 128
#define BUFF_SIZE 256

int ret = 0;
int errorcount = 0;
unsigned int health;
unsigned char perrorinfo[BUFF_SIZE] = {0};
unsigned int perrorcode [ERROR_CODE_MAX_NUM] = {0};

ret = dsmi_get_device_health(0, &health);
if (ret != 0) {
    // todo
    return ERROR;
}

// 只有非健康下，才会有错误码信息
if (health != 0) {
    ret = dsmi_get_device_errorcode(0, &errorcount, perrorcode);
    if(ret != 0 || (errorcount == 0)) {
```

```
        // todo
        return ret;
    }

    for (int i = 0; i < errorcount; i++) {
        ret = dsmi_query_errorstring(0, perrorcode[i], perrorinfo, BUFF_SIZE);
        if(ret != 0) {
            // todo
            continue;
        }
    }
}
```

3.7 dsmi_get_chip_info

函数原型

int dsmi_get_chip_info(int device_id, struct dsmi_chip_info_stru *chip_info)

功能说明

获取昇腾AI处理器的相关信息。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。

chip_info	输出	获取昇腾AI处理器的相关信息。
		chip_info结构体定义：
		`#define MAX_CHIP_NAME 32`
		`struct dsmi_chip_info_stru{`
		`  unsigned char chip_type[MAX_CHIP_NAME];`
		`  unsigned char chip_name[MAX_CHIP_NAME];`
		`  unsigned char chip_ver[MAX_CHIP_NAME];`
		`};`

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

- 在虚拟化直通、容器直通场景，返回的信息和物理机场景保持一致。
- 昇腾310 AI处理器场景下，无约束。

调用示例

```
int ret = 0;
struct dsmi_chip_info_stru info = {{0},{0},{0}};
ret = dsmi_get_chip_info(0, &info);
if(ret != 0) {
    // todo
    return ret;
}
```

3.8 dsmi_get_device_die

函数原型

int dsmi_get_device_die(int device_id, struct dsmi_soc_die_stru *pdevice_die)

功能说明

获取指定设备的DIE ID。

参数说明

参数名	输入/输出	描述
device_id	输入	该参数由两部分组成： 高16位表示指定dieid类型号。 0 ： 表示SOC DIE 低16位表示指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。

pdevice_die	输出	返回DIE信息
		die id结构体信息：
		#define DSMI_SOC_DIE_LEN 5
		struct dsmi_soc_die_stru {
		unsigned int soc_die[DSMI_SOC_DIE_LEN]; /**< 5 soc_die array sizet */
		};

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

异常处理

无。

约束说明

无。

调用示例

```
int ret;
struct dsmi_soc_die_stru pdevice_die = {0};

ret = dsmi_get_device_die(0,&pdevice_die);
if(ret != 0) {
    // todo
    return ret;
}
```

3.9 dsmi_get_board_id

函数原型

int dsmi_get_board_id(int device_id, unsigned int *board_id)

功能说明

获取单板的ID。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。
board_id	输出	单板的ID。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
unsigned int board_id = 0;
ret = dsmi_get_board_id (0,&board_id);
if(ret != 0) {
    // todo
    return ret;
}
```

3.10 dsmi_get_board_info

函数原型

int dsmi_get_board_info(int device_id, struct dsmi_board_info_stru *pboard_info)

功能说明

获取单板信息，包括单板的board_id、pcb_id、bom_id、slot_id。

昇腾310 AI处理器场景下，该接口支持PCIe标卡（只支持board_id和slot_id）、mini模块（包含mini模块作为RC或EP的场景）。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。

pboard_info	输出	返回单板信息。
		单板信息结构体如下：
		struct dsmi_board_info_stru {
		unsigned int board_id; // 单板的board_id
		unsigned int pcb_id;
		unsigned int bom_id;
		unsigned int slot_id; // 如果单板上有多个昇腾AI处理器，查询是哪个昇腾AI处理器，0：A平面，1：B平面，2：C平面，3：D平面。
		};

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

无。

调用示例

```
int ret = 0;
struct dsmi_board_info_stru board_info = {0};
ret = dsmi_get_board_info(0,&board_info);
if(ret != 0) {
    // todo
    return ret;
}
```

3.11 dsmi_get_aicore_info

函数原型

int dsmi_get_aicore_info(int device_id, struct dsmi_aicore_info_stru *pdevice_aicore_info)

功能说明

查询aicore的频率信息。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。

pdevice_aicore_info	输出	返回aicore信息。
		aicore信息结构体如下：
		struct dsmi_aicore_info_stru {
		unsigned int freq; // 额定频率，单位是MHZ
		unsigned int curfreq; // 当前频率，单位是MHZ
		};
		说明
		AI core freq（额定频率）：AI core 在TDP功耗和场景下，能够持续运行的频率。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

异常处理

昇腾310 AI处理器场景下，无约束。

约束说明

无。

调用示例

```
unsigned int dev_id = 0;
struct dsmi_aicore_info_stru pdevice_aicore_info = {0};
int ret;

ret = dsmi_get_aicore_info(dev_id, &pdevice_aicore_info);
if(ret != 0) {
    // todo
    return ret;
}
```

3.12 dsmi_get_device_frequency

函数原型

int dsmi_get_device_frequency(int device_id, int device_type, unsigned int *pfrequency)

功能说明

获取昇腾AI处理器的频率。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。

参数名	输入/输出	描述
device_type	输入	设备类型，目前支持如下几种，数值和具体设备类型对应如下。 昇腾310 AI处理器场景下，支持1、2、6、7、9这几种类型。 ● 1：内存 ● 2：控制CPU ● 6：HBM ● 7：AI CORE当前频率 ● 9：AI CORE额定频率 **说明** AI Core额定频率 ： AI Core表示在TDP功耗和场景下，能够持续运行的频率。 HBM：昇腾310 AI处理器场景下查询成功，但实际结果无意义。
pfrequency	输出	频率，单位MHZ。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

昇腾310 AI处理器场景下，无约束。

调用示例

```
int ret = 0;
int device_type = 1;
unsigned int frequency;

ret = dsmi_get_device_frequency(0, device_type, &frequency);
if(ret != 0) {
    // todo
    return ret;
}
```

3.13 dsmi_get_device_temperature

函数原型

int dsmi_get_device_temperature(int device_id, int *ptemperature)

功能说明

查询昇腾AI处理器的温度。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。
ptemperature	输出	昇腾AI处理器的温度：单位摄氏度，精度为1摄氏度，16位带符号类型。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

昇腾310 AI处理器场景下，无约束。

调用示例

```
int ret;
int temp = 0;

ret = dsmi_get_device_temperature(0, &temp);
if(ret != 0) {
    // todo
    return ret;
}
```

3.14 dsmi_get_device_power_info

函数原型

int dsmi_get_device_power_info(int device_id, struct dsmi_power_info_stru * pdevice_power_info)

功能说明

查询设备功耗。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。

pdevice_power_info	输出	设备额定功耗：单位为W，精度为0.1W。16位无符号short类型，小字节序。
		转换为W的计算公式：value = reading * 0.1;
		`struct dsmi_power_info_stru {`
		`  unsigned short power;`
		`};`
		说明
		昇腾310 AI处理器场景下，获取的功耗为额定功耗。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

昇腾310 AI处理器场景下，无约束。

调用示例

```
int ret;
struct dsmi_power_info_stru  powerinfo = {0};

ret = dsmi_get_device_power_info(0, &powerinfo);
if(ret != 0) {
    // todo
    return ret;
}
```

3.15 dsmi_get_device_voltage

函数原型

int dsmi_get_device_voltage(int device_id, unsigned int *pvoltage)

功能说明

查询昇腾AI处理器的电压。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。
pvoltage	输出	昇腾AI处理器的电压：精度为0.01V。转换为V的计算公式：value=pvoltage*0.01， 例如，通过本接口读取到的pvoltage为31时，此时实际电压为0.31V。 **说明** 昇腾310 AI处理器场景下，获取的电压为CPU核电压。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

昇腾310 AI处理器场景下，无约束。

调用示例

```
int ret = 0;
unsigned int voltage;

ret = dsmi_get_device_voltage(0, &voltage);
if(ret != 0) {
    // todo
    return ret;
}
```

3.16 dsmi_get_device_utilization_rate

函数原型

int dsmi_get_device_utilization_rate(int device_id, int device_type, unsigned int *putilization_rate)

功能说明

获取昇腾AI处理器的利用率。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。
device_type	输入	设备类型，目前支持如下几种，数值和具体设备类型对应如下。 昇腾310 AI处理器场景下，支持1、2、3、4、5、6、8这几种类型。 ● 1：内存 ● 2：AI CORE ● 3：AI CPU ● 4：控制CPU ● 5：内存带宽 ● 6：HBM ● 8：DDR **说明** HBM：昇腾310 AI处理器场景下查询成功，但实际结果无意义。 DDR：昇腾310 AI处理器场景下查询成功，但实际结果无意义。
putilization_rate	输出	昇腾AI处理器的利用率，对应device_type单元的利用率，单位：%。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

昇腾310 AI处理器场景下，无约束。

调用示例

```
int ret = 0;
int dev_id = 0;
unsigned int putilization_rate;
ret = dsmi_get_device_utilization_rate(dev_id, 1, &putilization_rate);
if(ret != 0) {
    // todo
    return ret;
}
```

3.17 dsmi_get_device_flash_count

函数原型

int dsmi_get_device_flash_count(int device_id, unsigned int *pflash_count)

功能说明

获取Flash个数。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。
pflash_count	输出	返回Flash个数。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

昇腾310 AI处理器场景下，无约束。

调用示例

```
int ret = 0;
unsigned int flash_count = 0;

ret = dsmi_get_device_flash_count(0, &flash_count);
if(ret != 0) {
    // todo
    return ret;
}
```

3.18 dsmi_get_device_flash_info

函数原型

int dsmi_get_device_flash_info(int device_id, unsigned int flash_index, dm_flash_info_stru *pflash_info)

功能说明

获取flash设备的信息。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。
flash_index	输入	Flash索引号，取值范围为0~**dsmi_get_device_flash_count**决定。
pflash_info	输出	返回Flash设备信息。 flash信息结构体定义 typedef struct dm_flash_info_stru { unsigned long long flash_id; /* combined device & manufacturer code */ unsigned short device_id; /* device id */ unsigned short vendor; /* the primary vendor id */ unsigned int state; /*flash health */ unsigned long long size; /* total size in bytes */ unsigned int sector_count; /* number of erase units */ unsigned short manufacturer_id; /* manufacturer id */ }DM_FLASH_INFO_STRU, dm_flash_info_stru; **说明** 返回flash信息结构体中state的说明： 昇腾310 AI处理器场景下，0x8表示正常，0x10表示非正常。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

昇腾310 AI处理器场景下，无约束。

调用示例

```
int i;
int ret = 0;
dm_flash_info_stru flash_info = {0};
unsigned int flash_count = 0;

ret = dsmi_get_device_flash_count(0, &flash_count);
if(ret != 0) {
    //todo
    return ret;
```

```
}

for (i = 0; i < flash_count; i++){
    ret = dsmi_get_device_flash_info(0, i, &flash_info);
    if(ret != 0) {
        //todo
        return ret;
    }
}
```

3.19 dsmi_get_memory_info

函数原型

int dsmi_get_memory_info(int device_id, struct dsmi_memory_info_stru *pdevice_memory_info)

功能说明

获取系统内存信息。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。

pdevice_memory_info	输出	返回内存信息，内存信息结构体：
		struct dsmi_memory_info_stru{
		unsigned long long memory_size;
		unsigned int freq;//频率，单位MHZ
		unsigned int utiliza;//占用率，单位%
		};
		说明
		返回内存信息结构体中memory_size单位说明：
		昇腾310 AI处理器场景下的单位为MB

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

昇腾310 AI处理器场景下，无约束。

调用示例

```
int ret;
struct dsmi_memory_info_stru device_memory_info = {0};

ret = dsmi_get_memory_info(0, &device_memory_info);
if(ret != 0) {
    //todo
    return ret;
}
```

3.20 dsmi_get_ecc_info

函数原型

int dsmi_get_ecc_info(int device_id, int device_type, struct dsmi_ecc_info_stru *pdevice_ecc_info)

功能说明

获取ECC信息。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。
device_type	输入	设备类型，目前只支持： 昇腾310 AI处理器场景下，只支持DSMI_DEVICE_TYPE_DDR。 typedef enum { DSMI_DEVICE_TYPE_DDR, DSMI_DEVICE_TYPE_SRAM, DSMI_DEVICE_TYPE_HBM, DSMI_DEVICE_TYPE_NPU, DSMI_HBM_RECORDED_SINGLE_ADDR, DSMI_HBM_RECORDED_MULTI_ADDR, DSMI_DEVICE_TYPE_NONE = 0xff } DSMI_DEVICE_TYPE;

pdevice_ecc_info	输出	返回ECC信息,对应结构体如下：
		struct dsmi_ecc_info_stru{
		int enable_flag; // 1:使能 0:禁用
		unsigned int single_bit_error_count;
		unsigned int double_bit_error_count;
		};
		说明
		当前支持单比特和多比特错误查询，如果查询到多比特错误，需要进行相关问题定位和处理。
		该接口获取的是实时的数据，系统重启之后数据清零。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

昇腾310 AI处理器场景下，无约束。

调用示例

```
int ret = 0;
struct dsmi_ecc_info_stru device_ecc_info = {0};

ret = dsmi_get_ecc_info(0, DSMI_DEVICE_TYPE_DDR, &device_ecc_info);
if (ret != 0) {
    // todo
    return ret;
}
```

3.21 dsmi_get_pcie_info

函数原型

int dsmi_get_pcie_info(int device_id, struct tag_pcie_idinfo *pcie_idinfo)

功能说明

查询PCIe设备信息。

昇腾310 AI处理器场景下，该接口只支持PCIe标卡。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。

