OneNET 行业开发平台

设备 SDK 用户手册

版本号：V1.4

拟制：_____ 日期：

审核：_____ 日期：

批准：_____ 日期：

修订记录

版本号	修订日期	修订内容	负责人
V1.0	2019.08.27	新建	武金磊、钱平、刘长亮
V1.1	2019.09.11	1、修改第 3.1、3.3 章节变量命名 2、“驱动接口”章节读设备 imei/模组 imei/设备 sn 接口移到“应用接口”章节 3、“低功耗接口”章节新增读取 psm 参数接口 4、“应用接口”章节新增读模组 imsi、读取网络参数、获取 APN、系统重启接口	刘长亮
V1.2	2019.09.12	1、添加 3.1.1 章节 2、添加 3.1.2 章节	邓俊
V1.3	2019.09.26	1、添加初始化前 hook； 2、更新 API 接口；	钱平
V1.4	2020.1.2	1、更新 3.1.1 章节 2、更新 3.1.2 章节	曹小英

目 录

1. 概述

本文档用于指导用户基于应用 SDK 进行行业开发平台产品需求的开发。

本文档介绍了应用 SDK 的整体框架及流程，并对用户开发流程及 API 做了详细说明。

2. SDK 说明

OneNET 行业开发平台基于 OneNET 平台屏蔽海量设备各式各样的南向接入协议，专注于应用数据的解析和下发，形成一套统一的行业开发平台解决方案。

基于该方向考虑，设备端需要应用设计平台化，形成一套脱离具体行业场景的应用 SDK，用户通过添加配置具体产品的资源及实现即可实现产品开发。

该应用 SDK 具备可配置性（资源及操作在平台可灵活配置并自动生成）、可适配性（兼容多款模组或者芯片）、可嵌入性（提供 Hook 函数以植入用户逻辑）并且支持 OpenCPU 和外接 MCU 两种产品模式。

2.1 框架设计

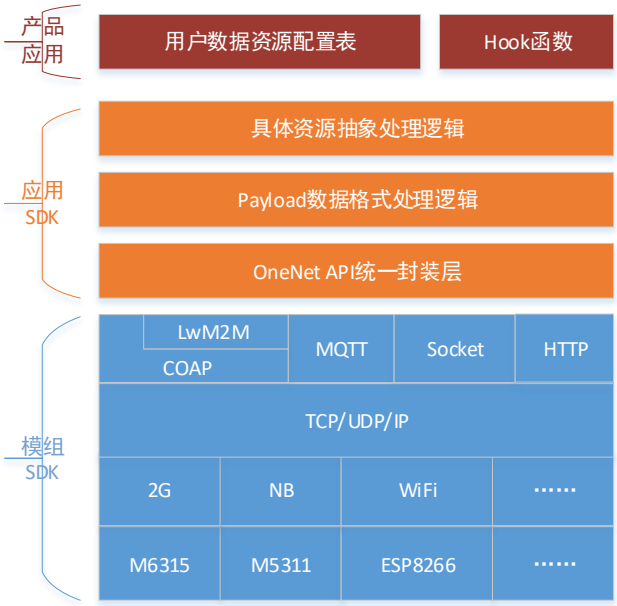

图 2-1 框架设计图

应用 SDK 的平台化设计如图 2-1 所示：

1) **可配置性：**应用 SDK 需严格采用代码和数据分离的设计思路，应用 SDK 只实现设备资源的抽象逻辑，一切与具体产品相关的用户数据均放置在资源配置表中，由用户根据产品需求自行配置。资源配置表核心功能是

配置产品资源以及资源句柄。

2) **适配性：**提供 OneNET 接入的统一 API，统一封装各模组/芯片的南向对接协议，以适配不同的硬件平台。对于不同的模组及南向协议，对用户都是透明的。

3) **可嵌入性：**针对用户需要植入产品特色逻辑的需求，如硬件自检，硬件初始化等，在系统运行过程的不同环节，提供若干 Hook 函数指针。用户可根据自己的需求将具体逻辑实现注册到应用 SDK 的 Hook 函数指针中。

