

## OneNET 平台 NB-IOT 接入开发文档

## 版本更新信息

欢迎访问 OneNET 官方网站注册用户，获取最新文档

版本号	修订日期	修订内容	作者	说明
V1.0	2018.5		肖勇、张小波、 卓定飞、敬威	初版
V2.0	2018.6.14	1. 优化了南北向开发流程图； 2. 对 API 进行分类； 3. 新增了第八章“常见问题”的内容	肖勇、张小波、 卓定飞、敬威	
V3.0	2018.6.20	1. 新增第三方应用 URL&Token 验证过程； 2. 新增 API 接口的表序；	肖勇、张小波、 卓定飞、敬威	
V3.1	2018.8.2	1. 新增 imei 查询设备信息； 2. 新增 FOTA 说明部分；	肖勇、陈小波、 张伟	

## 目录

第一章 文档说明.....	5
第二章 OneNET 平台 NB 设备开发流程总述.....	6
2.1 NB 设备接入 OneNET 平台流程图.....	6
2.2 应用开发流程图.....	6
第三章 NB 设备接入 OneNET 平台.....	8
3.1 终端设备接入 OneNET 平台前的准备工作.....	8
3.1.1 SDK 移植到 MCU.....	9
3.1.2 SDK 移植到 NB 通信模组.....	9
3.1.3 SDK 移植到 NB 芯片.....	10
3.2 终端设备接入 OneNET 平台步骤.....	11
3.2.1 OneNET 平台创建产品及设备.....	12
3.2.2 终端设备软硬件初始化.....	15
3.2.3 终端创建设备及资源.....	15
3.2.4 登录 OneNET 平台.....	16
3.2.5 平台订阅&发现设备资源.....	17
第四章 第三方应用开发.....	18
4.1 第三方应用接入 OneNET 平台.....	18
4.1.1 第三方应用平台接入验证程序.....	19
4.1.2 OneNET 平台配置第三方应用平台.....	19
4.2 OneNET 平台数据推送.....	21
4.2.1 明文消息.....	21
4.2.2 密文消息.....	22
4.2.3 消息相关字段说明.....	23
4.2.4 加密算法详述.....	23
4.3 API 接口.....	24
4.3.1 创建设备.....	25
4.3.2 查看单个设备信息.....	26
4.3.3 删除设备.....	27
4.3.4 根据 IMEI 查询设备信息.....	27
4.3.5 读设备资源.....	28

---

4.3.6 写设备资源.....	29
4.3.7 下发命令.....	30
4.3.8 获取资源列表.....	30
4.3.9 订阅.....	31
4.3.10 离线命令.....	32
4.3.11 触发器.....	36
4.3.12 批量查询设备状态.....	41
4.3.13 批量查询设备最新数据.....	42
4.3.14 查看数据点.....	43
第五章 接入实例.....	47
5.1 MCU 侧工作流程说明.....	48
5.1.1 创建设备（dev）.....	48
5.1.2 向设备添加资源.....	49
5.1.3 登录请求.....	50
5.2 OneNET 平台侧数据收发流程说明.....	51
5.2.1 数据接收.....	51
5.2.2 指令下发.....	52
第六章 资源下载.....	58
6.1 数据推送 SDK 下载.....	58
6.2 NB-IoT API 下载.....	58
6.3 NB-IoT 开发板资料下载.....	58
第七章 NB-IoT 接入协议说明.....	59
7.1 基于 NB-IoT 的 LWM2M 协议.....	59
7.1.1 LWM2M 协议逻辑实体与逻辑接口.....	60
7.1.2 LWM2M 协议栈.....	61
7.2 基于 NB-IoT 的 CoAP 协议.....	62
7.2.1 CoAP 协议栈.....	62
7.2.2 块传输.....	64
7.2.3 安全传输.....	66
第八章 常见问题 FAQ.....	67
8.1 终端设备接入 OneNET 平台的常见问题.....	67
8.2 API 调用的常见问题.....	68
8.3 使用 OneNET 平台遇到的常见问题.....	69
8.4 OneNET 平台 FOTA 功能.....	71