参数名	输入/输出	描述

pcie_idinfo	输出	返回PCIe设备信息，对应结构如下：
		typedef struct tag_pcie_idinfo{
		unsigned int deviceid;//设备ID
		unsigned int venderid;//厂商ID
		unsigned int subvenderid;//子厂商ID
		unsigned int subdeviceid;//子设备ID
		unsigned int bdf_deviceid;//BDF（Bus，Device，Function）中的设备ID
		unsigned int bdf_busid;//BDF（Bus，Device，Function）中的总线ID
		unsigned int bdf_funcid;//BDF（Bus，Device，Function）中的功能ID
		}TAG_PCIE_IDINFO, tag_pcie_idinfo;

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

无。

调用示例

```
int ret;
struct tag_pcie_idinfo pcie_idinfo = {0};

ret = dsmi_get_pcie_info(0, &pcie_idinfo);
if(ret != 0) {
    // todo
    return ret;
}
...
```

3.22 dsmi_get_soc_sensor_info

函数原型

int dsmi_get_soc_sensor_info (int device_id, int sensor_id,TAG_SENSOR_INFO *tsensor_info)

功能说明

获取SOC传感器信息。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。
sensor_id	输入	指定传感器索引，具体如下值： • 0：CLUSTER_TEMP_ID， 表示CLUSTER温度；返回值对应输出联合体中的uchar成员； • 1：PERI_TEMP_ID，表示PERI温度；返回值对应输出联合体中的uchar成员； • 2：AICORE0_TEMP_ID, 表示AICORE0温度；返回值对应输出联合体中的uchar成员； • 3：AICORE1_TEMP_ID，表示AICORE1温度；返回值对应输出联合体中的uchar成员； • 4：AICORE_LIMIT_ID，AICORE限核状态返回结果是0，不限核返回结果是 1；返回值对应输出联合体中的uchar成员； • 5：AICORE_TOTAL_PER_ID，表示AICORE脉冲总周期；返回值对应输出联合体中的uchar成员； • 6：AICORE_ELIM_PER_ID，表说aicore可消除周期；返回值对应输出联合体中的uchar成员； • 7：AICORE_BASE_FREQ_ID，表示aicore基准频率 MHZ；返回值对应输出联合体中的ushort成员； • 8：NPU_DDR_FREQ_ID，表示DDR频率单位MHZ；返回值对应输出联合体中的ushort成员； • 9：THERMAL_THRESHOLD_ID，返回值对应输出联合体中的temp[2]成员；temp[0]为温保限频温度，temp[1]为系统复位温度； • 10：NTC_TEMP_ID，返回值对应输出联合体中的ntc_tmp[4]成员；ntc_tmp[0] ntc_tmp[1] ntc_tmp[2] ntc_tmp[3]分别对应四个热敏电阻温度。昇腾310 AI处理器场景下，board_id支持000、1000、2000、004、1004、2004，如果board_id不在该范围内，查询热敏电阻温度时，接口返回报错。您可以先调用**dsmi_get_board_info**接口查询board_id； • 11：SOC_TEMP_ID，表示SOC最高温；返回值对应输出联合体中的uchar成员； 昇腾310 AI处理器场景下，支持0~11；

参数名	输入/输出	描述
tsensor_info	输出	类型定义如下： #define DSMI_TAG_SENSOR_TEMP_LEN 2 #define DSMI_TAG_SENSOR_NTC_TEMP_LEN 4 #define SENSOR_DATA_MAX_LEN 16 typedef union tag_sensor_info { unsigned char uchar; unsigned short ushort; unsigned int uint; signed int iint; signed char temp[DSMI_TAG_SENSOR_TEMP_LEN]; /**< 2 temp size */ signed int ntc_tmp[DSMI_TAG_SENSOR_NTC_TEMP_LEN]; /**< 4 ntc_tmp size */ unsigned int data[SENSOR_DATA_MAX_LEN]; } TAG_SENSOR_INFO;

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

昇腾310 AI处理器场景下，无约束。

调用示例

```
int ret;
TAG_SENSOR_INFO sensor_info = {0};

ret = dsmi_get_soc_sensor_info(0, 11, &sensor_info);
if(ret != 0) {
    // todo
    return ret;
}
...
```

3.23 dsmi_get_mini2mcu_heartbeat_status

函数原型

int dsmi_get_mini2mcu_heartbeat_status(int device_id, unsigned char *status, unsigned int *disconn_cnt)

功能说明

获取Device对MCU的心跳状态和计数，如果获取了多个Device的状态为connect情况下，disconn_cnt值越小代表链路越稳定。同一时刻一块PCIe标卡只有1个Device与MCU之间是连通的。

昇腾310 AI处理器场景下，该接口只支持PCIe标卡。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。
status	输出	心跳状态： 0：disconnect 1：connect
disconn_cnt	输出	心跳失联次数： 范围：0~9999

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码**。

约束说明

调用该接口的程序必须在物理机下运行。

昇腾310 AI处理器PCIe标卡场景下，如果返回不支持错误码，表示对应Device与MCU物理上不连通。

调用示例

```
int ret;
unsigned char tmp_status = 0;
unsigned int tmp_disconn_cnt = 0;

ret =dsmi_get_mini2mcu_heartbeat_status(0, &tmp_status, &tmp_disconn_cnt);
if(ret != 0) {
    // todo
    return ret;
}
...
```

3.24 dsmi_dft_get_elable

函数原型

int dsmi_dft_get_elable(int device_id, int item_type, char *elabel_data, int *len)

功能说明

获取指定设备的elable信息。

昇腾310 AI处理器场景下，该接口只支持miniRC场景。

参数说明

参数名	输入/输出	描述
device_id	输入	该参数由两部分组成： 高16位表示字符设备eeprom号，范围为0~实际eeprom个数。 低16位表示指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。

参数名	输入/输出	描述
item_type	输入	查询指定标签项。 装备标签项： • 0x10: Chassis Type • 0x11: Chassis Part Number • 0x12: Chassis Serial Number • 0x20: Mfg. Date / Time • 0x21: Board Manufacturer • 0x22: Board Product Name • 0x23: Board Serial Number • 0x24: Board Part Number • 0x25: FRU File ID • 0x30: Product Manufacturer Name • 0x31: Product Name • 0x32: Product Part/Model Number • 0x33: Product Version • 0x34: Product Serial Number • 0x35: Asset Tag • 0x36: FRU File ID • 0x50: 预留 • 0x60: System Manufacturer Name • 0x61: System Product Name • 0x62: System Version • 0x63: System Serial Number
elabel_data	输出	输出标签数据字符串。
len	输出	输出数据长度。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

调用该接口的程序必须在物理机下运行。

调用示例

```
int ret = 0;
int test_item = 0x10;
char elable_data_read[256] = {0};
int len = sizeof(elable_data_read);

ret = dsmi_dft_get_elable(0, test_item,elable_data_read,&len);
if(ret != 0) {
    // todo
    return ret;
}
...
```

3.25 dsmi_enable_container_service

函数原型

int dsmi_enable_container_service(void)

功能说明

初始化当前容器所有设备，用于后续逻辑id与物理id之间转换。

参数说明

无。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

无

调用示例

```
...
int ret = 0;
ret = dsmi_enable_container_service();
if(ret != 0) {
    // todo
    return ret;
}
...
```


3.26 dsmi_get_phyid_from_logicid

函数原型

int dsmi_get_phyid_from_logicid(unsigned int logicid, unsigned int *phyid)

功能说明

通过昇腾AI处理器逻辑ID获取昇腾AI处理器物理ID。

参数说明

参数名	输入/输出	描述
logicid	输入	昇腾AI处理器的逻辑ID。 用户调用**3.1 dsmi_get_device_count**接口获取可用的Device数量后，这个logicid的取值范围：[0, (device_count-1)]
phyid	输出	昇腾AI处理器的物理ID。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

无。

调用示例

```
...
int ret = 0;
unsigned int  logicid = 0;
unsigned int phyid=0;
ret = dsmi_get_phyid_from_logicid (logicid, &phyid);
if(ret != 0) {
    // todo
    return ret;
}
...
```

3.27 dsmi_get_logicid_from_phyid

函数原型

int dsmi_get_logicid_from_phyid(unsigned int phyid, unsigned int *logicid)

功能说明

通过昇腾AI处理器物理ID获取昇腾AI处理器逻辑ID。

参数说明

参数名	输入/输出	描述
phyid	输入	昇腾AI处理器的物理ID。 **说明** 可执行 ls /dev/davinci* 命令获取设备的物理ID，如显示/dev/davinci0，表示设备的物理ID为0。
logicid	输出	昇腾AI处理器的逻辑ID。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码**。

约束说明

昇腾310 AI处理器容器场景不支持该接口，在该场景下使用该接口获取的结果不保证正确性。

调用示例

```
...
int ret;
unsigned int  phyid = 0;
unsigned int logicid =0;

ret = dsmi_get_logicid_from_phyid(phyid,&logicid);
if(ret != 0) {
    //todo
    return ret;
}
...
```

3.28 dsmi_passthru_mcu

函数原型

int dsmi_passthru_mcu(int device_id, struct passthru_message_stru *passthru_message)

功能说明

消息转发接口，HOST消息通过Device转发MCU。对于需要通过Device获取MCU信息的功能，在HOST构建消息，把消息发送到Device，Device再转发到MCU。

昇腾310 AI处理器场景下，该接口只支持PCIe标卡。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号。 有效值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。 昇腾310 AI处理器场景下，一个PCIe标卡有4个Device，但在同一时刻只有1个Device与MCU是连通的，为确保Host消息能通过Device正常转发MCU，您需要先调用**dsmi_get_mini2mcu_heartbeat_status**接口获取哪些Device与MCU之间是连通的（状态为connect），再将处于有连通状态的Device的ID作为入参传入dsmi_passthru_mcu接口。

passthru_message	输入	消息内容，消息结构如下：
		struct passthru_message_stru{
		unsigned int src_len; /*最长传输32个字节*/
		unsigned int rw_flag; /*0 read ,1 write*/
		struct dmp_message_stru src_message;
		struct dmp_message_stru dest_message;
		};

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码**。

约束说明

无。

调用示例

```
int ret;
struct passthru_message_stru assthru_message = {0};
assthru_message.rw_flag = 1;
assthru_message.src_len = 13;
assthru_message.src_message.data.req.opcode = 0x08;
...
ret = dsmi_passthru_mcu(0,&assthru_message);
if(ret != 0) {
    //todo
    return ret;
}
...
```

3.29 dsmi_get_device_cgroup_info

函数原型

int dsmi_get_device_cgroup_info(int device_id, struct tag_cgroup_info *cg_info)

功能说明

获取cgroup内存信息，包括cgroup最大内存数、历史使用最大内存数、当前使用内存数。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。

cg_info	输出	cgroup信息数据结构：
		struct tag_cgroup_info {
		unsigned long long limit_in_bytes; /**< maximum number of used memory */
		unsigned long long max_usage_in_bytes; /**< maximum memory used in history */
		unsigned long long usage_in_bytes; /**< current memory usage */
		};
		说明
		SMP系统下获取到的数据是整个OS的cgroup内存信息，而不是单个设备的cgroup内存信息。

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

约束说明

无。

调用示例

```
...
int ret = 0;
struct tag_cgroup_info cgp_info = {0};
ret = dsmi_get_device_cgroup_info(0, &cgp_info);
if(ret != 0) {
//todo：记录日志
return ret;
}
...
```

3.30 dsmi_get_pcie_bdf

函数原型

int dsmi_get_pcie_bdf(int device_id, struct tag_pcie_bdfinfo *pcie_idinfo)

功能说明

查询PCIe设备信息。

昇腾310 AI处理器场景下，支持PCIe标卡、mini模块（仅支持mini模块作为EP的场景）。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，有效值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。

pcie_idinfo	输出	PCIe设备信息。
		typedef struct tag_pcie_bdfinfo{
		unsigned int bdf_deviceid;//BDF（Bus，Device，Function）中的设备ID
		unsigned int bdf_busid;//BDF（Bus，Device，Function）中的总线ID
		unsigned int bdf_funcid;//BDF（Bus，Device，Function）中的功能ID
		}TAG_PCIE_BDFINFO, tag_pcie_bdfinfo;

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码**。

异常处理

无。

约束说明

无。

调用示例

```
int ret = 0;
struct tag_pcie_bdfinfo pcie_bdf = {0};
ret = dsmi_get_pcie_bdf(0, &pcie_bdf);
if(ret != 0) {
//todo：记录日志
return ret;
}
...
```

3.31 dsmi_get_computing_power_info

函数原型

int dsmi_get_computing_power_info(int device_id, int computing_power_type, struct dsmi_computing_power_info *computing_power_info)

功能说明

查询算力信息。当前仅支持查询aicore核数。

参数说明

参数名	输入/输出	描述
device_id	输入	设备ID，取值范围：0~7。
computing_power_type	输入	获取算力信息的类型。 1：返回aicore核数。

computing_power_info	输出	#define COMPUTING_POWER_INFO_RESERVE_NUM 3
		struct dsmi_computing_power_info {
		unsigned int data1;
		unsigned int reserve[COMPUTING_POWER_INFO_RESERVE_NUM];
		};

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

约束说明

无。

调用示例

```
...
int ret;
struct dsmi_computing_power_info computing_power_info;
ret = dsmi_get_computing_power_info (dev_id, type, &computing_power_info);
if (ret) {
    //todo : 记录日志
    return ERROR;
}
```

3.32 dsmi_get_device_alarminfo

函数原型

int dsmi_get_device_alarminfo(int device_id, int* alarmcount, struct dsmi_alarm_info_stru *palarminfo)

功能说明

查询设备故障告警信息。

昇腾310 AI处理器场景下，支持PCIe标卡、mini模块（仅支持mini模块作为EP的场景）。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
alarmcount	输出	告警数量，取值范围：0~128。
palarminfo	输出	告警信息。palarminfo 长度至少为128 *sizeof (struct dsmi_alarm_info_stru) 字节。详细的告警信息请参见《Ascend 310故障告警处理手册》。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码**。