2.2 流程设计

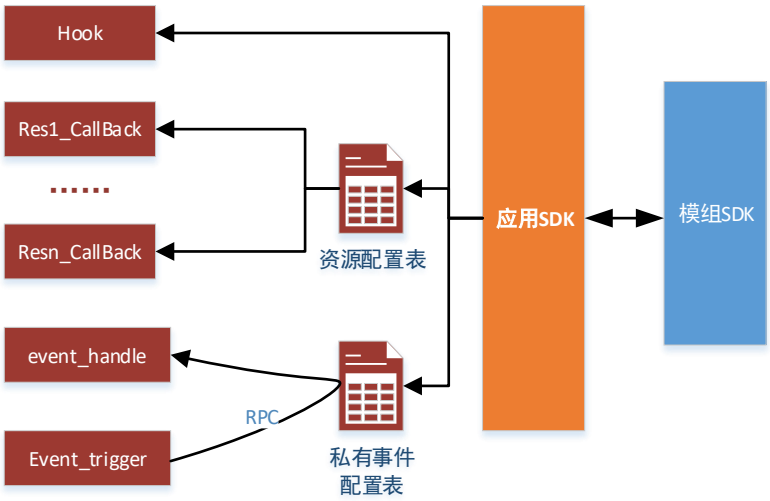

图 2-2 流程图

如图 2-2 所示，应用 SDK 的用户实现只需要实现 Hook 函数，资源回调函数以及用户私有事件句柄即可实现产品的开发。

除了基本的资源处理外，应用 SDK 为方便用户的产品开发，采用 HooK 点的方式允许用户在主体逻辑中加入自己的处理逻辑；提供了私有事件配置表，允许用户根据产品需求主动触发产品私有事件逻辑，整个过程参数透明传输，类似用户进行了 RPC 调用。

1) **Hook 函数：**应用 SDK 共提供了 5 个 Hook 点，分别用于 OneNET 上线前、OneNET 上线后、休眠前、唤醒后以及系统级错误调用。

2) **资源句柄：**平台会根据用户的资源选择，自动渲染资源句柄空函数，用户进行填充。

3) **用户私有事件函数：**平台会生成私有事件配置表，用户可根据产品需求添加用户事件 ID（范围 301~400）及实现函数，然后选择事件触发点进行参数传递和触发，实现基于线程的 RPC 调用。

3. 用户操作

3.1. 平台配置

3.1.1. 页面配置

在行业开发平台完成产品创建和功能点定义，定义好产品功能后，行业开发平台会根据定义的功能自动生成设备端的 SDK，每次功能变更后需要重新下载 SDK。

1) **产品创建**

登录行业开发平台后，选择设备生产的角色，在首页可进行产品创建，产品类别请根据您实际的类别选择，不同类别有不同的推荐功能点；目前行业开发平台支持搭载 NB、WiFi 和 2G 模组的设备接入，以 TLV 格式传输。

图 3-1 创建产品

2) **功能点配置**

进入产品详情后，点击“产品开发”模块，根据您的需求定义设备功能点，功能点分为基础功能点、组合功能点、固定上报功能点三大类。具体配置方式如下：

a) 基础功能点：

基础功能点根据功能类型分为设备功能和系统功能，如图 3-2 和图 3-3 所示。

设备功能为设备所需的属性和事件数据；系统功能为平台提前制定的功能并能对这些功能提供服务，如固件版本，将用于固件升级，如故障数，将用于故障统计，输出报表等。

图 3-2 基础功能点-设备功能

图 3-3 基础功能点-系统功能

设备功能定义提供“标准功能”和“添加自定义功能”两类，用户可以自行选择行业开发平台提前定义好的标准功能点。当行业开发平台定义的标准功能点不能满足您的需求时，您可以通过添加自定义功能来满足差异化需求，如图 3-4 所示。系统功能只能选择平台提前定义好的功能。

图 3-4 设备功能-添加自定义功能

b) 组合功能点：

用户希望将多个基础功能点组合，进行上报或者下发时需要新增组合功能点，同一个组合功能点中包含属性或事件中的一类。在选择组合功能点的传输类型后，筛选后的功能点可以在页面的穿梭框中进行调整。如图 3-5 所示。

图 3-5 组合功能点

c) 固定上报功能点：

固定上报功能点指的是设备每次上报时都会上报一次的全局功能点。固定上

报功能点添加方式与基础功能点类似，既可以从平台定义好的标准功能点中选择，也可以自定义。目前仅提供消息 id 和上报时间两个功能。

图 3-6 固定上报

3.1.2 SDK 下载

完成功能定义后需要下载编译环境和设备 SDK，每次功能点变更时都需要重新下载设备 SDK，设备开发完成，上电后可以在图 3-7 所示页面进行功能调试。

图 3-7 设备开发调试页面

完成产品功能定义后，点击下一步进入到设备开发测试环节，用户可以在该页面中下载 SDK 进行本地调试。

点击下载“SOC SDK”，即行业开发平台根据“产品功能定义”环节中的功能点自动生成的 SDK，SOC SDK 文件目录如图 3-8 所示，下载的压缩文件名中的数字表示产品 ID。

