MN316
OpenCPU CTWing 开发指导手册

NB-IoT 系列

版本：V1.0.0

日期：2021 年 2 月

中移物联网有限公司
iot.10086.cn

服务与支持

如果您有任何关于模组产品及产品手册的评论、疑问、想法，或者任何无法从本手册中找到答案的疑问，请通过以下方式联系我们。

中移物联网有限公司

OneMO 官网： onemo10086.com

邮箱： SmartModule@cmiot.chinamobile.com

客户服务热线： 400-110-0866

微信公众号： CMOneMO

文档声明

注意

本手册描述的产品及其附件特性和功能，取决于当地网络设计或网络性能，同时也取决于用户预先安装的各种软件。由于当地网络运营商、ISP，或当地网络设置等原因，可能也会造成本手册中描述的全部或部分产品及其附件特性和功能未包含在您的购买或使用范围之内。

责任限制

除非合同另有约定，中移物联网有限公司对本文档内容不做任何明示或暗示的声明或保证，并且不对特定目的适销性及适用性或者任何间接的、特殊的或连带的损失承担任何责任。

在适用法律允许的范围内，在任何情况下，中移物联网有限公司均不对用户因使用本手册内容和本手册中描述的产品而引起的任何特殊的、间接的、附带的或后果性的损坏、利润损失、数据丢失、声誉和预期的节省而负责。

因使用本手册中所述的产品而引起的中移物联网有限公司对用户的最大赔偿（除在涉及人身伤害的情况中根据适用法律规定的损害赔偿外），不应超过用户为购买此产品而支付的金额。

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。公司保留随时修改本手册中任何信息的权利，无需进行提前通知且不承担任何责任。

商标声明

中国移动 China Mobile 为中国移动注册商标。

本手册和本手册描述的产品中出现的其他商标、产品名称、服务名称和公司名称，均为其各自所有者的财产。

进出口法规

出口、转口或进口本手册中描述的产品（包括但不限于产品软件和技术数据），用户应遵守相关进出口法律和法规。

隐私保护

关于我们如何保护用户的个人信息等隐私情况，请查看相关隐私政策。

操作系统更新声明

操作系统仅支持官方升级；如用户自己刷非官方系统，导致安全风险和损失由用户负责。

固件包完整性风险声明

固件仅支持官方升级；如用户自己刷非官方固件，导致安全风险和损失由用户负责。

版权所有©中移物联网有限公司。保留一切权利。

关于文档

修订记录

版本	发布日期	作者	描述
V1.0.0	2021/2/24	张雄威	初版

目录

1 功能介绍

CTWing 业务涉及到的 API 接口主要包括：

接口	描述
int cm_ctw_create (char* ip, int port);	仅支持 IP 模式，使用时请传入 CTWing 平台 LwM2M 协议服务器 IP。
int cm_ctw_delete (void);	执行后将会强制注销设备，并删除 ip、port 等配置。
int cm_ctw_cfg (const cm_ctw_cfg_t* cfg);	配置重传及过滤
int cm_ctw_open (int lifetime, cm_ctw_cb_t* cb);	开始注册
int cm_ctw_close (void);	注销设备
int cm_ctw_update (void);	更新注册
int cm_ctw_send (const char* data, int len);	发送数据
int cm_ctw_send_ex (const char* data, int len, cm_ctw_coap_ack_type_e con, cm_ctw_rai_type_e rai, unsigned char seq);	自定义发送数据
int cm_ctw_fota_query_state (cm_ctw_fota_state_t* state);	获取 FOTA 升级状态
void cm_ctw_fota_get_version (char version [16]) __attribute__((weak));	FOTA 版本号获取，此接口由用户实现。

1.1 创建及注册

模组必须在注册至 CTWing 平台后，才能与平台进行数据收发业务。平台端的设备创建过程请参考《中国电信物联网开放平台文档》，本文档仅介绍模组侧的执行过程。另外，模组仅支持“明文”数据加密格式，“认证方式”为“IMEI 认证”，“Endpoint 格式”为“imei”。