异常处理

无。

约束说明

无。

调用示例

```
#define ALARM_INFO_MAX_NUM              (128)
...
int ret = 0;
int alarmcount = 0;
struct dsmi_alarm_info_stru *palarminfo = NULL;

palarminfo = (struct dsmi_alarm_info_stru *)malloc(sizeof(struct dsmi_alarm_info_stru) *
ALARM_INFO_MAX_NUM);
if (!palarminfo) {
    //todo:记录日志
    return –ENOMEM;
}
ret = dsmi_get_device_alarminfo(0, &alarmcount, palarminfo);
if(ret != 0) {
//todo:记录日志
return ret;
}

free(palarminfo);
...
```

3.33 dsmi_get_driver_health

函数原型

int dsmi_get_driver_health(unsigned int *phealth)

功能说明

驱动健康状态。

昇腾310 AI处理器场景下，该接口支持PCIe标卡、mini模块（包含mini模块作为RC或EP的场景）。

参数说明

参数名	输入/输出	描述
phealth	输出	驱动健康状态的指针。 例如，**phealth**的值以十六进制形式显示时，健康状态值为： ● 0：正常 ● 1：一般告警 ● 2：重要告警 ● 3：紧急告警 ● ffffffff：该设备不存在或者未启动

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

约束说明

无。

调用示例

```
...
int ret = 0;
unsigned int  health;
ret = dsmi_get_driver_health(&health);
...
```

3.34 dsmi_get_driver_errorcode

函数原型

int dsmi_get_driver_errorcode(int *errorcount, unsigned int *perrorcode)

功能说明

查询驱动故障码。

昇腾310 AI处理器场景下，该接口支持PCIe标卡、mini模块（包含mini模块作为RC或EP的场景）。

参数说明

参数名	输入/输出	描述
errorcount	输出	错误码数量，取值范围：0~128。
perrorcode	输出	错误码。perrorcode长度至少为128* sizeof(unsigned int) 字节。 详细的错误码描述请参见《黑匣子错误码信息列表》。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码**。

约束说明

无。

调用示例

```
#define ERROR_CODE_MAX_NUM              (128)
...
int ret = 0;
int errorcount = 0;
unsigned int perrorcode[ERROR_CODE_MAX_NUM] = {0};
ret = dsmi_get_driver_errorcode(&errorcount, perrorcode);
if(ret != 0) {
//todo:记录日志
return ret;
}
...
```

3.35 dsmi_set_device_info

3.35.1 接口原型

函数原型

int dsmi_set_device_info(unsigned int device_id, DSMI_MAIN_CMD main_cmd, unsigned int sub_cmd, const void *buf, unsigned int buf_size)

功能说明

设置device的信息的通用接口，对各模块信息进行配置。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	unsigned int	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。
main_cmd	输入	DSMI_MAIN_CMD	模块cmd信息，执行用于获取对应模块的信息。 typedef enum { DSMI_MAIN_CMD_DVPP = 0, DSMI_MAIN_CMD_ISP, DSMI_MAIN_CMD_TS_GROUP_NUM, DSMI_MAIN_CMD_CAN, DSMI_MAIN_CMD_UPGRADE = 5, DSMI_MAIN_CMD_UFS, DSMI_MAIN_CMD_OS_POWER, DSMI_MAIN_CMD_LP, DSMI_MAIN_CMD_MEMORY, DSMI_MAIN_CMD_RECOVERY, DSMI_MAIN_CMD_TS, DSMI_MAIN_CMD_CHIP_INF, DSMI_MAIN_CMD_QOS, DSMI_MAIN_CMD_TEMP = 50, DSMI_MAIN_CMD_SVM, DSMI_MAIN_CMD_VDEV_MNG, DSMI_MAIN_CMD_MAX, } DSMI_MAIN_CMD; **说明** 设置场景目前支持DSMI_MAIN_CMD_CAN、DSMI_MAIN_CMD_UFS、DSMI_MAIN_CMD_RECOVERY、DSMI_MAIN_CMD_TEMP、DSMI_MAIN_CMD_SVM命令字。
sub_cmd	输入	unsigned int	详细见本节后章节功能说明。
buf	输入	void *	用于配置相应设备的配置信息。
size	输入	unsigned int	buf数组的长度。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

异常处理

参考本节后各主命令章节功能说明。

调用示例

```
int ret;
int dev_id = 0;
int buf = 0;
int size = sizeof(int);
int uart_index = 0; // 根据环境上具体的串口，使用合理的值。

ret = dsmi_set_device_info(dev_id, DSMI_MAIN_CMD_UART,
DSMI_UART_SUB_CMD_MAKE(uart_index, DSMI_UART_SUB_CMD_SPEED), &buf, size);
if (ret){
    // todo
    return ret;
}
```

3.35.2 DSMI_MAIN_CMD_EX_CONTAINER 命令说明

函数原型

int dsmi_set_device_info(unsigned int device_id, DSMI_MAIN_CMD main_cmd, unsigned int sub_cmd, const void *buf, unsigned int buf_size)

功能说明

设置device的信息的通用接口，对各模块信息进行配置。通过main_cmd及sub_cmd区分不同配置项。

昇腾310 AI处理器场景下，该接口仅支持mini模块作为RC的场景。

不同的场景下配置项不同，详细配置内容请参考**命令字说明**。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号。 有效值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
main_cmd	输入	指定配置项对应主命令字。
sub_cmd	输入	指定配置项对应子命令字。
buf	输入	指定device配置信息，详细配置内容请参考**命令字说明**。
size	输入	指定device配置有效字节数。

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

返回值-2或-22表示该接口不支持输入的命令字，其它返回值意义见**11 返回码**。

命令字说明

主命令字（main_cmd）	子命令字（sub_cmd）	功能描述	参数说明	使用场景
DSMI_MAIN_CMD_EX_CONTAINER	DSMI_EX_CONTAINER_SUB_CMD_SHARE	支持指定device映射到多容器。	• buf：device映射到多容器的取值。取值为int类型。 – 0x1：使能device映射到多容器。 – 0x0：关闭device映射到多容器。 • size：device映射到多容器的数据结构大小固定为4字节。	mini模块的RC场景

约束说明

配置容器共享设备能力时sub_cmd，buf和buf_size之间必须要满足以下关系，如果不满足会导致接口调用失败。

sub_cmd	buf对应的数据类型	buf_size
DSMI_EX_CONTAINER_SUB_CMD_SHARE	int	sizeof(int)

调用该接口的程序必须在物理机的root用户下运行。

调用示例

```
int ret = 0;
int device_id = 0;
int dev_share = 1;//开启共享
unsigned int size;
ret=dsmi_set_device_info(device_id, DSMI_MAIN_CMD_EX_CONTAINER,
```

```
DSMI_EX_CONTAINER_SUB_CMD_SHARE, &dev_share, size);
if(ret != 0){
//todo:记录日志
return ret;
}
...
```

3.35.3 DSMI_MAIN_CMD_GPIO 命令说明

函数原型

int dsmi_set_device_info(unsigned int device_id, DSMI_MAIN_CMD main_cmd, unsigned int sub_cmd, const void *buf, unsigned int buf_size)

功能说明

获取device的信息的通用接口，获取各模块中的状态信息。通过main_cmd及sub_cmd区分不同配置项。

mini模块场景下，支持该接口，详细配置内容请参考命令字说明。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号。 当前实际支持的设备号，通过dsmi_list_device接口获取。
main_cmd	输入	指定配置项对应主命令字。
sub_cmd	输入	指定配置项对应子命令字。

buf	输入	获取device配置信息，详细配置内容请参考命令字说明。struct devdrv_gpio_ctrl{
		unsigned int gpio_cmd;
		unsigned int gpio;
		int value;
		};
		gpio当前可配置引脚为：444、500、504
		说明
		504引脚在200 RC形态下被休眠唤醒功能占用，可通过配置dts停止休眠唤醒功能。具体可参见《**Atlas 200 AI加速模块软件安装与维护指南（RC场景）**》。

参数名	输入/输出	描述
size	输入	获取device配置有效字节数。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

命令字说明

主命令字（main_cmd）	子命令字（sub_cmd）	功能描述	参数说明	使用场景
DSMI_MAIN_CMD_GPIO	DSMI_GPIO_SUB_CMD_SET_VALUE	支持设置指定gpio寄存器值。	• buf：struct devdrv_gpio_ctrl结构体。参数说明如下： – gpio_cmd：DSMI_GPIO_SUB_CMD_SET_VALUE – gpio：gpio管脚号 – value：设置值 • size：数据结构大小固定为sizeof(struct devdrv_gpio_ctrl)。	仅支持mini模块（包含mini模块作为RC或EP的场景）。

主命令字（main_cmd）	子命令字（sub_cmd）	功能描述	参数说明	使用场景
DSMI_MAIN_CMD_GPIO	DSMI_GPIO_SUB_CMD_DIRECT_OUTPUT	支持设置指定gpio为输出模式并设置寄存器值。	• buf：struct devdrv_gpio_ctrl结构体。参数说明如下： – gpio_cmd：DSMI_GPIO_SUB_CMD_DIRECT_OUTPUT – gpio：gpio管脚号 – value：设置值 • size：数据结构大小固定为sizeof(struct devdrv_gpio_ctrl)。	仅支持mini模块（包含mini模块作为RC或EP的场景）。

主命令字（main_cmd）	子命令字（sub_cmd）	功能描述	参数说明	使用场景
DSMI_MAIN_CMD_GPIO	DSMI_GPIO_SUB_CMD_DIRECT_INPUT	支持设置指定gpio为输入模式。	• buf：struct devdrv_gpio_ctrl结构体。参数说明如下： – gpio_cmd：DSMI_GPIO_SUB_CMD_DIRECT_INPUT – gpio：gpio管脚号 – value：任意值 • size：数据结构大小固定为sizeof(struct devdrv_gpio_ctrl)。	仅支持mini模块（包含mini模块作为RC或EP的场景）。

约束说明

调用该接口的程序必须在物理机的root用户下运行，若在物理机的非root用户，或在容器下运行，则会返回权限错误。

调用示例

```
int ret = 0;
int device_id = 0;
unsigned int size = sizeof(struct devdrv_gpio_ctrl);
struct devdrv_gpio_ctrl gpio_ctrl;
gpio_ctrl.gpio = 500;
gpio_ctrl.gpio_cmd = (unsigned int)DSMI_GPIO_SUB_CMD_SET_VALUE;
gpio_ctrl.value = 1;
ret=dsmi_set_device_info(device_id, DSMI_MAIN_CMD_GPIO, DSMI_GPIO_SUB_CMD_SET_VALUE, &
gpio_ctrl, size);
if(ret != 0){
    //todo:记录日志
    return ret;
}
```

3.36 dsmi_get_device_info

3.36.1 接口原型

函数原型

int dsmi_get_device_info(unsigned int device_id, DSMI_MAIN_CMD main_cmd, unsigned int sub_cmd, void *buf, unsigned int *size)

功能说明

获取device的信息的通用接口，获取各模块中的状态信息。

参数说明

参数名	输入/输出	类型	描述
device_id	输入	unsigned int	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。
main_cmd	输入	DSMI_MAIN_CMD	模块cmd信息，执行用于获取对应模块的信息。 typedef enum { DSMI_MAIN_CMD_DVPP = 0, DSMI_MAIN_CMD_ISP, DSMI_MAIN_CMD_TS_GROUP_NUM, DSMI_MAIN_CMD_CAN, DSMI_MAIN_CMD_UART, DSMI_MAIN_CMD_UPGRADE, DSMI_MAIN_CMD_UFS, DSMI_MAIN_CMD_OS_POWER, DSMI_MAIN_CMD_LP, DSMI_MAIN_CMD_MEMORY, DSMI_MAIN_CMD_RECOVERY, DSMI_MAIN_CMD_TEMP = 50, DSMI_MAIN_CMD_MAX, } DSMI_MAIN_CMD;
sub_cmd	输入	unsigned int	详细见本节后章节功能说明。
buf	输出	void *	用于接收设备信息的的返回值。
size	输入/输出	unsigned int *	buf数组的输入/输出长度。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码**。

异常处理

无。

约束说明

无。

调用示例

```
...
int ret;
int dev_id;
int type = 0;
int status = 0;
int cameraIndex;
int length = sizeof(int);
dev_id = 0;
cameraIndex = 0;
type = 0;
ret = dsmi_get_device_info(dev_id, DSMI_MAIN_CMD_ISP,
DSMI_ISP_SUB_CMD_MAKE(cameraIndex, type), &status, &length);
if(ret != 0) {
    // todo
}
// todo
...
```

3.36.2 DSMI_MAIN_CMD_EX_CONTAINER 命令说明

函数原型

int dsmi_get_device_info(unsigned int device_id, DSMI_MAIN_CMD main_cmd, unsigned int sub_cmd, void *buf, unsigned int *size)

功能说明

获取device的信息的通用接口，获取各模块中的状态信息。通过main_cmd及sub_cmd区分不同配置项。

昇腾310 AI处理器场景下，该接口仅支持mini模块作为RC的场景。

不同的场景下配置项不同，详细配置内容请参考**命令字说明**。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号。 有效值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。
main_cmd	输入	指定配置项对应主命令字。
sub_cmd	输入	指定配置项对应子命令字。
buf	输出	获取device配置信息，详细配置内容请参考**命令字说明**。
size	输出	获取device配置有效字节数。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码**。

返回值-2或-22表示该接口不支持输入的命令字，其它返回值意义见**11 返回码**。

命令字说明

主命令字（main_cmd）	子命令字（sub_cmd）	功能描述	参数说明	使用场景
DSMI_MAIN_CMD_EX_CONTAINER	DSMI_EX_CONTAINER_SUB_CMD_SHARE	支持指定device映射到多容器。	• buf：device映射到多容器的取值。取值为int类型。 – 0x1：使能device映射到多容器。 – 0x0：关闭device映射到多容器。 • size：device映射到多容器的数据结构大小固定为4字节。	mini模块的RC场景