点击下载“SOC 编译环境”，编译环境文件结构如图 3-9 所示。

用户只需将 SOC SDK 中的 “cmiot_user.c”和“ cmiot_user.h” 两个文件

复制到编译环境文件夹中 user 文件夹下，替换掉编译环境文件中原有的 cmiot_user.c 和 cmiot_user.h 文件即可进行开发。

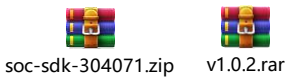

图 3-8 SOC SDK 文件目录

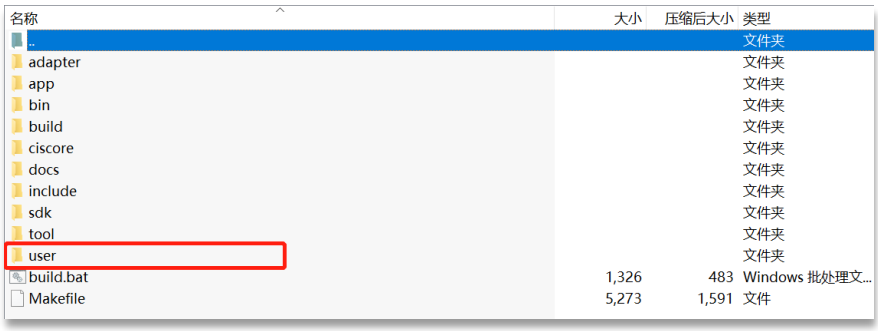

图 3-9 编译环境文件目录

3.1.3 资源配置表

cmiot_user.c 文件中，资源配置表需要平台根据基础功能，组合功能以及固定功能生成单独生成。具体规则如下：

1) 资源功能配置表

平台配置生成资源名，FuncID 以及数据类型，如图 3-10 所示数组。

```c
#include "cmiot_pub.h"


/* 普通功能点数组 */
static deviceFuncHandle_t devFuncList[] = {
    FUNC_HANDLE_UP(product_key_id, 0xe801, STRING),
    FUNC_HANDLE_UP(device_imsi_id, 0xe802, STRING),
    FUNC_HANDLE_UP(psk_auth_code,  0xe803, STRING),
    FUNC_HANDLE_UP(envhumidity, 28674, INT),
    FUNC_HANDLE_BOTH(targethumidity, 28675, INT),
    FUNC_HANDLE_BOTH(windspeed, 20484, ENUM),
    FUNC_HANDLE_BOTH(fog, 20485, ENUM),
    FUNC_HANDLE_BOTH(light, 12297, BOOL),
    FUNC_HANDLE_UP(signal_strength, 32779, FLOAT),
    FUNC_HANDLE_UP(software_version, 57356, STRING),
    FUNC_HANDLE_DOWN(reset, 57361, STRING),
    FUNC_HANDLE_UP(alarm_num, 24595, INT),
    FUNC_HANDLE_UP(up_success_rate, 32788, FLOAT),
    FUNC_HANDLE_BOTH(switch, 12309, BOOL)
};
```

图 3-10 资源功能配置表

2) 组合功能配置表

平台配置生成资源名，FuncID 以及数据类型，并根据 FuncID（十进制）生成数组名字，并生成注释，如图 3-11 所示。

其中“measurement”表示自定义的组合功能点字段名称；“46336”表示自定义的组合功能点 ID 号；“28674、28675、20484 和 20485”表示该组合功能点钟包含的 4 个基础功能点 ID 号。

```c
/* 组合功能点数组 */
static uint16_t GROUP_FUNC_NAME(hewu_reg,42240)[] = {
    0xe801,
    0xe802,
};
static uint16_t GROUP_FUNC_NAME(measurements,46336)[] = {
        28674,
        28675,
        20484,
        20485
};

/*组合功能点数组索引表*/
static groupFuncIndex_t groupFuncIndex[] = {
    GROUP_FUNC_INDEX(42240, ARRAY_SIZE(GROUP_FUNC_NAME(hewu_reg,42240)), GROUP_FUNC_NAME(hewu_reg,42240)),
        GROUP_FUNC_INDEX(46336, ARRAY_SIZE(GROUP_FUNC_NAME(measurements,46336)), GROUP_FUNC_NAME(measurements,46336))
};
```

图 3-11 组合功能配置表

注意：如果资源功能配置表有相同的资源句柄，则不再重复生成相同句柄。

3) 固定上报功能配置表

平台配置生成资源名，FuncID 以及数据类型，如图 3-12 所示数组。任何消息上报，都会在结尾附带该数据资源值。

```c
/* 固定功能点数组 */
static deviceFuncHandle_t devTailList[] = {
        FUNC_HANDLE_UP(messageid, 63232, STRING),
        FUNC_HANDLE_UP(messagetime, 63233, STRING)
};
```