1.1.1 创建设备

创建设备
函数原型
int cm_ctw_create(char* ip, int port);
备注
仅支持服务器 ip 配置
示例

```
char* server_ip = "221.229.214.202";
int server_port = 5683;

/*配置服务器信息*/
int ret = cm_ctw_create(server_ip, server_port);          //配置 CTWing 服务器

if(ret != 0)
{
    cm_printf("ctw create err,%d\n", ret);
    goto ctw_exit;
}
```

1.1.2 注册至平台

调用接口注册至平台后，结果将以异步形式告知用户，用户须在回调函数中处理对应的响应结果。

注册至平台
函数原型
int cm_ctw_open(int lifetime, cm_ctw_cb_t* cb);
备注
lifetime 为设备保活时间，cb 为注册的回调函数集合，注册成功后将会触发 evt_cb 回调函数： evt_cb(1)，注册成功； evt_cb(7)，订阅成功。
示例

```
ctw_cb.send_cb = test_ctw_send_cb;
ctw_cb.recv_cb = test_ctw_recv_cb;
ctw_cb.evt_cb = test_ctw_event_cb;
ctw_cb.fota_cb = test_ctw_fota_cb;
ctw_cb.seq_cb = test_ctw_seq_cb;                    //回调函数设置

int wait_time = 0;                                  //注册至服务器
ret = cm_ctw_open(86400, &ctw_cb);

if(ret != 0)
{
    cm_printf("ctw open err,%d\n", ret);
    goto ctw_exit;
}
```

1.2 发送数据

在设备成功注册至平台后，用户可调用发送接口发送数据至平台。

1.2.1 标准数据发送

标准数据发送
函数原型
int cm_ctw_send(const char* data, int len);
参数描述
<data>
待发送数据
<len>
数据长度
备注
发送 CON 数据，待接收到服务器 ACK 后立即释放 RRC； 执行后触发 open 时传入的 send_cb 回调函数，接收到 ACK 后触发 evt_cb 回调函数： send_cb(mID)，表示该数据消息 ID 为 mID； evt_cb(9，mID)，消息 ID 为 mID 的数据发送成功； evt_cb(10，mID)，消息 ID 为 mID 的数据发送失败。
示例

```
char test_data[] = "foo foo foo foo";
int ret = cm_ctw_send(test_data, strlen(test_data));

if(ret != 0)
{
    cm_printf("ctw send err,%d\n", ret);
}
```

1.2.2 自定义数据发送

自定义数据发送	
函数原型	
int cm_ctw_send_ex(const char* data, int len, cm_ctw_coap_ack_type_e con, cm_ctw_rai_type_e rai, unsigned char seq);	
参数描述	
<con>	
0	NON
1	CON
<rai>	
0	与基站协商释放
1	发送数据后立即释放
2	接收到下行包后立即释放
<seq>	
空口回传序列号，有效范围 1-255，传入 0 表示不需要携带该标识。	
备注	
发送成功/失败，均会触发 open 时传入的 send_cb 回调函数执行； 若携带空口回传标识，则将会触发 open 时传入的 seq_cb 回调函数执行。 seq_cb(seq, 0)，数据发送成功； seq_cb(seq, -1)，数据发送失败。	
示例	

```
char test_data[] = "bar bar bar bar";
int ret = cm_ctw_send_ex(test_data, strlen(test_data),
CM_CTW_COAP_TYPE_NON, CM_CTW_RAI_1, 123);          //NON 模式，发送后立即释放 RRC，空口标识 123。

if(ret != 0)
{
    cm_printf("ctw send err,%d\n", ret);
}
```