约束说明

查询容器共享设备能力时sub_cmd，buf和buf_size之间必须要满足以下关系，如果不满足会导致接口调用失败。

sub_cmd	buf对应的数据类型	buf_size
DSMI_EX_CONTAINER_SUB_CMD_SHARE	int	sizeof(int)

调用示例

```
int ret = 0;
int device_id = 0;
int dev_share;
unsigned int size;
ret=dsmi_get_device_info(device_id, DSMI_MAIN_CMD_EX_CONTAINER,
DSMI_EX_CONTAINER_SUB_CMD_SHARE, &dev_share, &size);
if(ret != 0){
//todo:记录日志
return ret;
}
...
```

3.36.3 DSMI_MAIN_CMD_GPIO 命令说明

函数原型

int dsmi_get_device_info(unsigned int device_id, DSMI_MAIN_CMD main_cmd, unsigned int sub_cmd, void *buf, unsigned int *size)

功能说明

获取device的信息的通用接口，获取各模块中的状态信息。通过main_cmd及sub_cmd区分不同配置项。

mini模块场景下，支持该接口，详细配置内容请参考命令字说明。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号。 当前实际支持的设备号，通过**dsmi_list_device**接口获取。
main_cmd	输入	指定配置项对应主命令字。
sub_cmd	输入	指定配置项对应子命令字。

buf	输出	获取device配置信息，详细配置内容请参考命令字说明。
		struct devdrv_gpio_ctrl{
		unsigned int gpio_cmd;
		unsigned int gpio;
		int value;
		};
		gpio当前可配置引脚为：444、500、504
		说明
		504引脚在200 RC形态下被休眠唤醒功能占用，可通过配置dts停止休眠唤醒功能。具体可参见《**Atlas 200 AI加速模块软件安装与维护指南（RC场景）**》。

size	输出	获取device配置有效字节数。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

命令字说明

主命令字（main_cmd）	子命令字（sub_cmd）	功能描述	参数说明	使用场景
DSMI_MAIN_CMD_GPIO	DSMI_GPIO_SUB_CMD_GET_VALUE	支持获取指定gpio值	• buf：struct devdrv_gpio_ctrl 结构体： 参数说明如下： – gpio_cmd：DSMI_GPIO_SUB_CMD_GET_VALUE – gpio：gpio管脚号 – value：输出结果 • size：数据结构大小固定为sizeof(struct devdrv_gpio_ctrl)。	仅支持mini模块（包含mini模块作为RC或EP的场景）。

约束说明

无。

调用示例

```
int ret = 0;
int device_id = 0;
unsigned int size;
struct devdrv_gpio_ctrl gpio_ctrl;
ret=dsmi_get_device_info(device_id, DSMI_MAIN_CMD_GPIO, DSMI_GPIO_SUB_CMD_GET_VALUE, &
gpio_ctrl, &size);
if(ret != 0){
    //todo:记录日志
    return ret;
}
```

3.37 dsmi_get_resource_info

函数原型

int dsmi_get_resource_info(unsigned int devid, struct dsmi_resource_para *para, struct dsmi_resource_info *info)

功能说明

查询资源信息。

参数说明

参数名	输入/输出	描述
devid	输入	指定设备号，取值范围与查询的资源类型有关。 • 当查询资源类型为DSMI_DEV_RESOURCE时，取值范围由**3.1 dsmi_get_device_count**决定。当前实际支持的设备号，通过**3.3 dsmi_list_device**接口获取。 • 当查询资源类型为DSMI_VDEV_RESOURCE时，取值范围请通过ls /dev查看vdavinci设备的编号。

参数名	输入/输出	描述

para	输入	结构体为
		#define DSMI_RESOURCR_PARA_RESERVE_MAX 8
		struct dsmi_resource_para {
		unsigned int owner_type;
		unsigned int owner_id;
		unsigned int resource_type;
		unsigned int tsid;
		unsigned int reserve[DSMI_RESOURCR_PARA_RESERVE_MAX]; /**< reserve */
		};
		owner_type类型包括如下：
		enum dsmi_owner_type {
		DSMI_DEV_RESOURCE = 0,
		DSMI_VDEV_RESOURCE,
		DSMI_PROCESS_RESOURCE,
		DSMI_MAX_OWNER_TYPE,
		};
		昇腾310 AI处理器场景下，EP形态owner_type仅支持DSMI_PROCESS_RESOURCE，而RC形态都不支持。
		当前支持的resource_type取值包括如下：
		enum dsmi_dev_resource_type {
		DSMI_DEV_STREAM_ID = 0,
		DSMI_DEV_EVENT_ID,
		DSMI_DEV_NOTIFY_ID,
		DSMI_DEV_MODEL_ID,
		DSMI_DEV_HBM_TOTAL,
		DSMI_DEV_HBM_FREE,
		DSMI_DEV_DDR_TOTAL, // 用户可用总内存
		DSMI_DEV_DDR_FREE, // 用户当前可用内存
		DSMI_DEV_PROCESS_PID, // 查询进程的pid列表
		DSMI_DEV_PROCESS_MEM, // 根据进程的pid查对应内存
		DSMI_DEV_INFO_TYPE_MAX,
		};
		当resource_type为DSMI_DEV_STREAM_ID、DSMI_DEV_EVENT_ID、DSMI_DEV_NOTIFY_ID、DSMI_DEV_MODEL_ID时，需要传入tsid，查询其它resource_type不需要tsid。
		当resource_type为DSMI_DEV_DDR_TOTAL、DSMI_DEV_DDR_FREE时，查询的数据是整个OS的内存信息，而不是单个设备的内存信息。

参数名	输入/输出	描述

info	输入/输出	结构体定义为
		#define DSMI_RESOURCE_INFO_RESERVE_MAX 8
		struct dsmi_resource_info {
		unsigned int buf_len;
		void *buf;
		unsigned int reserve[DSMI_RESOURCE_INFO_RESERVE_MAX]; /* reserve */
		};

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

异常处理

无。

约束说明

当前仅支持DSMI_DEV_RESOURCE，DSMI_VDEV_RESOURCE，DSMI_PROCESS_RESOURCE，其它owner_type返回不支持。

- 当入参为DSMI_DEV_RESOURCE时，无约束。
- 当入参为DSMI_VDEV_RESOURCE时，调用该接口的程序必须在物理机的root用户下运行。
- 当入参为DSMI_PROCESS_RESOURCE时，对应的resource_type需为DSMI_DEV_PROCESS_PID或DSMI_DEV_PROCESS_MEM。查询进程内存前需先查询进程的pid，用户传入查询的个数，通过buf_len承载，由于资源限制，可查询最多为32个，即buf_len为128（32*sizeof(int)）；根据进程的pid查询对应的内存大小，用户输入的pid通过owner_id承载。

获取设备资源信息时，resource_type、buf和buf_len之间必须要满足以下关系，如果不满足会导致接口调用失败或返回数据错误。

resource_type	buf对应的数据类型	buf_len对应的大小	单位
DSMI_DEV_STREAM_ID DSMI_DEV_EVENT_ID DSMI_DEV_NOTIFY_ID DSMI_DEV_MODEL_ID	unsigned int	sizeof(unsigned int)	个 表示id的总数
DSMI_DEV_HBM_TOTAL DSMI_DEV_HBM_FREE	unsigned long long	sizeof(unsigned long long)	Byte
DSMI_DEV_DDR_TOTAL DSMI_DEV_DDR_FREE	unsigned long long	sizeof(unsigned long long)	Byte
DSMI_DEV_PROCESS_PID	int	sizeof(buf)	-
DSMI_DEV_PROCESS_MEM	unsigned long	sizeof(unsigned log)	Byte

调用示例

```
struct dsmi_resource_para para = {0};
struct dsmi_resource_info info = {0};
unsigned long long data;
int ret;

para.owner_type = DSMI_DEV_RESOURCE;
para.resource_type = DSMI_DEV_STREAM_ID;
para.tsid = 0;
info.buf = &data;
info.buf_len = sizeof(data);

ret = dsmi_get_resource_info(0, &para, &info);
if (ret) {
    // todo, 表示查询失败，打印返回值信息进行定位
    // return ret;
}
// todo
```

3.38 dsmi_read_fault_event

函数原型

int dsmi_read_fault_event(int device_id, int timeout, struct dsmi_event_filter filter, struct dsmi_event *event)

功能说明

查询设备故障或恢复事件的接口。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号：取值范围由dsmi_get_device_count决定，当前实际支持的设备号，通过dsmi_list_device接口获取，订阅指定设备的故障事件； 订阅所有设备(-1)：订阅当前环境所有设备的故障事件。
timeout	输入	timeout >= 0：阻塞等待timeout(ms)时间，最大阻塞时间为30000ms(30s)； timeout = -1：阻塞等待永不超时。

filter	输入	可只订阅满足指定条件的事件，过滤条件如下：
		#define DSMI_EVENT_FILTER_FLAG_EVENT_ID (1UL << 0)
		#define DSMI_EVENT_FILTER_FLAG_SERVERITY (1UL << 1)
		#define DSMI_EVENT_FILTER_FLAG_NODE_TYPE (1UL << 2)
		#define DMS_MAX_EVENT_RESV_LENGTH 32
		struct dsmi_event_filter {
		unsigned long long filter_flag; /* 可单独使能某个过滤条件，也可将全部条件同时使能，过滤条件如下：
		0: 不使能过滤条件
		DSMI_EVENT_FILTER_FLAG_EVENT_ID: 只接收指定的事件
		DSMI_EVENT_FILTER_FLAG_SERVERITY: 只接收指定级别及以上的事件
		DSMI_EVENT_FILTER_FLAG_NODE_TYPE: 只接收指定节点类型的事件 */
		unsigned int event_id; /* 接收指定的事件 */
		unsigned char severity; /* 接收指定级别及以上的事件：见struct dms_fault_event结构体中severity定义 */
		unsigned char node_type; /* 接收指定节点类型的事件 */
		unsigned char resv[DMS_MAX_EVENT_RESV_LENGTH]; /* < reserve 32bytes */
		};

参数名	输入/输出	描述

event	输出	输出事件结构体定义如下：
		struct dsmi_event {
		enum dsmi_event_type type; /* 事件类型 */
		union {
		struct dms_fault_event dms_event; /* 事件内容 */
		} event_t;
		};
		type：
		当前只支持DMS_FAULT_EVENT类型，枚举定义如下：
		enum dsmi_event_type {
		DMS_FAULT_EVENT = 0,
		DSMI_EVENT_TYPE_MAX
		};
		dms_event：
		DMS_FAULT_EVENT类型对应的事件内容定义如下：
		#define DMS_MAX_EVENT_NAME_LENGTH 256
		#define DMS_MAX_EVENT_DATA_LENGTH 32
		#define DMS_MAX_EVENT_RESV_LENGTH 32
		struct dms_fault_event {
		unsigned int event_id; /* 事件id */
		unsigned short deviceid; /* 设备号 */
		unsigned char node_type; /* 节点类型 */
		unsigned char node_id; /* 节点id */
		unsigned char sub_node_type; /* 子节点类型 */
		unsigned char sub_node_id; /* 子节点id */
		unsigned char severity; /* 事件级别 0：提示，1：次要，2：重要，3：紧急 */
		unsigned char assertion; /* 事件类型 0：故障恢复，1：故障产生，2：一次性事件 */
		int event_serial_num; /* 告警序列号 */
		int notify_serial_num; /* 通知序列号 */
		unsigned long long alarm_raised_time; /* 事件产生时间：自1970年1月1日0点0分0秒开始至今的毫秒数 */
		char event_name[DMS_MAX_EVENT_NAME_LENGTH]; /* 事件描述信息 */
		char additional_info[DMS_MAX_EVENT_DATA_LENGTH]; /* 事件附加信息 */
		unsigned char resv[DMS_MAX_EVENT_RESV_LENGTH]; /**< reserve 32bytes */
		};

返回值

类型	描述
int	处理结果，返回0成功，其他失败返回**错误码**。

异常处理

无

约束说明

调用dsmi_read_fault_event接口后，timeout时间内产生的故障可以正常上报。

支持多进程不支持多线程，最大支持64个进程同时调用。

支持物理机场景下调用。

调用示例

```
....
struct dsmi_event_filter filter;
struct dsmi_event event = {0};
int dev_id, timeout, ret;

dev_id = -1; /* 订阅当前环境所有设备的故障事件 */
timeout = 1000;  /* 阻塞等待1000(ms) */
/* 过滤条件：只接收指定级别及以上的事件且只接收指定节点类型的事件 */
filter.filter_flag = DSMI_EVENT_FILTER_FLAG_SERVERITY |
DSMI_EVENT_FILTER_FLAG_NODE_TYPE;
filter.severity = 2; /* 只订阅2~3级别的事件 */
filter.node_type = 0x40; /* 只订阅模块ID为SOC类型的事件 */

ret = dsmi_read_fault_event(dev_id, timeout, filter, &event);
if(ret) {
// todo
}
// todo
....
```


4 MAC 地址管理

4.1 dsmi_get_mac_count

函数原型

int dsmi_get_mac_count(int device_id, int *count)

功能说明

查询MAC地址数量。

昇腾310 AI处理器场景下，该接口只支持miniRC场景。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。
count	输出	查询到的MAC地址数量。 昇腾310 AI处理器场景下，取值范围：0~1。

返回值

类型	描述
int	处理结果，返回0成功，失败返回错误码。

约束说明

昇腾310 AI处理器场景下，无约束。

调用示例

```
int ret;
int count;
ret = dsmi_get_mac_count(0,&count);
if(ret != 0) {
    //todo
    return ret;
}
...
```

4.2 dsmi_get_mac_addr

函数原型

int dsmi_get_mac_addr (int device_id, int mac_id, char *pmac_addr, unsigned int mac_addr_len)