图 3-12 固定上报功能配置表

注意：固定上报功能配置隐藏协议版本号，不由用户配置，程序隐藏自带，以便平台根据不同协议版本进行解析。

4) 用户私有事件配置表：

平台生成资源配置表，由用户根据产品需求在本地添加，如图 3-13 所示。

```
C cmiot_user.c ×
user > C cmiot_user.c
52          GROUP_FUNC_INDEX(46336, ARRAY_SIZE(GROUP_FUNC_NAME(measurements,46336)), GROUP_FUNC_N
53      };
54
55
56      /* 用户私有事件配置表，注：只有在登录onenet后可用 */
57      static usrEventHandle_t usrEventHandle[] = {
58          USR_EVENT_HANDLE(CMIOT_EVENT_USER_NOTIFY_RESULT, notify_result),
59      };
60
```

图 3-13 用户私有事件配置函数

3.2 本地默认配置

cmiot_user.h 文件中包含了一些本地默认配置参数，如有需要，用户可以自行配置。

```
/*  调试串口默认为 uart0 */
#define CMIOT_HAL_UART_DBG          HAL_UART_0

/*  注册和物重试次数限制  */
#define HEWU_REGIST_CNT_MAX         5

/*注册和物一次超时时间(ms) */
#define HEWU_REGIST_TIMEOUT_MS          5000

/*  和物服务器 IP 地址  */
#define HEWU_COAP_SERVER_HOST           "183.230.40.32"
#define HEWU_COAP_SERVER_PORT           26009

/*  驻网失败重试次数限制  */
#define CMIOT_CELL_REG_RETRY_MAX            3

/*  上线 OneNET 失败重试次数限制  */
#define CMIOT_ONENET_ONLINE_RETRY_MAX           3

/*  连接 OneNET 平台的保活时间(单位 s) */
#define CMIOT_ONENET_OPEN_LIFETIME              (60 * 60 * 25)

/* UPDATE 的保活时间(单位 s) */
```

```
#define CMIOT_ONENET_UPDATE_LIFETIME          (200)

/* UPDATE 的发送间隔时间(单位 ms) */
#define CMIOT_ONENET_UPDATE_PERIOD            (60*1000)

/*  上报最大重传次数  */
#define CMIOT_ONENET_NOTIFY_MAX_RETRANS       3
```

3.3 开发说明

3.3.1 Hook 开发

OneNET 行业开发平台设备 SDK 共包含 6 个 Hook 函数（cmiot_user.c 文件），供用户实现相关业务逻辑。如表 3-1 所示。

序号	Hook 函数	说明	功能描述
1	func_hook_init_ahead	系统初始化前	该 Hook 函数在系统初始化前，即主线程启动处，用户可在该函数中实现上电后需要即时处理的业务
2	func_hook_online_ahead	OneNET 上线前	该 Hook 函数在系统初始化后、OneNET 上线流程前调用，用户可在该函数中进行外围设备以及其他业务相关资源的初始化等工作
3	func_hook_online_after	OneNET 上线后	该 Hook 函数在 OneNET 上线成功后调用，用户可在该函数中实现功能点上报等业务
4	func_hook_sleep_ahead	系统休眠前	该 Hook 函数在系统进入深度休眠（PSM 模式）前调用，用户可在该函数中完成休眠前的清场等工作
5	func_hook_wakeup_extin	系统被外部引脚唤醒后	系统在深度休眠（PSM 模式）中若被外部引脚唤醒，将调用该 Hook 函数。该函数为中断回调函数，不能在其中执行耗时操作
6	func_hook_sys_err	发生系统级错误	系统在驻网重试次数、上线重试次数达到 cmiot_def.h 中定义的最大重试次数时（即 CMIOT_CELL_REG_RETRY_MAX 和 CMIOT_ONENET_ONLINE_RETRY_MAX，该值可由用户配置），以及 FOTA 发生异常时，将调用该 Hook 函数；用户可根据不同错误类型（即 SYS_ERR_CREG_ERR、SYS_ERR_ONLINE_ERR、SYS_ERR_FOTA_ERR）进行相应处理。

3.3.2 资源句柄开发

平台会根据用户的资源配置，自动渲染出资源句柄的空函数，由用户进行填充。资源句柄可分为两类，读资源句柄和写资源句柄。

1) 读资源句柄

读资源句柄的原型如下，val 为读取缓存的起始地址，buf_len 为读取的数据长度。

```
int (*deviceFuncGet_t)(uint8_t *val, uint16_t bufLen);
```

对于字符型和透传型功能点，渲染出的读资源句柄空函数如下：

```
int func_strtype_and_buftype_get(uint8_t *val, uint16_t bufLen)
{
   char *value = (char *)val;
   /* 将获取的字符串或透传型数据写入 value 指向的地址，并返回字符串或透传型数据长度，长度不能超过 buf_len，否则返回 0 */

   return len;
}
```

对于其他数据类型的功能点，渲染出的读资源句柄空函数如下：

```
int func_othertype_get(uint8_t *val, uint16_t bufLen)
{
      type value;
      /* 请填入功能点值的获取逻辑，并将值赋给变量 value */

      return set_by_binary(&value, val, bufLen, sizeof(type));
}
```