1.3 接收数据

UE 接收数据使用异步方式通知用户，当接收到 CTWing 服务器下发的数据时，将会触发用户在 open 时传入的 recv_cb 回调函数。

1.3.1 接收回调

接收回调
函数原型
typedef void (* cm_ctw_recv_cb)(const char* data, int len);
备注
请避免在回调函数中处理耗时的操作，建议仅做数据拷贝。
示例

```
void test_ctw_recv_cb(const char* data, int len)          //接收数据回调函数。
{
    if(data == NULL || len <= 0)
    {
        return;
    }

    char* recv_data = (char*)cm_malloc(len + 1);

    memcpy(recv_data, data, len);
    recv_data[len] = '\0';
    cm_printf("+CTWRECV:%d,%s\n", len,
recv_data);
                                                          //打印接收到的数据，示例仅供参考。
    cm_free(recv_data);
}
```

1.4 更新注册

当设备成功注册至平台后，用户需要周期性的更新注册，保证设备一直处于保活期，正常与 CTWing 平台进行数据收发业务；否则，当 lifetime 过期后，数据将发送失败，平台数据也无法下达。

1.4.1 更新注册

更新注册
函数原型
int cm_ctw_update(void);
备注
执行后，设备将以 open 时传入的 lifetime 重置在平台的保活时间； 当 liftetime 未过期，执行 update 后将会更新注册；当 lifetime 已过期，执行 update 后将会重新注册。 更新成功后，将触发 open 时传入的 evt_cb 回调函数，详情请参考源码头文件描述。 evt_cb(3)，更新成功。

1.5 注销及删除设备

用户在 CTWing 业务完成后，可调用接口注销或删除设备。如果需要频繁的进行创建和删除操作，请在删除完成后（触发注销成功回调函数）再进行下一次创建。

1.5.1 注销

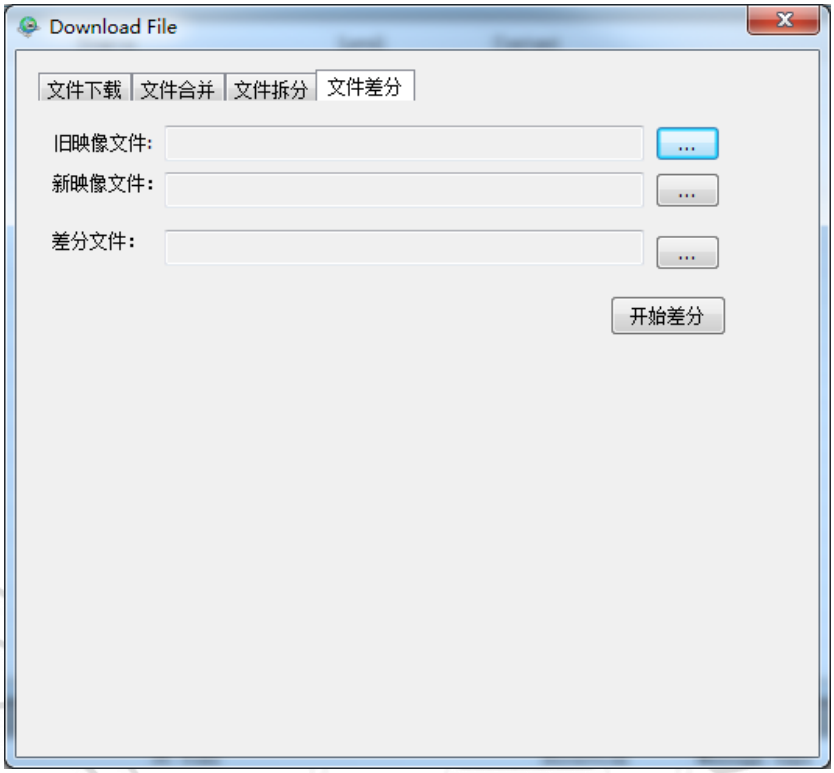
注销
函数原型
int cm_ctw_close(void);
备注
注销成功会触发 open 时传入的 evt_cb 回调函数。 evt_cb(5)，注销成功。