功能说明

获取指定设备的MAC地址。

昇腾310 AI处理器场景下，该接口只支持miniRC场景。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。
mac_id	输入	取值范围：0~3。 **说明** 该参数在当前版本不使用。默认值为0，请保持默认值即可。
pmac_addr	输出	输出6个字节的MAC地址。
mac_addr_len	输入	MAC地址长度，固定长度6，单位byte。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

昇腾310 AI处理器场景下，无约束。

调用示例

```
int ret;
int i;
int count = -1;
char mac_addr[6] = {0};
ret = dsmi_get_mac_count(0,&count);
...
for (i = 0; i < count; i++){
    ret = dsmi_get_mac_addr(0, i, mac_addr, 6);
    if(ret != 0) {
        //todo
        return ret;
    }
    ...
}
...
```

4.3 dsmi_set_mac_addr

函数原型

int dsmi_set_mac_addr(int device_id, int mac_id, const char *pmac_addr, unsigned int mac_addr_len)

功能说明

设置指定设备的MAC地址。

昇腾310 AI处理器场景下，该接口只支持miniRC场景。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。
mac_id	输入	取值范围：0~3。 **说明** 该参数在当前版本不使用。默认值为0，请保持默认值即可。
pmac_addr	输入	设置6个字节的MAC地址。
mac_addr_len	输入	MAC地址长度，固定长度6，单位byte。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

调用该接口的程序必须在物理机的root用户下运行。

调用示例

```
int ret;
int mac_id = 0;
char mac_addr[6] = {0x52,0x12,0x36,0x26,0x82,0x66};
ret = dsmi_set_mac_addr(0, mac_id, mac_addr, 6);
if(ret != 0) {
    //todo
    return ret;
}
...
```

5 风扇管理

5.1 dsmi_get_fan_count

函数原型

int dsmi_get_fan_count(int device_id, int *count)

功能说明

获取Device上小风扇数，大部分场景一个昇腾AI处理器只有一个风扇，但可能存在支持多个风扇的情况，如果有多个小风扇，提供查询有几个风扇的接口。

昇腾310 AI处理器场景下，该接口只支持mini模块（包含mini模块作为RC或EP的场景）。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。
count	输出	查询小风扇个数，取值范围：目前固定为1。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

无。

调用示例

```
int ret = 0;
int count = 0;
ret = dsmi_get_fan_count(0, &count);
if(ret != 0) {
    //todo
    return ret;
}
...
```

5.2 dsmi_get_fan_speed

函数原型

int dsmi_get_fan_speed(int device_id, int fan_id, int *speed)

功能说明

查询指定风扇的转速，为风扇实际转速，单位RPM。

昇腾310 AI处理器场景下，该接口只支持mini模块（包含mini模块作为RC或EP的场景）。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。
fan_id	输入	如果有多个风扇，fan_id从1开始编号，大于等于1表示查询指定fan_id的风扇转速。 如果fan_id为0，则表示查询所有风扇的平均速度。
speed	输出	输出风扇转速值数组，由调用者申请。 调用成功后，该空间存储为风扇转速，单位为RPM，即转/分钟。 取值范围：0~(18000±10%)

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

无。

调用示例

```
int ret = 0;
int speed = 0;
int count = 0;
ret = dsmi_get_fan_count(0, &count);
...
for (i = 1; i <= count; i++){
    ret = dsmi_get_fan_speed(0, i, &speed);
    if(ret != 0) {
        //todo
        return ret;
    }
    ...
}
...
```


6 配置管理

6.1 dsmi_config_ecc_enable

函数原型

int dsmi_config_ecc_enable(int device_id, DSMI_DEVICE_TYPE device_type, int enable_flag)

功能说明

配置存储ECC的标记为使能或禁用。调用该接口配置ECC标记后，需要执行reboot命令重启系统，配置才能生效。

配置ECC标记后，可通过**6.2 dsmi_get_ecc_enable**接口查看ECC标记。

mini模块作为RC场景下，在容器中不支持该接口。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。
device_type	输入	设备类型，目前支持DSMI_DEVICE_TYPE_DDR。 typedef enum { DSMI_DEVICE_TYPE_DDR, DSMI_DEVICE_TYPE_SRAM, DSMI_DEVICE_TYPE_HBM, DSMI_DEVICE_TYPE_NPU, DSMI_DEVICE_TYPE_NONE = 0xff } DSMI_DEVICE_TYPE;
enable_flag	输入	1：使能 0：禁用

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

- 调用该接口的程序必须在物理机的root用户下运行。
- 使能ECC后，会导致可用DDR内存减少，推理性能下降，且可能存在波动。
- 此接口设置的内容会涉及flash擦写，由于flash擦写次数是有限制的，所以不建议频繁调用该接口。

调用示例

```
int ret = 0;
ret = dsmi_config_ecc_enable(0, DSMI_DEVICE_TYPE_DDR, 1);
if(ret != 0) {
    //todo
    return ret;
}
ret = dsmi_config_ecc_enable(0, DSMI_DEVICE_TYPE_DDR, 0);
if(ret != 0) {
    //todo
    return ret;
}
```

6.2 dsmi_get_ecc_enable

函数原型

int dsmi_get_ecc_enable(int device_id, DSMI_DEVICE_TYPE device_type, int *enable_flag)

功能说明

查询存储ECC的使能标记。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。
device_type	输入	设备类型，目前只支持DSMI_DEVICE_TYPE_DDR。 typedef enum { DSMI_DEVICE_TYPE_DDR, DSMI_DEVICE_TYPE_SRAM, DSMI_DEVICE_TYPE_HBM, DSMI_DEVICE_TYPE_NPU, DSMI_HBM_RECORDED_SINGLE_ADDR, DSMI_HBM_RECORDED_MULTI_ADDR, DSMI_DEVICE_TYPE_NONE = 0xff } DSMI_DEVICE_TYPE;
enable_flag	输出	1：使能 0：禁用

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

昇腾310P AI处理器容器内不支持该接口，在该场景下使用将会返回错误。

昇腾310 AI处理器场景下，无约束。

调用示例

```
int ret = 0;
int enable_flag = 0;
```

```
ret = dsmi_get_ecc_enable(0, DSMI_DEVICE_TYPE_DDR, &enable_flag);
if(ret != 0) {
    //todo
    return ret;
}
...
```

6.3 dsmi_get_system_time

函数原型

int dsmi_get_system_time(int device_id, unsigned int *ntime_stamp)

功能说明

获取系统时间。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。
ntime_stamp	输出	表示从1970/01/01 00:00:00至今的秒数值。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

无。

调用示例

```
int ret = 0;
unsigned int time = 0;

ret = dsmi_get_system_time (0, &time);
if(ret != 0) {
    //todo
    return ret;
}
...
```


6.4 dsmi_set_device_ip_address

函数原型

int dsmi_set_device_ip_address (int device_id, int port_type, int port_id, ip_addr_t ip_address, ip_addr_t mask_address)

功能说明

设置ip地址和mask地址。

昇腾310 AI处理器MiniRC场景下，不支持该接口。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由dsmi_get_device_count决定。当前实际支持的设备号，通过dsmi_list_device接口获取。
port_type	输入	指定网口类型。 昇腾310 AI处理器场景下，取值范围为VNIC：0x00。用户在调用接口时可以调用宏：DSMI_VNIC_PORT (0)。
port_id	输入	指定网口号。 昇腾310 AI处理器场景下为保留字段，取值范围：【0~255】。

ip_address	输入	`#define DSMI_ARRAY_IPV4_NUM 4`
		`#define DSMI_ARRAY_IPV6_NUM 16`
		
		`typedef struct ip_addr {`
		`  union {`
		`  unsigned char ip6[DSMI_ARRAY_IPV6_NUM];`
		`  unsigned char ip4[DSMI_ARRAY_IPV4_NUM];`
		`  } u_addr;`
		`  enum ip_addr_type ip_type;`
		`} ip_addr_t;`
		ip_addr_type结构体定义如下：
		`enum ip_addr_type {`
		`  /** IPv4 */`
		`  IPADDR_TYPE_V4 = 0U,`
		`  /** IPv6 */`
		`  IPADDR_TYPE_V6 = 1U,`
		`  /** IPv4+IPv6 ("dual-stack") */`
		`  IPADDR_TYPE_ANY = 2U`
		`};`

参数名	输入/输出	描述

mask_address	输入	#define DSMI_ARRAY_IPV4_NUM 4
		#define DSMI_ARRAY_IPV6_NUM 16
		
		typedef struct ip_addr {
		union {
		unsigned char ip6[DSMI_ARRAY_IPV6_NUM];
		unsigned char ip4[DSMI_ARRAY_IPV4_NUM];
		} u_addr;
		enum ip_addr_type ip_type;
		} ip_addr_t;
		ip_addr_type结构体定义如下：
		enum ip_addr_type {
		/** IPv4 */
		IPADDR_TYPE_V4 = 0U,
		/** IPv6 */
		IPADDR_TYPE_V6 = 1U,
		/** IPv4+IPv6 ("dual-stack") */
		IPADDR_TYPE_ANY = 2U
		};
		说明
		以ip_address的ip_type为准，mask_address的ip_type无效。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

只支持IPV4地址。

- 调用该接口的程序必须在root用户下运行。
- 昇腾310 AI处理器场景下，仅支持物理机。

调用示例

```
int ret = 0;
int device_id = 0;
int port_type = 0;
int port_id = 0;
unsigned int ip_addr = 0xC801A8C0; //192.168.1.200
unsigned int mask_addr = 0x00FFFFFF; //255.255.255.0
ip_addr_t ip_address = {0};
ip_addr_t ip_mask_address={0};

memcpy(&(ip_address.u_addr.ip4[0]),&ip_addr,4);
memcpy(&(ip_mask_address.u_addr.ip4[0]),&mask_addr,4);
ret = dsmi_set_device_ip_address(device_id, port_type, port_id, ip_address, ip_mask_address);
if(ret != 0) {
    //todo
    return ret;
```

```
}
...
```

6.5 dsmi_get_device_ip_address

函数原型

int dsmi_get_device_ip_address (int device_id, int port_type, int port_id, ip_addr_t *ip_address, ip_addr_t *mask_address)

功能说明

获取IP地址和mask地址。

昇腾310 AI处理器MiniRC场景下，不支持该接口。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。
port_type	输入	指定网口类型。 昇腾310 AI处理器场景下，取值范围为VNIC：0x00。用户在调用接口时可以调用宏：DSMI_VNIC_PORT (0)。
port_id	输入	指定网口号。 昇腾310 AI处理器场景下为保留字段，取值范围：【0~255】。

ip_address	输入和输出	`#define DSMI_ARRAY_IPV4_NUM 4`
		`#define DSMI_ARRAY_IPV6_NUM 16`
		
		`typedef struct ip_addr {`
		`  union {`
		`  unsigned char ip6[DSMI_ARRAY_IPV6_NUM];`
		`  unsigned char ip4[DSMI_ARRAY_IPV4_NUM];`
		`  } u_addr;`
		`  enum ip_addr_type ip_type;  //这是一个输入值`
		`} ip_addr_t;`
		ip_addr_type结构体定义如下：
		`enum ip_addr_type {`
		`  /** IPv4 */`
		`  IPADDR_TYPE_V4 = 0U,`
		`  /** IPv6 */`
		`  IPADDR_TYPE_V6 = 1U,`
		`  /** IPv4+IPv6 ("dual-stack") */`
		`  IPADDR_TYPE_ANY = 2U`
		`};`

参数名	输入/输出	描述

mask_address	输入和输出	#define DSMI_ARRAY_IPV4_NUM 4
		#define DSMI_ARRAY_IPV6_NUM 16
		
		typedef struct ip_addr {
		union {
		unsigned char ip6[DSMI_ARRAY_IPV6_NUM];
		unsigned char ip4[DSMI_ARRAY_IPV4_NUM];
		} u_addr;
		enum ip_addr_type ip_type; //这是一个输入值
		} ip_addr_t;
		ip_addr_type结构体定义如下：
		enum ip_addr_type {
		/** IPv4 */
		IPADDR_TYPE_V4 = 0U,
		/** IPv6 */
		IPADDR_TYPE_V6 = 1U,
		/** IPv4+IPv6 ("dual-stack") */
		IPADDR_TYPE_ANY = 2U
		};
		说明
		以ip_address的ip_type为准，mask_address的ip_type无效。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

昇腾310 AI处理器场景下，无约束。

调用示例

```
int ret = 0;
int device_id = 0;
int port_type = 0;
int port_id = 0;
ip_addr_t ip_address = {0};
ip_addr_t ip_mask_address={0};

ret = dsmi_get_device_ip_address(device_id, port_type, port_id, &ip_address, &mask_address);
if(ret != 0) {
    //todo
    return ret;
}
```

6.6 dsmi_set_user_config

函数原型

int dsmi_set_user_config(int device_id, const char *config_name, unsigned int buf_size, unsigned char *buf)

功能说明

设置用户配置。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。

参数名	输入/输出	描述
config_name	输入	目前已实现功能的配置项名称如下，配置项名称的字符串长度最大为32。 昇腾310 AI处理器场景下，已实现功能的配置项：ddr_ecc_enable、aicpu_config、ssh_status、set_nve_level、recovery_mode_passwd。不支持用户自定义名称配置。 配置项功能说明如下： ● "ddr_ecc_enable"：用于使能或禁用ECC。 ● "aicpu_config"：用于配置AI CPU和Control CPU的配比。 ● "ssh_status"：用于打开或关闭ssh服务。miniRC场景下不支持该配置。 ● "set_nve_level"：用于设置功耗和算力档位。 ● "recovery_mode_passwd"：使能维护OS登录，并将商用OS中的HwHiAiUser和root用户密码同步到维护OS，首次登录时不要求修改密码。调用该接口前需要保证商用OS中已有HwHiAiUser和root用户，且已正常设置密码，密码要求使用SHA512算法加密。仅支持 miniRC场景。 **说明** 昇腾310 AI处理器场景下: ssh_status：打开ssh服务之后，请务必立即修改Device的ssh登录密码（包括HwHiAiUser用户和root用户的密码）。由于配置持久化功能，打开ssh服务之后不会自动关闭，建议在使用完成后及时主动关闭ssh服务。ssh服务开启后，系统默认的ssh服务占用的内存大小限制为50MB，若用户想取消此限制，可进行如下操作： 1. 以HwHiAiUser用户登录Device，然后切换到root用户。 2. 执行如下命令取消对ssh服务占用内存大小的限制。 cat /sys/fs/cgroup/memory/sshdmemory/cgroup.procs |xargs -n1 >>/sys/fs/cgroup/memory/cgroup.procs;rmdir /sys/fs/cgroup/memory/sshdmemory/ **需要注意，每次Host重启后，都会恢复默认50MB的ssh服务内存限额。** 调用该接口配置标记后，需要执行reboot命令重启系统，配置才能生效。配置生效后，可通过**6.7 dsmi_get_user_config**接口查看配置结果。