2）写资源句柄

写资源句柄的原型如下，val 为写入数据的起始地址，val_len 为写入的数据长度。

```
returnCode_e (*deviceFuncSet_t)(uint8_t *val, uint16_t valLen);
```

对于字符型功能点，渲染出的写资源句柄空函数如下：

```
returnCode_e func_strtype_set(uint8_t *val, uint16_t valLen)
{
      val[valLen] = '\0';
      /** 根据变量 val 的值，填入下发控制逻辑 */
```

```
    /** 成功返回 RETURN_CODE_OK，失败返回 RETURN_CODE_FAILED */
    return RETURN_CODE_OK;
}
```

对于透传型功能点，渲染出的写资源句柄空函数如下：

```
returnCode_e func_buftype_set(uint8_t *val, uint16_t valLen)
{
    /* 根据变量 val 与 valLen 的值，填入下发控制逻辑 */

    /* 成功返回 RETURN_CODE_OK，失败返回 RETURN_CODE_FAILED */
    return RETURN_CODE_OK;
}
```

对于其他功能点，渲染出的写资源句柄空函数如下：

```
returnCode_e func_othertype_set(uint8_t *val, uint16_t valLen)
{
    type value = *(type *)(val);
    /* 根据变量 val 与 valLen 的值，填入下发控制逻辑 */

    /* 成功返回 RETURN_CODE_OK，失败返回 RETURN_CODE_FAILED */
    return RETURN_CODE_OK;
}
```

3.3.3 用户 RPC 开发

用户可根据产品需求，在私有事件配置表中添加用户事件 ID（范围 301~400）及相应实现函数，然后选择事件触发点进行参数传递和触发，实现基于线程的 RPC 调用。

平台生成的私有事件配置表如下：

```
/* 用户私有事件配置表，注：只有在登录 onenet 后可用 */
static usrEventHandle_t usrEventHandle [] = {
    USR_EVENT_HANDLE(CMIOT_EVENT_USER_NOTIFY_RESULT,
notify_result),
    USR_EVENT_HANDLE(302, example)
};
```

其中，CMIOT_EVENT_USER_NOTIFY_RESULT 为固定事件（事件 ID 为 301），用来通知用户功能点上报的结果。该事件的处理函数如下，用户可在该函数中填入功能点上报成功或失败后的处理逻辑。

```
returnCode_e func_notify_result_exec(void *val, uint16_t bufLen)
{
```

```
    notifyPara_t notifyRes = *(notifyPara_t *)val;
    LOG_PRINT(MODULE_TYPE_USER, LOG_LEVEL_INFO, "func %d
notify %s, respId: %d\r\n", notifyRes.funcId, notifyRes.result ? "succ" : "failed",
notifyRes.respId);
    /* 请填入上报结果事件的具体处理逻辑 */

    return RETURN_CODE_OK;
}
```

302 为 RPC 调用示例，用户可调用以下接口发送用户事件。其中，cmiot_user_event_send_from_isr 可以在中断服务程序中调用。

```
/**
 *  \brief 发送用户事件
 *
 *  \param [in] eventType 用户事件 ID
 *  \param [in] data      要发送的数据
 *  \param [in] len       要发送的数据长度
 *  \param [in] waitTime  队列满时的等待时长（单位 ms）
 *  \return RETURN_CODE_OK：成功 其他：失败
 */
returnCode_e cmiot_user_event_send(cmiotEventType_e eventType, void *data,
uint32_t len, uint32_t waitTime);

/**
 *  \brief 在中断中发送用户事件
 *
 *  \param [in] eventType 用户事件 ID
 *  \param [in] data      要发送的数据
 *  \param [in] len       要发送的数据长度
 *  \return RETURN_CODE_OK：成功 其他：失败
 */
returnCode_e cmiot_user_event_send_from_isr(cmiotEventType_e eventType, void
*data, uint32_t len);
```

队列接收到 302 事件时，将调用 func_example_exec，该函数需由用户实现，完成该事件的 RPC 调用。

```
returnCode_e func_example_exec(void *val, uint16_t buf_len)
{
    /* 请填入用户事件的具体处理逻辑 */
```

```
    return RETURN_CODE_OK;
}
```

3.3.4 主动上报

用户可调用以下接口发起 notify，将功能点上报到 OneNET 平台。该接口需传入两个参数，功能点 ID（普通功能点或组合功能点）和响应 ID。（注：OneNET 平台要求，在 247 秒内，每次调用该接口传入的响应 ID 应不重复，否则 OneNET 将视为重复消息丢弃。）