## 第一章 文档说明

编写此开发文档是为了让开发人员采用 NB 模组的设备快速接入 OneNET 平台，形成基于 NB 设备的物联网开发应用。

通过阅读此文档，开发人员可以了解 OneNET 平台接入的总体流程，包括 NB 设备接入 OneNET 平台步骤，基于 OneNET 平台开发上层应用。

文档适用人员：熟悉 NB 协议、设备接入侧需要具备一定的嵌入式开发人员，北向 API 调用的开发人员，需要具备一定的应用软件开发能力。

注意：建议以公司名义先注册 OneNET 平台账号

## 第二章 OneNET 平台 NB 设备开发流程总述

基于 OneNET 平台的 NB 设备接入及应用开发流程分两个阶段进行，第一阶段为 NB 设备接入 OneNET 平台，第二阶段为基于设备上传数据流的应用开发。下面将用流程图的形式分别对设备接入和应用开发进行综述。

注意：流程图中的数字标号从小到大表示流程的步骤。

### 2.1 NB 设备接入 OneNET 平台流程图

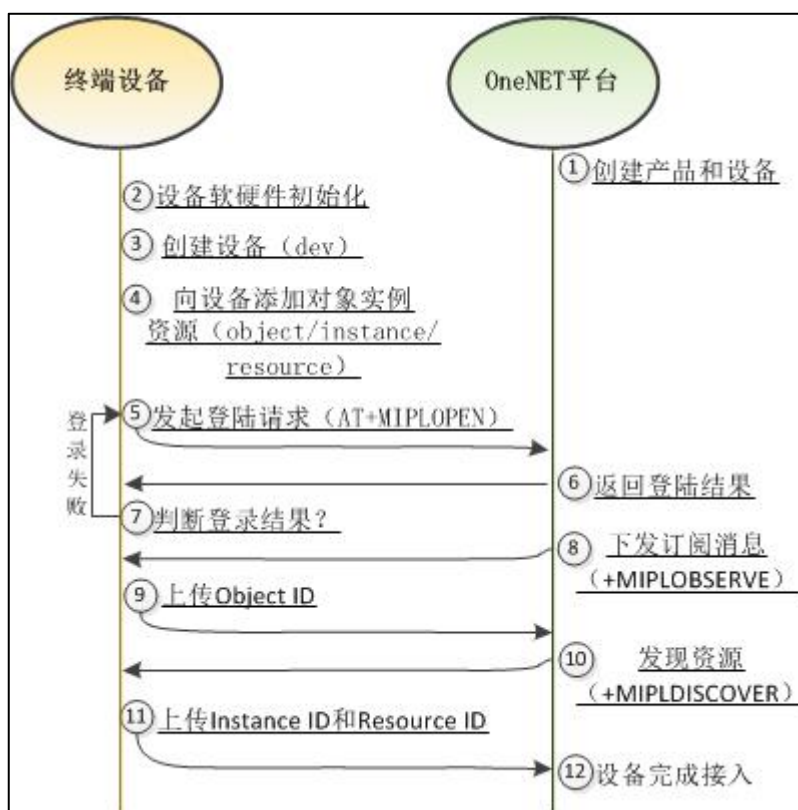


图2-1 NB设备接入OneNET平台流程图

### 2.2 应用开发流程图

OneNET 平台提供的应用开发 API，分为需要终端响应和只需 OneNET 平台响应两类，关于 API 的所属类别及名称请参考第四章表 4-3。下面分别用两个流程图描述这两类 API 的开发流程。图 2-2 为需要终端响应的 API 调用流程，其中虚线部分只有离线命令的读/写/下发命令才有该流程。图 2-3 为只需平台响应的 API 调用流程。