参数名	输入/输出	描述
buf_size	输入	buf长度，单位为byte，最大长度为1K byte。 昇腾310 AI处理器场景下，支持：ddr_ecc_enable、aicpu_config、ssh_status、set_nve_level、recovery_mode_passwd配置。 目前支持处理的配置项名称如下： ● 如果配置"ddr_ecc_enable"：buf_size参数配置为1。 ● 如果配置"aicpu_config"：buf_size参数配置为1。 ● 如果配置"ssh_status"：buf_size参数配置为1。 ● 如果配置"set_nve_level"：buf_size参数配置为1。 ● 如果配置"recovery_mode_passwd"：buf_size参数配置为1024。

参数名	输入/输出	描述
buf	输入	buf指针，指向配置项内容。 昇腾310 AI处理器场景下，支持：ddr_ecc_enable、aicpu_config、ssh_status、set_nve_level、recovery_mode_passwd配置。 目前支持处理的配置项名称如下： ● 针对"ddr_ecc_enable"配置项，配置项内容支持如下选项： 1：表示使能ECC 0：表示禁用ECC 昇腾310 AI处理器场景下，默认值为0。 ● 针对"aicpu_config"配置项，配置项内容支持如下选项，默认值是240： 192：表示2个AI CPU，6个Control CPU 240：表示4个AI CPU，4个Control CPU 252：表示6个AI CPU，2个Control CPU ● 针对"ssh_status"配置项，配置项内容支持如下选项，默认值是0： 1：打开ssh服务 0：关闭ssh服务 ● 针对"set_nve_level"配置项，配置项内容支持如下选项，默认值是3： 0：功耗和算力档位为low 1：功耗和算力档位为middle 2：功耗和算力档位为high 3：功耗和算力档位为full ● 针对"recovery_mode_passwd"配置项，配置项内容未使用，默认值是0。 ● 除了如上固定的名称外，其他配置项内容请根据实际情况配置。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码**。

约束说明

- 昇腾310 AI处理器场景下。
  - 调用该接口的程序必须在物理机的root用户下运行。
  - 使能ECC后，会导致可用DDR内存减少，推理性能下降。

- 此接口设置的内容会涉及flash擦写，由于flash擦写次数是有限制的，所以不建议频繁调用该接口。

调用示例

```
#define BUF_SIZE 1
int ret = 0;
int device_id = 0;
char *config_name = "ddr_ecc_enable";
unsigned char buf[BUF_SIZE] = {0};

ret=dsmi_set_user_config(device_id,config_name, BUF_SIZE, buf);
if(ret != 0){
    //todo
    return ret;
}
...
```

6.7 dsmi_get_user_config

函数原型

int dsmi_get_user_config(int device_id, const char *config_name, unsigned int buf_size, unsigned char *buf)

功能说明

获取用户配置。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。

参数名	输入/输出	描述
config_name	输入	目前支持查询的配置项名称如下，配置项名称的字符串长度最大为32。 昇腾310 AI处理器场景下，支持已实现功能的配置项：ddr_ecc_enable、aicpu_config、ssh_status、get_nve_level。不支持获取用户自定义名称配置。 目前支持处理的配置项名称功能说明如下： ● "ddr_ecc_enable"：用于使能或禁用ECC。 ● "aicpu_config"：用于配置AI CPU和Control CPU的配比。 ● "ssh_status"：通过**6.6 dsmi_set_user_config**设置的ssh服务状态，取值如下 – 0:关闭 – 1:开启 **说明** 查询接口只查询设置接口配置的ssh_status,不适用于查询用户通过文件系统或者手工执行命令开启ssh功能的状态。 miniRC场景下不支持该配置。 ● "get_nve_level"：用于获取功耗和算力档位。
buf_size	输入	buf长度，最大长度为1K byte。 该参数的配置，请参见**6.6 dsmi_set_user_config**。
buf	输出	buf指针，指向配置项内容。 具体配置项内容的含义，请参见**6.6 dsmi_set_user_config**。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码**。

约束说明

无。

调用示例

```
#define BUF_SIZE 1
```

```
int ret = 0;
int device_id = 0;
char *config_name = "ddr_ecc_enable";
unsigned char buf[BUF_SIZE] = {0};

ret=dsmi_get_user_config(device_id, config_name, BUF_SIZE, buf);
if(ret != 0){
    //todo
    return ret;
}
...
```

6.8 dsmi_clear_user_config

函数原型

int dsmi_clear_user_config(int device_id, const char *config_name)

功能说明

重置用户配置。

默认值请参见**6.6 dsmi_set_user_config**。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。

参数名	输入/输出	描述
config_name	输入	目前支持查询的配置项名称如下，配置项名称的字符串长度最大为32。 昇腾310 AI处理器场景下，支持已实现功能的配置项：ddr_ecc_enable、aicpu_config、ssh_status、recovery_mode_passwd。不支持获取用户自定义名称配置。 目前支持处理的配置项名称功能说明如下： • "ddr_ecc_enable"：用于使能或禁用ECC。 • "aicpu_config"：用于配置AI CPU和Control CPU的配比。 • "recovery_mode_passwd"：用于禁止用户登录维护OS。仅支持miniRC场景。 • "ssh_status"：通过**6.6 dsmi_set_user_config**设置的ssh服务状态，取值如下 – 0:关闭 – 1:开启 **说明** 查询接口只查询设置接口配置的ssh_status,不适用于查询用户通过文件系统或者手工执行命令开启ssh功能的状态。 miniRC场景下不支持该配置。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码**。

约束说明

昇腾310 AI处理器场景下，调用该接口的程序必须在物理机的root用户下运行。此接口设置的内容会涉及flash擦写，由于flash擦写次数是有限制的，所以不建议频繁调用该接口。

调用示例

```
int ret = 0;
int device_id = 0;
char *config_name = "ddr_ecc_enable";

ret = dsmi_clear_user_config(device_id, config_name);
if(ret != 0){
    //todo
    return ret;
}
...
```

6.9 dsmi_get_pcie_error_rate

函数原型

int dsmi_get_pcie_error_rate(int device_id, struct dsmi_chip_pcie_err_rate_stru *pcie_err_code_info)

功能说明

获取PCIe链路误码信息。

昇腾310 AI处理器场景下，支持PCIe标卡、mini模块（仅支持mini模块作为EP的场景）。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号。 有效值范围：0~63，当前实际支持的设备号，通过**3.3 dsmi_list_device**接口获取。
pcie_err_code_info	输出	typedef struct dsmi_chip_pcie_err_rate_stru { unsigned int reg_deskew_fifo_overflow_intr_status; unsigned int reg_symbol_unlock_intr_status; unsigned int reg_deskew_unlock_intr_status; unsigned int reg_phystatus_timeout_intr_status; unsigned int symbol_unlock_counter; unsigned int pcs_rx_err_cnt; unsigned int phy_lane_err_counter; unsigned int pcs_rcv_err_status; unsigned int symbol_unlock_err_status; unsigned int phy_lane_err_status; unsigned int dl_lcrc_err_num; unsigned int dl_dcrc_err_num; } PCIE_ERR_RATE_INFO_STU;

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码**。

约束说明

仅支持固件1.73.5.6.50及以上版本。

调用示例

```
int ret = 0;
int device_id = 0;
```

```
struct dsmi_chip_pcie_err_rate_stru *pcie_err_code_info = NULL;
pcie_err_code_info = (struct dsmi_chip_pcie_err_rate_stru *)
   malloc(sizeof(struct dsmi_chip_pcie_err_rate_stru));
if(!pcie_err_code_info) {
    return –EINVAL;
}
ret=dsmi_get_pcie_error_rate(device_id, pcie_err_code_info);
if(ret != 0){
//todo:记录日志
free(pcie_err_code_info);
return ret;
}
…
```

6.10 dsmi_clear_pcie_error_rate

函数原型

int dsmi_clear_pcie_error_rate(int device_id)

功能说明

清除当前PCIe链路误码统计信息。

昇腾310 AI处理器场景下，支持PCIe标卡、mini模块（仅支持mini模块作为EP的场景）。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号。 有效值范围：0~63，当前实际支持的设备号，通过**3.3 dsmi_list_device**接口获取。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码**。

约束说明

调用该接口的程序必须在物理机的root用户下运行，若在物理机的非root用户，或在容器下运行，则会返回权限错误。

仅支持固件1.73.5.6.50及以上版本。

调用示例

```
int ret = 0;
int device_id = 0;
```



```
ret=dsmi_clear_pcie_error_rate(device_id);
if(ret != 0){
//todo:记录日志
return ret;
}
…
```


7 软件升级

7.1 dsmi_get_component_count

函数原型

int dsmi_get_component_count(int device_id, unsigned int *component_count)

功能说明

获取可升级组件的个数，不包含recovery组件。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。
component_count	输出	返回组件的个数。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

昇腾310 AI处理器场景下，调用该接口的程序必须在root用户下运行。

调用示例

```
int ret = 0;
unsigned int component_num = 0;

ret =dsmi_get_component_count(0, &component_num);
if(ret != 0) {
    //todo
    return ret;
}
```

7.2 dsmi_get_component_list

函数原型

int dsmi_get_component_list(int device_id, DSMI_COMPONENT_TYPE *component_table, unsigned int component_count)

功能说明

获取可升级组件列表，不包含recovery组件。

需要先调用**dsmi_get_component_count**接口获取可升级组件的个数后，再调用dsmi_get_component_list接口获取组件列表。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。
component_count	输入	component_table的长度，表示获取的组件个数。

参数名	输入/输出	描述
component_table	输出	返回可升级组件列表，具体值含义如下： 昇腾310 AI处理器场景下，目前支持DSMI_COMPONENT_TYPE_NVE、DSMI_COMPONENT_TYPE_XLOADER、DSMI_COMPONENT_TYPE_M3FW、DSMI_COMPONENT_TYPE_UEFI、DSMI_COMPONENT_TYPE_TEE。 在mini模块作为rc的场景中，还支持: DSMI_COMPONENT_TYPE_KERNEL、 DSMI_COMPONENT_TYPE_DTB、 DSMI_COMPONENT_TYPE_ROOTFS typedef enum dsmi_component_type { DSMI_COMPONENT_TYPE_NVE, DSMI_COMPONENT_TYPE_XLOADER, DSMI_COMPONENT_TYPE_M3FW, DSMI_COMPONENT_TYPE_UEFI, DSMI_COMPONENT_TYPE_TEE, DSMI_COMPONENT_TYPE_KERNEL, DSMI_COMPONENT_TYPE_DTB, DSMI_COMPONENT_TYPE_ROOTFS, DSMI_COMPONENT_TYPE_IMU, DSMI_COMPONENT_TYPE_IMP, DSMI_COMPONENT_TYPE_AICPU, DSMI_COMPONENT_TYPE_HBOOT1_A, DSMI_COMPONENT_TYPE_HBOOT1_B, DSMI_COMPONENT_TYPE_HBOOT2, DSMI_COMPONENT_TYPE_DDR, DSMI_COMPONENT_TYPE_LP, DSMI_COMPONENT_TYPE_HSM, DSMI_COMPONENT_TYPE_SAFETY_ISLAND, DSMI_COMPONENT_TYPE_HILINK, DSMI_COMPONENT_TYPE_RAWDATA, DSMI_COMPONENT_TYPE_SYSDRV, DSMI_COMPONENT_TYPE_ADSAPP, DSMI_COMPONENT_TYPE_COMISOLATOR, DSMI_COMPONENT_TYPE_CLUSTER, DSMI_COMPONENT_TYPE_CUSTOMIZED, DSMI_COMPONENT_TYPE_SYS_BASE_CONFIG, DSMI_COMPONENT_TYPE_RECOVERY, DSMI_COMPONENT_TYPE_HILINK2, DSMI_COMPONENT_TYPE_LOGIC_BIST, DSMI_COMPONENT_TYPE_ATF, DSMI_COMPONENT_TYPE_MAX, UPGRADE_AND_RESET_ALL_COMPONENT = 0xFFFFFFF7, UPGRADE_ALL_IMAGE_COMPONENT = 0xFFFFFFFD, UPGRADE_ALL_FIRMWARE_COMPONENT = 0xFFFFFFFE, UPGRADE_ALL_COMPONENT = 0xFFFFFFFF } DSMI_COMPONENT_TYPE;

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

昇腾310 AI处理器场景下，调用该接口的程序必须在root用户下运行。

调用示例

```
int ret = -EINVAL;
unsigned int component_num = 0;
DSMI_COMPONENT_TYPE *component_table = NULL;
ret =dsmi_get_component_count(0, &component_num);
if(ret != 0) {
   //todo
   return ret;
}
component_table = (DSMI_COMPONENT_TYPE*)malloc(sizeof(DSMI_COMPONENT_TYPE) *
component_num);
if(component_table == NULL) {
   //todo
   return ret;
}
ret = dsmi_get_component_list(0, component_table, component_num);
if(ret != 0) {
   //todo
   free(component_table);
   return ret;
}
...
```

7.3 dsmi_upgrade_start

函数原型

int dsmi_upgrade_start(int device_id, DSMI_COMPONENT_TYPE component_type, const char *file_name)