功能点上报结果将通过 CMIOT_EVENT_USER_NOTIFY_RESULT 事件通知用户，用户可在该事件 RPC 函数中进行后续逻辑处理。

```
/**
 *  \brief 上报功能点
 *
 *  \param [in] funcId 功能点 ID
 *  \param [in] respId 响应 ID
 *  \return RETURN_CODE_OK：成功 其他：失败
 */
returnCode_e cmiot_user_onenet_notify (uint16_t funcId, uint16_t respId);

/**
 *  \brief 在中断服务程序中上报功能点
 *
 *  \param [in] funcId 功能点 ID
 *  \param [in] respId 响应 ID
 *  \return RETURN_CODE_OK：成功 其他：失败
 */
returnCode_e cmiot_user_onenet_notify_fromisr (uint16_t funcId, uint16_t respId);
```

3.4 API 接口

3.4.1 驱动接口

GPIO 接口详细请参照代码示例及 hal_gpio.h。

SDK 支持 56K Flash 空间，地址范围 0x02000 - 0x0FFFF。注意，擦除函数会擦除指定地址区域所涉及的 flash 块，以 4K 字节为单位。

3.4.1.1 cmiot_hal_gpio_init

原型	returnCode_e cmiot_hal_gpio_init(halGpioConfig_t *config);
描述	GPIO 口初始化函数
参数	config gpio 口的配置数据
返回值	returnCode_e, 具体含义参见 returnCode_e 枚举类型定义

3.4.1.2 cmiot_uart_open

原型	void cmiot_uart_open(hal_uart_port_t uart_port,hal_uart_baudrate_t baud,hal_uart_callback_t user_cb);
描述	串口初始化函数
参数	uart_port 要初始化的串口号 baud 波特率 user_cb 数据接收回调函数
返回值	无

3.4.1.3 cmiot_uart_open_custom

原型	void cmiot_uart_open_custom(hal_uart_port_t uart_port,hal_uart_config_t *l_config,hal_uart_callback_t user_cb);
描述	串口初始化函数
参数	uart_port 要初始化的串口号 l_config 详细的 UART 配置结构体指针 user_cb 数据接收回调函数
返回值	无

3.4.1.4 cmiot_uart_open

原型	void cmiot_uart_open(hal_uart_port_t uart_port,hal_uart_baudrate_t baud,hal_uart_callback_t user_cb);
描述	串口初始化函数
参数	uart_port 要初始化的串口号 baud 波特率 user_cb 数据接收回调函数
返回值	无

3.4.1.5 cmiot_uart_receive

原型	int cmiot_uart_receive(hal_uart_port_t uart_port,unsigned char * pbuf,unsigned int len);
描述	从指定串口的缓冲区获取串口数据
参数	uart_port 串口号 pbuf 保存数据的指针 len 欲接收的数据长度
返回值	0：成功 -1：失败

3.4.1.6 cmiot_uart_send

原型	int cmiot_uart_send(hal_uart_port_t uart_port,unsigned char * pbuf,unsigned int len);
描述	串口发送函数
参数	uart_port 串口号 pbuf 数据指针 len 要发送的数据长度
返回值	0：成功 -1：失败

3.4.1.7 cmiot_uart_get_available_receive_bytes

原型	int cmiot_uart_get_available_receive_bytes(hal_uart_port_t uart_port);
描述	获取当前串口缓冲区接收到的数据长度
参数	uart_port 串口号
返回值	缓冲区收到的数据长度

3.4.1.8 cmiot_uart_close

原型	void cmiot_uart_close(hal_uart_port_t uart_port)
描述	串口关闭函数
参数	uart_port 串口号
返回值	无

3.4.1.9 cmiot_user_flash_write

原型	returnCode_e cmiot_user_flash_write(uint32_t addr, unsigned char *wData, uint16_t len, flashWriteType_e wType);

描述	写 flash
参数	Addr 地址 wData 数据 len 长度 wType 写类型
返回值	returnCode_e，具体含义参见 returnCode_e 枚举类型定义

3.4.1.10 cmiot_user_flash_read

原型	returnCode_e cmiot_user_flash_read(uint32_t addr, unsigned char *rData, uint16_t len);
描述	读取 flash
参数	Addr 读取地址 rData 数据 len 长度
返回值	returnCode_e，具体含义参见 returnCode_e 枚举类型定义

3.4.1.11 cmiot_user_flash_erase

原型	returnCode_e cmiot_user_flash_erase (uint32_t addr, uint16_t len);
描述	擦除指定地址范围所在的 FLASH 分区(4K 整数倍)
参数	Addr 地址 Len 长度
返回值	returnCode_e，具体含义参见 returnCode_e 枚举类型定义

3.4.2 软件定时器接口

定时器接口及其使用方法跟 FREERTOS 一致，函数定义请参照头文件 timers.h

3.4.3 低功耗接口

3.4.3.1 cmiot_lock_sleep

原型	void cmiot_lock_sleep(void);
描述	关闭模组浅睡眠，执行了此函数之后，设备将无法进入浅睡眠以及 PSM 等各种省电模式
参数	无