在应用开发平台调用 API 之前，**必须**与 OneNET 平台建立连接，建立连接的过程和 API 调用流程一并在图中进行了标注。

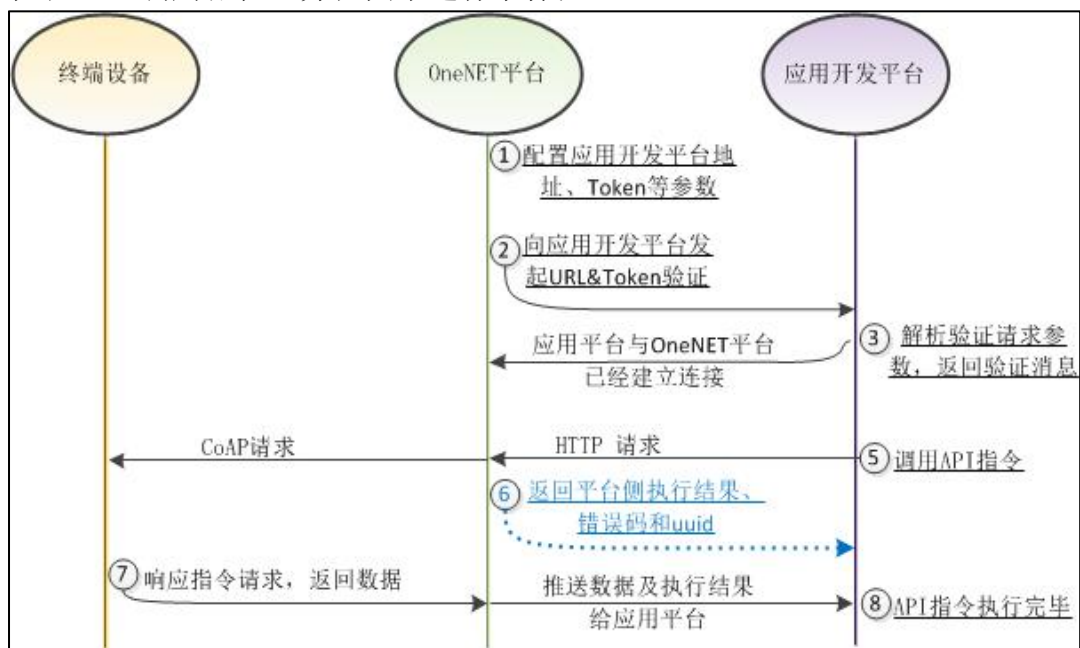


图2-2 需要终端响应的API调用流程



图2-3 只需平台响应的API调用流程

### 第三章 NB 设备接入 OneNET 平台

终端设备与 OneNET 平台进行数据交互需经过网络运营商提供的核心网，根据 OneNET 平台与网络运营商的关系，终端设备、核心网络、OneNET 平台以及第三方应用四者的网络关系可归纳如图 3-1 所示。