1.5.2 删除设备

删除设备
函数原型
int cm_ctw_delete(void);
备注
执行后将会强制注销设备，触发 open 时传入的 evt_cb 回调函数；并且删除 create 时配置的服务器等信息。 evt_cb(5)，注销成功。

1.6 FOTA 升级

FOTA 升级由 CTWing 平台端触发，差分升级包使用 LogView 工具生成。旧映像文件选择当前的 image 固件，新映像文件选择要升级的 image 固件，选择差分文件存放位置，点击“开始差分”即可。



平台端开始升级流程后，模组侧的 fota_cb 回调函数将会触发执行，用户可在回调函数中获取实时的升级状态，详细的回调函数状态说明请参考源码中的头文件。

使用此功能时请务必保证差分升级包的正确性，升级过程中请确保模组不会断电，否则将会产生无法预知的后果！

1.6.1 版本号获取

版本号获取
函数原型
void cm_ctw_fota_get_version(char version[16]) __attribute__((weak));
备注
此接口定义为弱引用类型，当用户有 FOTA 升级业务时，必须自行实现此接口； 版本号长度不大于 16 字节，仅支持数字、字母、‘_’、‘.’。
示例

```
void cm_ctw_fota_get_version(char version[16])                    //获取版本号
{
    if(version == NULL)
    {
        return;
    }

    int len = strlen(g_ctw_fota_version);
    len = (len > 16) ? 16 : len;

    memcpy(version, g_ctw_fota_version, len);
}
```

1.6.2 查询 FOTA 升级状态

用户注册 fota_cb 回调函数后，FOTA 升级开始将会触发回调函数执行，用户可在回调函数中获取升级状态，也可以调用 cm_ctw_fota_query_state 手动查询。

查询 FOTA 升级状态

函数原型

int cm_ctw_fota_query_state(cm_ctw_fota_state_t* state);

备注

关于 state 结构的描述及升级状态说明请参考源码头文件说明，此处不再赘述。

示例

```
void test_ctw_fota_query(void)                                    //升级状态查询
{
    cm_ctw_fota_state_t state = {0};

    int ret = cm_ctw_fota_query_state(&state);

    if(ret != 0)
    {
        cm_printf("ctw fota state err,%d\n", ret);
        return;
    }

    switch(state.state)
    {
        case CM_CTW_FOTA_IDLE://空闲态
        {
            cm_printf("+FOTA:%d,%d\n", state.state, state.n1);
        }
        break;
        case CM_CTW_FOTA_DOWNLOADING://下载中
        {
            cm_printf("+FOTA:%d,%d,%d\n", state.state, state.n1,
state.n2);
        }
        break;
        case CM_CTW_FOTA_DOWNLOADED://下载完成
        {
            cm_printf("+FOTA:%d\n", state.state);
        }
        break;
        case CM_CTW_FOTA_UPDATING://升级中
        {
            cm_printf("+FOTA:%d\n", state.state);
        }
        break;
        case CM_CTW_FOTA_UPDATED://升级完成
        {
            cm_printf("+FOTA:%d,%d\n", state.state, state.n1);
        }
        break;
        default:
        break;
    }
}
```

1.7 深睡眠及唤醒

如果模组在进入深睡眠之前已成功注册至 CTWing 平台，则模组进入深睡眠后将会备份 CTWing 的接入状态，并将 open 时传入的回调函数一并备份；当模组被唤醒后，将会恢复 CTWing 的状态及回调函数，用户可直接进行数据收发业务，无需再重新注册。成功恢复后，将会触发回调函数 evt_cb(13)，用户可在回调函数中获取恢复状态并执行相应的初始化操作。

2 附录

2.1 参考文档

序号	文档名称	备注
[1]	MN316_OpenCPU MANUAL	API 接口手册