返回值	无

3.4.3.2 cmiot_unlock_sleep

原型	void cmiot_unlock_sleep(void);
描述	开启浅睡眠，与关闭浅睡眠函数成对使用，开启之后，设备就会按照正常流程，在该进入省电模式的时候进入省电模式
参数	无
返回值	无

3.4.3.3 cmiot_sleep_create

原型	returnCode_e cmiot_sleep_create(cmiot_sleep_callback_t callback , uint32_t sleepTime);
描述	开启睡眠定时
参数	Callback 回调函数 sleepTime 睡眠时间(单位 ms)
返回值	returnCode_e，具体含义参见 returnCode_e 枚举类型定义

3.4.3.4 cmiot_sleep_delete

原型	returnCode_e cmiot_sleep_delete(void);
描述	删除睡眠定时
参数	无
返回值	returnCode_e，具体含义参见 returnCode_e 枚举类型定义

3.4.3.5 cmiot_device_psm_set

原型	returnCode_e cmiot_device_psm_set(char *tau, char *actTime);
描述	设置 psm
参数	tau tau 值 actTime active time 值
返回值	returnCode_e，具体含义参见 returnCode_e 枚举类型定义

3.4.3.6 cmiot_device_psm_get

原型	returnCode_e cmiot_device_psm_get(ril_power_saving_mode_setting_req_t

	*req);
描述	读取 psm
参数	req psm 参数
返回值	returnCode_e，具体含义参见 returnCode_e 枚举类型定义

3.4.3.7 cmiot_edrx_set

原型	returnCode_e cmiot_edrx_set(int edrxMode, char *edrxValue);
描述	设置 EDRX
参数	mode EDRX 模式，0 或 1，代表关闭或使能 edrxValue EDRX 时间参数
返回值	returnCode_e，具体含义参见 returnCode_e 枚举类型定义

3.4.3.8 cmiot_edrx_get

原型	returnCode_e cmiot_edrx_get(ril_read_eDRX_dynamic_parameters_rsp_t *edrxrdpValue);
描述	读取 EDRX 生效值
参数	edrxrdpValue EDRX 生效参数
返回值	returnCode_e，具体含义参见 returnCode_e 枚举类型定义

3.4.3.9 cmiot_set_rai

原型	returnCode_e cmiot_set_rai(uint8_t raiType);
描述	设置 NBIOT RAI 功能
参数	raiType 0: No information available (or none of the other options apply) 1: TE will send only 1 UL packet and no DL packets expected 2: TE will send only 1 UL packet and only 1 DL packet expected
返回值	returnCode_e，具体含义参见 returnCode_e 枚举类型定义

3.4.3.10 cmiot_ cfun_excute

原型	void cmiot_cfun_excute(int cfun);
描述	配置 cfun 功能
参数	cfun 0: 开启飞行模式；

	1: 关闭飞行模式；
返回值	无

3.4.3.11 cmiot_ cfun_read

原型	void cmiot_cfun_read(int *cfun);
描述	读取 cfun 值
参数	cfun 0:飞行模式开启； 1:飞行模式关闭；
返回值	无

3.4.4 应用接口

3.4.4.1 func_hook_init_ahead

原型	void func_hook_init_ahead(void);
描述	该 Hook 函数在系统初始化前，即主线程启动处，用户可在该函数中实现上电后需要即时处理的业务；
参数	无
返回值	无

3.4.4.2 func_hook_online_ahead

原型	void func_hook_online_ahead(void);
描述	OneNET 上线前 hook 函数，该 Hook 函数在系统初始化后、OneNET 上线流程前调用，用户可在该函数中进行外围设备以及其他业务相关资源的初始化等工作
参数	无
返回值	无

3.4.4.3 func_hook_online_after

原型	void func_hook_online_after(void);
描述	OneNET 上线后 hook 函数，该 Hook 函数在 OneNET 上线成功后调用，用户可在该函数中实现功能点上报等业务
参数	无
返回值	无

3.4.4.4 func_hook_sleep_ahead

原型	void func_hook_sleep_ahead(void);
描述	系统休眠前 hook 函数，该 Hook 函数在系统进入深度休眠（PSM 模式）前调用，用户可在该函数中完成休眠前的清场等工作
参数	无
返回值	无

3.4.4.5 func_hook_wakeup_extin

原型	void func_hook_wakeup_extin(void);
描述	系统被外部引脚唤醒后 hook 函数，系统在深度休眠（PSM 模式）中若被外部引脚唤醒，将调用该 Hook 函数。该函数为中断回调函数，不能在其中执行耗时操作
参数	无
返回值	无