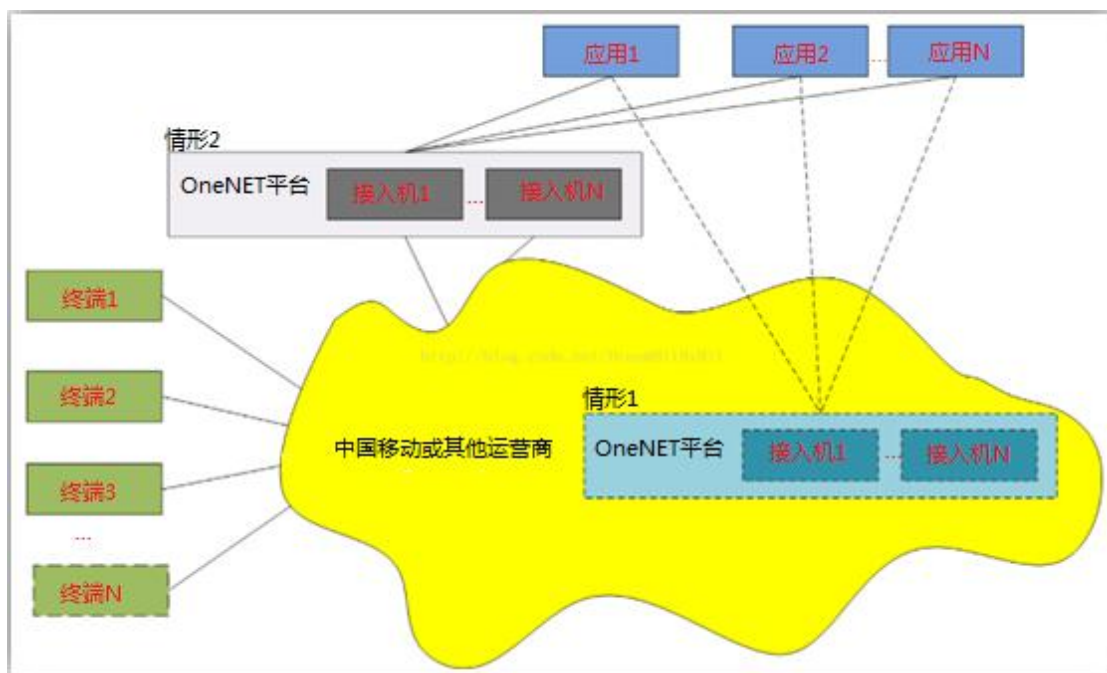


图3-1 终端设备与服务器网络关系

图 3-1 中终端设备用户如果选择中国移动网络运营商，可以与 OneNET 平台实现无缝对接。图 3-1 中如果终端设备用户选择电信或者联通运营商，则需要电信和联通运营商更改对 OneNET 平台接入限制。电信目前是要求接入 OC 平台，联通 NB 暂时未规划。

#### 3.1 终端设备接入 OneNET 平台前的准备工作

在设备接入 OneNET 平台之前，设备侧需完成由 OneNET 平台提供的基础通信套件 SDK 的移植工作。目前，按 SDK 集成在终端的方式可将终端分为如下三类：

- (1) “MCU+NB 通信模组”架构中，SDK 移植到 MCU 中；
- (2) “MCU+NB 通信模组”架构中，SDK 移植到 NB 通信模组中；
- (3) “NB 芯片”架构中，SDK 移植到 NB 芯片中。



### 3.1.1 SDK 移植到 MCU

此终端类型，除了在 MCU 中移植基础通信套件 SDK 外，还需在 MCU 中完成设备应用程序开发、抽象接口实现和底层操作系统及驱动实现。用户需要调用基础通信套件封装的接口完成对资源的操作。如图 3-2 所示为 MCU+NB 通信模组架构及软件开发层次分布。

目前，用户如果使用这种软硬件架构的终端设备接入 OneNET 平台，需要自行将 SDK 移植到 MCU，NB 通信模组只是作为透传的传输通道。

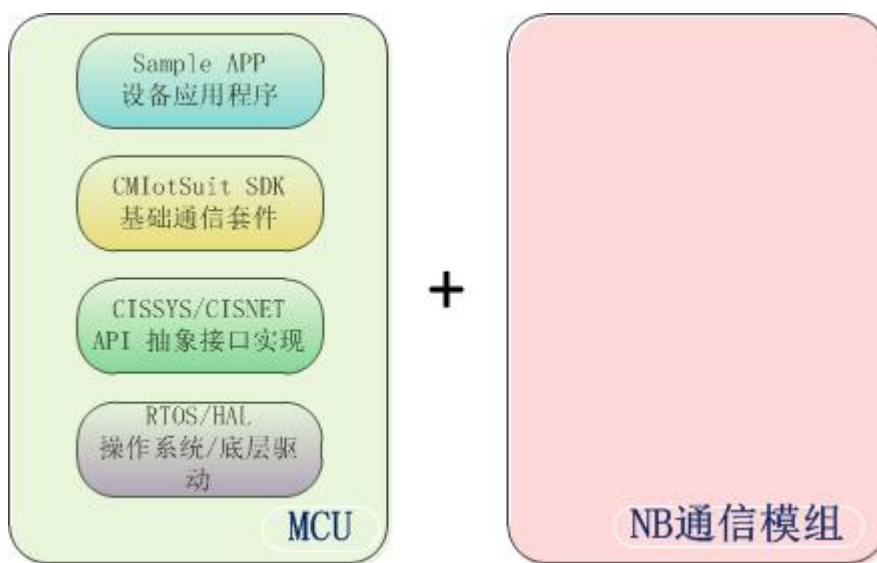


图3-2 SDK移植到MCU中的软硬件架构图

### 3.1.2 SDK 移植到 NB 通信模组

此终端类型，除了在 NB 通信模组中移植基础通信套件 SDK，还需在 NB 通信模组中完成基础通信套件 AT 指令封装、抽象接口实现和底层操作系统及驱动实现，MCU 中只需要完成设备应用程序的开发。用户需要调用模组提供的 AT 指令完成对资源的操作。如图 3-3 所示为 MCU+NB 通信模组架构及软件开发层次分布。

目前，支持接入 OneNET 平台的此种软硬件架构的终端设备中，已完成移植接入 OneNET 平台的 SDK 模组如表 3-1 所示。另外，如有需要模组厂家需要移植接入 OneNET 平台的 SDK，请与 OneNET 平台的商务经理联系。