功能说明

启动指定昇腾AI处理器的升级，指定升级固件类型、升级的文件名，本接口返回成功仅代表升级命令发送成功，不代表升级完成；升级是否完成需要通过**dsmi_upgrade_get_state**查看状态和进度来决定，状态为idle且进度为100%代表本次升级成功，其他情况则视为升级失败。

⚠ 注意

为避免出现固件间的版本不兼容问题，不建议用户使用该接口对单个或者部分固件进行升级。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。

参数名	输入/输出	描述
component_type	输入	固件类型，目前支持如下几种： 昇腾310 AI处理器场景下，目前支持DSMI_COMPONENT_TYPE_NVE、DSMI_COMPONENT_TYPE_XLOADER、DSMI_COMPONENT_TYPE_M3FW、DSMI_COMPONENT_TYPE_UEFI、DSMI_COMPONENT_TYPE_TEE、UPGRADE_AND_RESET_ALL_COMPONENT、UPGRADE_ALL_COMPONENT 。 在mini模块作为RC的场景中，还支持: DSMI_COMPONENT_TYPE_KERNEL、 DSMI_COMPONENT_TYPE_DTB、 DSMI_COMPONENT_TYPE_ROOTFS 在mini模块作为RC的场景中，不支持UPGRADE_AND_RESET_ALL_COMPONENT。 结构体定义如下： typedef enum dsmi_component_type { DSMI_COMPONENT_TYPE_NVE, DSMI_COMPONENT_TYPE_XLOADER, DSMI_COMPONENT_TYPE_M3FW, DSMI_COMPONENT_TYPE_UEFI, DSMI_COMPONENT_TYPE_TEE, DSMI_COMPONENT_TYPE_KERNEL, DSMI_COMPONENT_TYPE_DTB, DSMI_COMPONENT_TYPE_ROOTFS, DSMI_COMPONENT_TYPE_IMU, DSMI_COMPONENT_TYPE_IMP, DSMI_COMPONENT_TYPE_AICPU, DSMI_COMPONENT_TYPE_HBOOT1_A, DSMI_COMPONENT_TYPE_HBOOT1_B, DSMI_COMPONENT_TYPE_HBOOT2, DSMI_COMPONENT_TYPE_DDR, DSMI_COMPONENT_TYPE_LP, DSMI_COMPONENT_TYPE_HSM, DSMI_COMPONENT_TYPE_SAFETY_ISLAND, DSMI_COMPONENT_TYPE_HILINK, DSMI_COMPONENT_TYPE_RAWDATA, DSMI_COMPONENT_TYPE_SYSDRV, DSMI_COMPONENT_TYPE_ADSAPP, DSMI_COMPONENT_TYPE_COMISOLATOR, DSMI_COMPONENT_TYPE_CLUSTER, DSMI_COMPONENT_TYPE_CUSTOMIZED, DSMI_COMPONENT_TYPE_SYS_BASE_CONFIG, DSMI_COMPONENT_TYPE_RECOVERY, DSMI_COMPONENT_TYPE_MAX, UPGRADE_AND_RESET_ALL_COMPONENT = 0xFFFFFFF7, UPGRADE_ALL_IMAGE_COMPONENT = 0xFFFFFFFD, UPGRADE_ALL_FIRMWARE_COMPONENT = 0xFFFFFFFE, UPGRADE_ALL_COMPONENT = 0xFFFFFFFF } DSMI_COMPONENT_TYPE;

参数名	输入/输出	描述
		说明 昇腾310 AI处理器场景下，以下component_type调用成功，但内部功能没有实现，实际结果无意义。 DSMI_COMPONENT_TYPE_M3FW DSMI_COMPONENT_TYPE_TEE 昇腾310 AI处理器在mini模块作为RC的场景中，以下component_type调用成功，但内部功能没有实现，实际结果无意义。 DSMI_COMPONENT_TYPE_M3FW DSMI_COMPONENT_TYPE_TEE DSMI_COMPONENT_TYPE_KERNEL DSMI_COMPONENT_TYPE_DTB DSMI_COMPONENT_TYPE_ROOTFS
file_name	输入	固件文件名（包含路径）。 当 component_type为UPGRADE_ALL_COMPONENT或UPGRADE_AND_RESET_ALL_COMPONENT 时，参数为带路径的配置文件，配置文件可参考{install_path}/firmware/tools/conf/upgrade.cfg； 当 component_type为其他类型时，为单独的带路径的组件包名。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码**。

约束说明

- 此接口设置的内容会涉及flash擦写，由于flash擦写次数是有限制的，所以不建议频繁调用该接口。
- 针对相同device_id，本接口不支持并发，如果并发调用，会返回错误。
- 当dsmi_upgrade_get_state获取的升级状态为：IS_UPGRADING、UPGRADE_NOT_SUPPORT、UPGRADE_SYNCHRONIZING时不可触发升级。
- 升级完成后，需要执行复位操作生效。
- 昇腾310 AI处理器场景下，还有如下约束：
  - 调用该接口的程序必须在物理机的root用户下运行，其他场景下则会返回错误。
  - 输入UPGRADE_AND_RESET_ALL_COMPONENT参数会清除Device的ssh登录密码。清除密码后，请务必立即修改Device的ssh登录密码（包括HwHiAiUser用户和root用户的密码）。

- /{install_path}/firmware/image/nve.bin镜像文件用于保存昇腾AI处理器的一些配置参数。当前版本已不再使用该镜像文件，但是部分老版本还在使用该镜像文件。

  由于存在从新版本降级到老版本的场景，为了确保降级之后昇腾AI处理器可以正常工作，当前版本会继续保留更新nve.bin镜像文件的功能，确保nve.bin镜像文件和需要降级的其它镜像文件的版本一致性。

调用示例

```
int ret = 0;

ret = dsmi_upgrade_start(0, DSMI_COMPONENT_TYPE_XLOADER , “./xloader.bin”);
if(ret != 0) {
    //todo
    return ret;
}
...
```

7.4 dsmi_upgrade_get_state

函数原型

int dsmi_upgrade_get_state(int device_id, unsigned char *schedule, unsigned char *upgrade_status)

功能说明

查询固件升级状态及进度。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。
schedule	输出	升级进度，0~100百分比。

upgrade_status	输出	升级状态，目前支持如下几种：
		typedef enum dsmi_upgrade_device_state {
		UPGRADE_IDLE_STATE = 0, // 升级完成，可触发升级
		IS_UPGRADING = 1, // 触发升级后，状态为正在升级中，不可触发升级
		UPGRADE_NOT_SUPPORT = 2, // 不支持升级，接口返回报错
		UPGRADE_UPGRADE_FAIL = 3, // 升级失败，可触发升级
		UPGRADE_STATE_NONE = 4, // 软件内部默认初始状态
		UPGRADE_WAITTING_RESTART = 5, // 表示正在等待重启
		UPGRADE_WAITTING_SYNC = 6, // 表示正在等待同步
		UPGRADE_SYNCHRONIZING = 7 // 表示正在同步升级状态
		} DSMI_UPGRADE_DEVICE_STATE;
		昇腾310 AI处理器场景下，升级状态的取值范围仅支持0~4。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

昇腾310 AI处理器场景下，调用该接口的程序必须在root用户下运行。

调用示例

```
int ret = 0;
unsigned char schedule;
unsigned char upgrade_status;
ret = dsmi_upgrade_get_state(0, &schedule, & upgrade_status);
if(ret != 0) {
    //todo
    return ret;
}
```

7.5 dsmi_upgrade_get_component_static_version

函数原型

int dsmi_upgrade_get_component_static_version(int device_id, DSMI_COMPONENT_TYPE component_type, unsigned char* version_str, unsigned int version_len, unsigned int *ret_len)

功能说明

查询组件的静态版本。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。

参数名	输入/输出	描述
component_type	输入	固件类型，目前支持如下几种： 昇腾310 AI处理器场景下，目前支持 DSMI_COMPONENT_TYPE_NVE、 DSMI_COMPONENT_TYPE_XLOADER、 DSMI_COMPONENT_TYPE_M3FW、 DSMI_COMPONENT_TYPE_UEFI、 DSMI_COMPONENT_TYPE_TEE、 在mini模块作为rc的场景中，还支持: DSMI_COMPONENT_TYPE_KERNEL、 DSMI_COMPONENT_TYPE_DTB、 DSMI_COMPONENT_TYPE_ROOTFS typedef enum dsmi_component_type { DSMI_COMPONENT_TYPE_NVE, DSMI_COMPONENT_TYPE_XLOADER, DSMI_COMPONENT_TYPE_M3FW, DSMI_COMPONENT_TYPE_UEFI, DSMI_COMPONENT_TYPE_TEE, DSMI_COMPONENT_TYPE_KERNEL, DSMI_COMPONENT_TYPE_DTB, DSMI_COMPONENT_TYPE_ROOTFS, DSMI_COMPONENT_TYPE_IMU, DSMI_COMPONENT_TYPE_IMP, DSMI_COMPONENT_TYPE_AICPU, DSMI_COMPONENT_TYPE_HBOOT1_A, DSMI_COMPONENT_TYPE_HBOOT1_B, DSMI_COMPONENT_TYPE_HBOOT2, DSMI_COMPONENT_TYPE_DDR, DSMI_COMPONENT_TYPE_LP, DSMI_COMPONENT_TYPE_HSM, DSMI_COMPONENT_TYPE_SAFETY_ISLAND, DSMI_COMPONENT_TYPE_HILINK, DSMI_COMPONENT_TYPE_RAWDATA, DSMI_COMPONENT_TYPE_SYSDRV, DSMI_COMPONENT_TYPE_ADSAPP, DSMI_COMPONENT_TYPE_COMISOLATOR, DSMI_COMPONENT_TYPE_CLUSTER, DSMI_COMPONENT_TYPE_CUSTOMIZED, DSMI_COMPONENT_TYPE_SYS_BASE_CONFIG, DSMI_COMPONENT_TYPE_RECOVERY, DSMI_COMPONENT_TYPE_HILINK2, DSMI_COMPONENT_TYPE_LOGIC_BIST, DSMI_COMPONENT_TYPE_ATF, DSMI_COMPONENT_TYPE_MAX, UPGRADE_AND_RESET_ALL_COMPONENT = 0xFFFFFFF7, UPGRADE_ALL_IMAGE_COMPONENT = 0xFFFFFFFD, UPGRADE_ALL_FIRMWARE_COMPONENT = 0xFFFFFFFE, UPGRADE_ALL_COMPONENT = 0xFFFFFFFF } DSMI_COMPONENT_TYPE;

参数名	输入/输出	描述
version_str	输出	存放固件版本号信息的内存空间, 大小不能小于64Byte。
version_len	输入	version_str的内存大小，单位Byte。
ret_len	输出	版本号长度。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

昇腾310 AI处理器场景下，调用该接口的程序必须在root用户下运行。

调用示例

```
int ret = 0;
unsigned int len = 0;
unsigned char version_str[64] = {0};

ret = dsmi_upgrade_get_component_static_version(0,
DSMI_COMPONENT_TYPE_XLOADER,version_str, 64, &len);
if(ret != 0) {
    //todo
    return ret;
}
...
```

7.6 dsmi_get_version

函数原型

int dsmi_get_version(int device_id, char* version_str, unsigned int version_len, unsigned int *ret_len)

功能说明

查询文件系统版本。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。
version_str	输出	返回系统版本。 该空间由用户申请，大小为64Byte。
version_len	输入	version_str的内存空间大小，单位Byte。
ret_len	输出	返回版本号长度。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

无

调用示例

```
int ret = 0;
int len = -1;
unsigned char version_str[64] = {0};

ret = dsmi_get_version (0, version_str, 64, &len);
if(ret != 0) {
    //todo
    return ret;
}
...
```

8 昇腾 AI 处理器复位启动

8.1 使用前必读

- 单独使用DSMI接口只能实现带内热复位功能。
- 通过DSMI接口实现带内热复位需要调用**dsmi_hot_reset_soc**接口。

8.2 接口说明

8.2.1 dsmi_pre_reset_soc

函数原型

int dsmi_pre_reset_soc(int device_id)

功能说明

昇腾AI处理器预复位，发起昇腾AI处理器预复位接口，预复位目的是解除上层驱动及软件对此昇腾AI处理器的依赖。预复位完成后可以对此昇腾AI处理器进行隔离或实际复位操作。

一般用于实际复位的操作流程：dsmi_pre_reset_soc -> reset –> dsmi_rescan_soc。

reset功能是通过BMC（Baseboard Management Controller）带外通道完成，仅支持华为服务器；为保证复位功能正常使用，请升级服务器的BMC到最新版本。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

调用该接口的程序必须在物理机root用户下运行。

调用该接口前，请停掉昇腾AI处理器上的所有业务。

调用示例

```
int ret = 0;
ret = dsmi_pre_reset_soc(0);
if(ret != 0) {
    //todo
    return ret;
}
...
```

8.2.2 dsmi_hot_reset_soc

函数原型

int dsmi_hot_reset_soc(int device_id)

功能说明

对指定昇腾AI处理器复位，包含预复位、复位和扫描三部分，用于昇腾AI处理器故障的恢复。预复位的目的是解除上层驱动及软件对此昇腾AI处理器的依赖。扫描的目的是将设备添加到系统中。

昇腾310 AI处理器场景下，该接口仅支持PCIe标卡和200EP场景，支持通过创建子进程并行复位所有芯片。

参数说明

参数名	输入/输出	描述
device_id	输入	当前支持设置为0xff，表示对Host侧服务器下所有昇腾AI处理器进行复位。 在昇腾310 AI处理器的AMP模式下，可以指定单个设备号进行复位，设备号有效值范围：0~63。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

异常处理

无。

约束说明

本接口返回成功仅代表复位命令发送成功，不代表复位完成；复位是否完成需要通过**dsmi_get_device_boot_status**调用返回的信息来决定。

调用该接口前，请停掉昇腾AI处理器上的所有业务。

该接口只是提供昇腾AI处理器复位功能。

调用该接口的程序必须在物理机root用户下运行。

调用示例

```
int ret;
//禁用网卡
ret = dsmi_hot_reset_soc(0xff);
if(ret != 0) {
    //todo
    return ret;
}
//循环查询设备上线状态，给已经上线设备使能网卡
…
```