3.4.4.6 func_hook_sys_err

原型	void func_hook_sys_err(sysErrType_e err);
描述	发生系统级错误 hook 函数，系统在驻网重试次数、上线重试次数达到 cmiot_user.h 中定义的最大重试次数时（即 CMIOT_CELL_REG_RETRY_MAX 和 CMIOT_ONENET_ONLINE_RETRY_MAX，该值可由用户配置），以及 FOTA 发生异常时，将调用该 Hook 函数；用户可根据不同错误类型（即 SYS_ERR_CREG_ERR、SYS_ERR_ONLINE_ERR、SYS_ERR_FOTA_ERR）进行相应处理。
参数	typedef enum sysErr { SYS_ERR_CREG_ERR, SYS_ERR_ONLINE_ERR, SYS_ERR_MAX, } sysErrType_e;
返回值	无

3.4.4.7 cmiot_user_event_send

原型	returnCode_e cmiot_user_event_send(cmiotEventType_e eventType, void *data, uint32_t len, uint32_t waitTime);
描述	发送用户事件
参数	eventType 用户事件 ID data 要发送的数据 len 要发送的数据长度 waitTime 队列满时的等待时长（单位 ms）
返回值	returnCode_e，具体含义参见 returnCode_e 枚举类型定义

3.4.4.8 cmiot_user_event_send_from_isr

原型	returnCode_e cmiot_user_event_send_from_isr(cmiotEventType_e eventType, void *data, uint32_t len);
描述	在中断中发送用户事件
参数	eventType 用户事件 ID data 要发送的数据 len 要发送的数据长度
返回值	returnCode_e，具体含义参见 returnCode_e 枚举类型定义

3.4.4.9 cmiot_onenet_notify

原型	returnCode_e cmiot_onenet_notify(uint16_t funcId, uint16_t respId);
描述	向队列发送主动上报消息
参数	funcId 功能点 ID respId 响应 ID
返回值	returnCode_e，具体含义参见 returnCode_e 枚举类型定义

3.4.4.10 cmiot_onenet_notify_fromisr

原型	returnCode_e cmiot_onenet_notify_fromisr(uint16_t funcId, uint16_t respId);
描述	在中断服务程序中上报功能点
参数	funcId 功能点 ID respId 响应 ID
返回值	returnCode_e，具体含义参见 returnCode_e 枚举类型定义

3.4.4.11 cmiot_device_imei_read

原型	int cmiot_device_imei_read(uint8_t* devImei);
描述	读设备 imei
参数	devImei 保存设备 IMEI 的地址
返回值	0：成功 -1：失败

3.4.4.12 cmiot_device_sn_read

原型	int cmiot_device_sn_read(uint8_t* devSn);
描述	读设备 sn
参数	devSn 保存设备 sn 的地址
返回值	0：成功 -1：失败

3.4.4.13 cmiot_mod_imei_read

原型	int cmiot_mod_imei_read(uint8_t* modImei);
描述	读模组 imei
参数	modImei 保存模组 IMEI 的地址
返回值	0：成功 -1：失败

3.4.4.14 cmiot_mod_imsi_read

原型	int cmiot_mod_imsi_read(uint8_t* modImsi);
描述	读模组 imsi
参数	modImsi 保存模组 imsi 的地址
返回值	0：成功 -1：失败

3.4.4.15 cmiot_module_get_csq

原型	returnCode_e cmiot_module_get_csq(int *csq);
描述	获取信号质量
参数	csq 保存信号质量的地址
返回值	returnCode_e, 具体含义参见 returnCode_e 枚举类型定义

3.4.4.16 cmiot_module_get_menginfo

原型	returnCode_e cmiot_module_get_menginfo(menginfo_t * menginfo);
描述	获取网络信息
参数	menginfo 保存网络信息的地址

返回值	returnCode_e，具体含义参见 returnCode_e 枚举类型定义

3.4.4.17 cmiot_module_pdp_parameter_get

原型	returnCode_e cmiot_module_pdp_parameter_get(pdpPara_t *pdpParaGet);
描述	获取 APN
参数	pdpParaGet　保存 APN 的地址
返回值	returnCode_e，具体含义参见 returnCode_e 枚举类型定义

3.4.4.18 cmiot_module_reboot

原型	void cmiot_module_reboot(void);
描述	执行系统重启
参数	无
返回值	returnCode_e，具体含义参见 returnCode_e 枚举类型定义

3.4.5 生产接口

3.4.5.1 imei/sn 录入接口

设备首次上电时，要求录入 IMEI 号和 SN 号，使用串口工具按以下命令格式录入。

AT+SETDEVID={"imei_dev":<imei>,"sn_dev":<sn>}

参数说明：

<imei>:设备 imei 号，用双引号括起来的 15 位数字。

<sn>:设备 sn 号，用双引号括起来的 20 位数字或字母组合，区分大小写。

注意：录入 IMEI 号和 SN 号时，必须使用英文字符输入，命令区分大小写，且必须勾选发送新行。