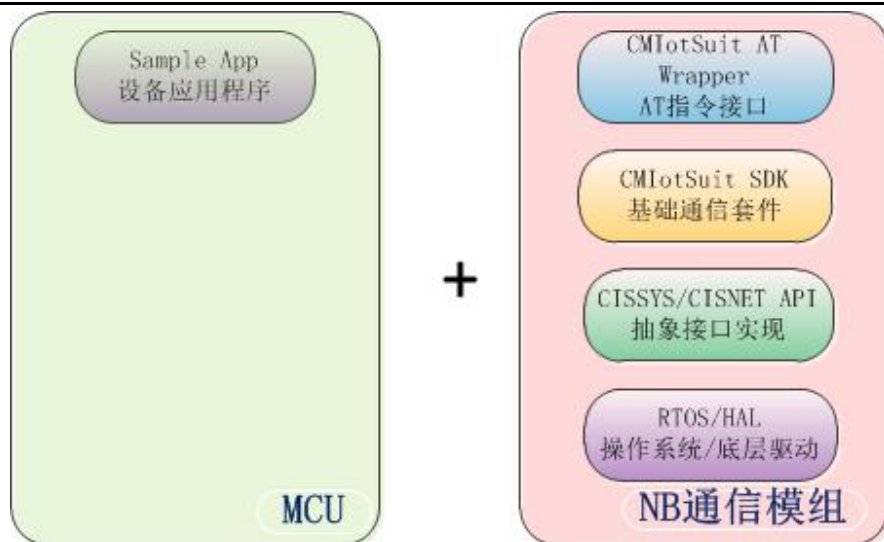


图3-3 SDK移植到NB通信模组中的软硬件架构图

表 3-1 已完成移植接入 OneNET 平台的 SDK 模组

厂商名称	产品类型	产品型号
中移物联网	模组	M5310
广和通	模组	N700-CN
移远	模组	BC95
骐俊物联	模组	ML5330
SIMCom	模组	SIM7000C-N
新华三	模组	IM2209
备注	中国移动集团终端公司会持续更新认证的模组	

### 3.1.3 SDK 移植到 NB 芯片

此终端类型，除了在 NB 芯片中移植基础通信套件 SDK，还需在 NB 芯片中完成设备应用程序开发、基础通信套件 AT 指令封装、抽象接口实现和底层操作系统及驱动实现。如图 3-4 所示为 NB 芯片架构及软件开发层次分布。

目前，支持接入 OneNET 平台的此种软硬件架构的终端设备中，已完成移植接入 OneNET 平台的 SDK NB 芯片如表 3-1 所示。另外，如有需要移植接入 OneNET 平台的 SDK，请与 OneNET 平台的商务经理联系。



图3-4 SDK移植到NB芯片中的软硬件架构图

表 3-2 已完成移植接入 OneNET 平台的 SDK NB 芯片

厂商名称	产品类型	产品型号
中兴微电子	芯片	RoseFinch7100
高通	芯片	MDM9206
MTK	芯片	MT2625
华为海思	芯片	Boudica120
	芯片	Boudica150

### 3.2 终端设备接入 OneNET 平台步骤

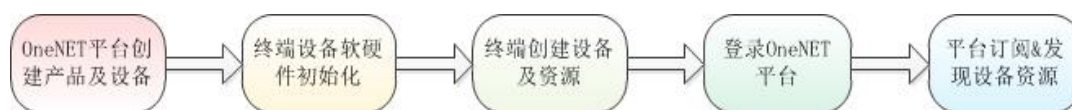


图3-5 终端设备接入OneNET平台的五个步骤

终端设备完成基础通信套件移植以及其他软件功能实现后，可按照如图 3-5 所示步骤实现终端设备接入 OneNET 平台。待终端设备收到平台下发的订阅/发现命令后，说明终端设备已经成功接入到 OneNET 平台。