8.2.3 dsmi_pcie_hot_reset

函数原型

int dsmi_pcie_hot_reset(int device_id)

功能说明

昇腾AI处理器带内复位接口，用于发起昇腾AI处理器复位。该接口调用成功表示复位命令执行完毕，并不代表芯片处于可用状态，可通过获取芯片健康状态等接口（如：dsmi_get_device_health）来确保芯片处于可用状态。

昇腾310 AI处理器场景下，该接口仅支持PCIe标卡，支持通过创建子进程并行复位所有芯片。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围：0~63，当前实际支持的设备号，通过dsmi_list_device接口获取。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码**。

异常处理

无。

约束说明

调用该接口的程序必须在物理机的root用户下运行；若在物理机的非root用户，或在容器下运行，则返回权限错误。

调用该接口前，请停掉昇腾AI处理器上的所有业务。

调用示例

```
pid_t pid;
int ret;
pid = fork(); //创建子进程运行hot_reset
if (pid < 0)
    printf("fork error\n");
else if (pid == 0) {
    ret = dsmi_pcie_hot_reset(0);
    if (ret != 0) {
    //todo: 记录日志
    return ret;
    }
...
```

8.2.4 dsmi_rescan_soc

函数原型

int dsmi_rescan_soc(int device_id)

功能说明

对指定昇腾AI处理器重新扫描。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

调用该接口的程序必须在物理机root用户下运行。

调用示例

```
int ret = 0;
ret = dsmi_rescan_soc(0);
if(ret != 0) {
    //todo
    return ret;
}
...
```

8.2.5 dsmi_get_device_boot_status

函数原型

int dsmi_get_device_boot_status(int device_id, enum dsmi_boot_status *boot_status)

功能说明

获取昇腾AI处理器系统的启动状态。

昇腾310 AI处理器场景下，该接口只支持PCIe标卡和200EP场景。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。

boot_status	输出	昇腾AI处理器的启动状态：
		enum dsmi_boot_status {
		DSMI_BOOT_STATUS_UNINIT = 0, /*未初始化*/
		DSMI_BOOT_STATUS_BIOS, /* BIOS启动中*/
		DSMI_BOOT_STATUS_OS, /* OS启动中*/
		DSMI_BOOT_STATUS_FINISH, /*启动完成*/
		DSMI_SYSTEM_START_FINISH = 16 /* dsmi服务进程启动完成 */
		};

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

无。

调用示例

```
int ret = 0;
enum dsmi_boot_status boot_status = 0;

ret = dsmi_get_device_boot_status (0, &boot_status);
if(ret != 0) {
    //todo
    return ret;
}
...
```

9 网关地址管理

9.1 dsmi_get_gateway_addr

函数原型

int dsmi_get_gateway_addr (int device_id, int port_type, int port_id, ip_addr_t *gtw_address)

功能说明

查询网关地址。

昇腾310 AI处理器MiniRC场景下，不支持该接口。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。
port_type	输入	指定网口类型。 昇腾310 AI处理器场景下，取值范围为VNIC：0x00。用户在调用接口时可以调用宏：DSMI_VNIC_PORT (0)。
port_id	输入	指定网口号。 昇腾310 AI处理器场景下为保留字段，取值范围：【0~255】。

参数名	输入/输出	描述

gtw_address	输出&输入	#define DSMI_ARRAY_IPV4_NUM 4
		#define DSMI_ARRAY_IPV6_NUM 16
		
		typedef struct ip_addr {
		union {
		unsigned char ip6[DSMI_ARRAY_IPV6_NUM];
		unsigned char ip4[DSMI_ARRAY_IPV4_NUM];
		} u_addr;
		enum ip_addr_type ip_type; //这是一个输入值
		} ip_addr_t;
		ip_addr_type结构体定义如下：
		enum ip_addr_type {
		/** IPv4 */
		IPADDR_TYPE_V4 = 0U,
		/** IPv6 */
		IPADDR_TYPE_V6 = 1U,
		/** IPv4+IPv6 ("dual-stack") */
		IPADDR_TYPE_ANY = 2U
		};

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码。**

约束说明

只支持IPV4地址。

调用示例

```
int ret = 0;
int device_id = 0;
int port_type = 0;
int port_id = 0;
ip_addr_t gateway_address={0};

ret = dsmi_get_gateway_addr(device_id, port_type, port_id, &gateway_address);
if(ret != 0) {
    //todo
    return ret;
}
...
```

9.2 dsmi_set_gateway_addr

函数原型

int dsmi_set_gateway_addr(int device_id, int port_type, int port_id, ip_addr_t gtw_address)

功能说明

设置网关地址。

昇腾310 AI处理器MiniRC场景下，不支持该接口。

参数说明

参数名	输入/输出	描述
device_id	输入	指定设备号，取值范围由**dsmi_get_device_count**决定。当前实际支持的设备号，通过**dsmi_list_device**接口获取。
port_type	输入	指定网口类型。 昇腾310 AI处理器场景下，取值范围为VNIC：0x00。用户在调用接口时可以调用宏：DSMI_VNIC_PORT (0)。
port_id	输入	指定网口号。 昇腾310 AI处理器场景下为保留字段，取值范围：【0~255】。

gtw_address	输入	`#define DSMI_ARRAY_IPV4_NUM 4`
		`#define DSMI_ARRAY_IPV6_NUM 16`
		
		`typedef struct ip_addr {`
		`  union {`
		`  unsigned char ip6[DSMI_ARRAY_IPV6_NUM];`
		`  unsigned char ip4[DSMI_ARRAY_IPV4_NUM];`
		`  } u_addr;`
		`  enum ip_addr_type ip_type;  //这是一个输入值`
		`} ip_addr_t;`
		ip_addr_type结构体定义如下：
		`enum ip_addr_type {`
		`  /** IPv4 */`
		`  IPADDR_TYPE_V4 = 0U,`
		`  /** IPv6 */`
		`  IPADDR_TYPE_V6 = 1U,`
		`  /** IPv4+IPv6 ("dual-stack") */`
		`  IPADDR_TYPE_ANY = 2U`
		`};`

返回值

类型	描述
int	处理结果，返回0成功，失败返回**错误码**。

异常处理

无。

约束说明

只支持IPV4地址。

调用该接口的程序必须在物理机的root用户下运行，其他场景将会返回错误。

调用示例

```
int ret = 0;
int device_id = 0;
int port_type = 1;
int port_id = 0;
unsigned int gateway_address = 0xC801A8C0; //192.168.1.200
ip_addr_t ip_gateway_address = {0};

memcpy(&(ip_gateway_address.u_addr.ip4[0]),&gateway_address,4);
ret = dsmi_set_gateway_addr(device_id, port_type, port_id,ip_gateway_address);
if(ret != 0) {
    //todo
    return ret;
}
...
```


10 不支持的接口

Ascend310如下接口不支持。当调用如下接口时，系统将返回不支持的提示信息。

接口名称	功能说明
dsmi_get_aicpu_info	查询AICPU的个数、最大运行频率、当前运行频率和利用率。
dsmi_get_hbm_info	查询hbm的频率、容量、利用率信息。
dsmi_get_llc_perf_para	查询LLC性能参数，包括LLC读命中率、写命中率和吞吐量。
dsmi_get_network_health	查询RoCE网卡的IP地址的联通状态。
dsmi_set_sec_revocation	实现密钥吊销功能，当前只支持Soc二级密钥吊销。
dsmi_get_can_status	查询指定CAN控制器状态信息。
dsmi_get_ufs_status	获取UFS的状态信息。
dsmi_get_sensorhub_status	查询SensorHub状态信息。
dsmi_get_lp_status	获取低功耗的状态信息。
dsmi_get_hiss_status	获取HISS子系统的状态信息。
dsmi_set_power_state	设置系统模式。
dsmi_get_gpio_status	获取gpio的状态信息。
dsmi_get_sochwfault	获取芯片的硬件错误信息。
dsmi_get_safetyisland_status	获取safetyisland的自身的状态信息。
dsmi_register_fault_event_handler	注册故障回调接口。
dsmi_get_sensorhub_config	获取SensorHub配置信息。
dsmi_create_capability_group	创建算力配置信息。
dsmi_delete_capability_group	删除算力分组配置信息。

接口名称	功能说明
dsmi_get_capability_group_info	获取算力分组配置信息。
dsmi_set_upgrade_attr	通过设置升级接口属性DSMI_UPGRADE_ATTR来调用相应的升级接口功能。
dsmi_get_total_ecc_isolated_pages_info	获取历史产生的ECC多bit错误数量以及隔离地址数。
dsmi_clear_ecc_isolated_statistics_info	清除FLASH里面记录的历史ECC多Bit错误数量和地址。
dsmi_get_device_ndie	获取指定设备的NDIE ID。
dsmi_create_vdevice	创建虚拟设备。
dsmi_destroy_vdevice	删除虚拟设备。
dsmi_check_partitions	检查当前环境的分区名称、分区大小是否与配置文件中配置的一致。
dsmi_get_chip_count	获取芯片数量。
dsmi_list_chip	获取芯片列表。
dsmi_get_device_count_from_chip	获取指定芯片的device数量。
dsmi_get_device_from_chip	获取指定device对应的芯片编号。

11 返回码

DSMI错误码	值	含义
DRV_ERROR_NONE	0	执行成功。
DRV_ERROR_NO_DEVICE	1	设备不存在。
DRV_ERROR_INVALID_DEVICE	2	无效的device id。
DRV_ERROR_INVALID_VALUE	3	无效的值
DRV_ERROR_INVALID_HANDLE	4	无效句柄，没有地方使用。
DRV_ERROR_INNER_ERR	7	内部错误。
DRV_ERROR_PARA_ERROR	8	参数错误。
DRV_ERROR_NOT_EXIST	11	外设不存在。
DRV_ERROR_BUSY	13	设备繁忙。
DRV_ERROR_WAIT_TIMEOUT	16	响应超时。
DRV_ERROR_IOCRL_FAIL	17	ioctl，返回失败。
DRV_ERROR_SEND_MESG	27	消息发送失败。
DRV_ERROR_OVER_LIMIT	37	超规格
DRV_ERROR_OPER_NOT_PERMITTED	46	权限错误。
DRV_ERROR_TRY_AGAIN	51	尚未就绪，请重试。
DRV_ERROR_MEMORY_OPT_FAIL	58	内存操作失败。
DRV_ERROR_PARTITION_NOT_RIGHT	86	分区一致性校验，发现存在不一致的分区

DSMI错误码	值	含义
DRV_ERROR_RESOURCE_OCCUPIED	87	资源（设备、进程等）被占用，无法使用该资源。 例如：容器中只要存在一个或以上的设备被其他容器占用，该容器下所有设备不可用，返回该错误码。
DRV_ERROR_RESUME	89	唤醒失败。
DEV_ERROR_BIST_HW_ERR	91	BIST测试硬件错误
DEV_ERROR_BIST_SW_ERR	92	BIST测试软件错误
DRV_ERROR_NOT_SUPPORT	0xfffe	命令码/功能不支持。
其他	未知非0值	表示其他错误。

12 调用示例

各接口说明处的“调用示例”是一个代码示例片段，此处以调用dsmi_get_device_health接口为例，给出完整的调用示例，说明需要include的头文件（dsmi_common_interface.h、ascend_hal_error.h）、调用接口的逻辑、返回值的打印等。

昇腾310 AI处理器场景下，您需要参见《CANN软件安装指南》部署环境，部署完成后，从环境的“/usr/local/Ascend/driver/include”目录下获取dsmi_common_interface.h、ascend_hal_error.h文件。

说明

/usr/local/Ascend表示软件包的默认安装目录，需根据实际情况替换。

示例代码文件 get_device_health.c

```
#include <stdio.h>
#include <stdlib.h>
#include <getopt.h>
#include <unistd.h>
#include "dsmi_common_interface.h"
int main(int argc, char *argv[])
{
    int ret = 0;
    int opt = 0;
    int test_case_num = 0;
    unsigned int phealth = 0;
    char optstring[20] = "p:s:gr";
    static struct option long_options[] =
    {
        {"process", 1, NULL, 'p'},
        {0, 0, 0, 0}
    };
    while ((opt = getopt_long(argc, argv, optstring, long_options, NULL)) != -1)
    {
        switch (opt)
        {
            case 'p':
            {
                if (argc < 3)
                {
                    printf("process test the para num: %d test_error. Input num should not be smaller than 3.test_fail \n",argc);
                    return -1;
```

```
            }
            test_case_num = strtol(argv[2], NULL, 10);
            switch (test_case_num)
            {
                case 1:
                {
                    printf("begin query health\n");
                    ret = dsmi_get_device_health(0, &phealth);
                    if (ret)
                    {
                        printf("call dsmi_get_device_health fail, ret = %d\n", ret);
                        return -1;
                    }
                    else
                    {
                        /*phealth type is unsigned int, printf value should use %u*/
                        printf("dsmi_get_device_health success,phealth:%u.\n", phealth);
                    }
                    break;
                }
                default:
                    break;
            }
        }
    }
    return 0;
}
```

Ascend EP 场景下使用调用示例

步骤1 以HwHiAiUser用户将get_device_health.c、dsmi_common_interface.h传到Host侧服务器的同一个目录下。

步骤2 以HwHiAiUser用户登录到Host侧服务器。

步骤3 执行如下命令，编译get_device_health.c中的代码，生成可执行文件get_device_health。

```
gcc get_device_health.c /usr/local/Ascend/driver/lib64/libdrvdsmi_host.so -L. -o get_device_health
```

说明

- /usr/local/Ascend表示Driver组件的默认安装路径，请根据实际情况替换。

步骤4 执行可执行文件get_device_health。

```
./get_device_health -p 1
```

----结束