### 3.2.1 OneNET 平台创建产品及设备

平台端创建的设备（或称“虚拟设备”）是真实设备在平台端的映射，用户可以通过平台端的“虚拟设备”对真实设备进行信息查询、命令下发、数据流管理、添加触发器等操作。

为了使用上述 OneNET 平台提供的设备云的强大功能，你可以参考以下步骤完成平台端“虚拟设备”的创建。

#### 3.2.1.1 注册用户账号

进入 OneNET 官网（<https://open.iot.10086.cn/>），点击首页右上角的“注册”按钮，注册用户账号。

OneNET 支持“**个人用户**”和“**企业用户**”两种注册方式，你可以根据您的实际情况选择注册方式。

以个人用户注册为例，用户需在对应选项依次填写用户名、用户密码、所在地、手机号码、图片验证码以及手机验证码，并完成注册。注册页面如图 3-6 所示。注册完成后，回到主页点击“登录”，即可进入 OneNET 的官方主页，并由此进入到你的“**开发者中心**”。

如果客户产品后续需要进行规模化量产，建议以企业的名义进行注册，注册时注意注册信息的准确填写，另外，个人版和企业版在功能上无差异。



The screenshot shows the OneNET user registration interface. At the top, there is a navigation bar with the OneNET logo and links for '首页' (Home), '案例与伙伴' (Cases & Partners), '开发文档' (Development Docs), '发现' (Discover), '社区' (Community), and '动态' (Dynamics). On the right side of the navigation bar are buttons for '登录' (Login) and '注册' (Register). Below the navigation bar, there are two tabs: '个人用户' (Personal User) and '企业用户' (Enterprise User). The '个人用户' tab is selected and highlighted with a red circle. The registration form includes the following fields: '用户名' (Username) with a yellow input field, '设置密码' (Set Password) with a yellow input field and a lock icon, '确认密码' (Confirm Password) with a white input field and a lock icon, and '手机 (中国+86)' (Mobile Phone (China+86)) with a white input field. There is also a '邮箱注册' (Email Registration) link at the bottom right. A blue button with a checkmark icon is located at the bottom right of the form.

图3-6 用户账号注册页面

### 3.2.1.2 创建产品和设备

在进入到你的“**开发者中心**”后，点击右上角的“**创建产品**”，在弹出页面中按照提示填写产品的基本信息，进行产品创建。在创建过程中，请您按照提示尽可能完整、全面地填写相应内容，这样更方便您后期对产品进行管理。

创建产品页面如图 3-7 和图 3-8，其中设备接入方式选择“**公开协议**”，联网方式选择“**NB-IoT**”，设备接入协议选择“**LWM2M**”。

产品创建完成后，页面弹出“**创建产品成功**”，如图 3-9 所示。点击“**立即添加设备**”，显示如图 3-10 所示界面。依次填写“**设备名称**”、**IMEI** 和 **IMSI**，并根据需要在“**是否开启自动订阅**”选项选择“**是**”或者“**否**”。如果用户开启自动订阅功能，终端设备创建的资源信息会自动更新到平台上对应“**虚拟设备**”的资源列表下。用户需要将“**虚拟设备**”里面的 IMEI 和 IMSI 更新到终端设备里面，用于平台对终端设备鉴权。

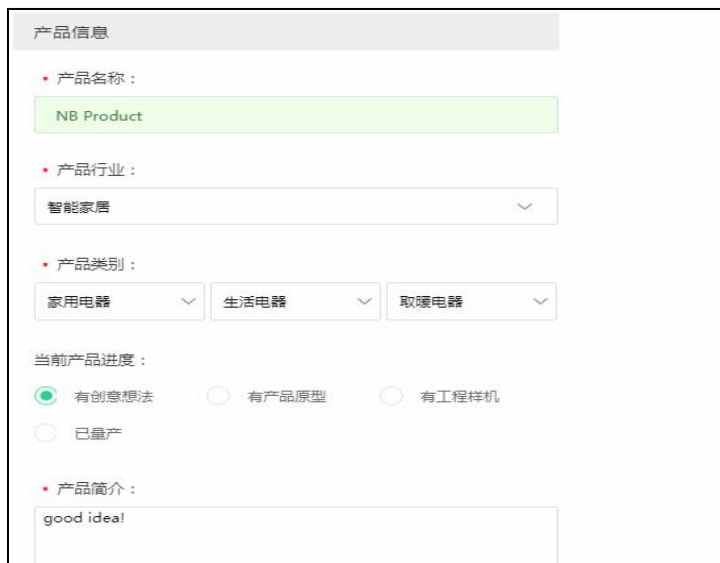


图3-7 创建产品页面

网络运营商：

☒ 移动    ☐ 电信    ☐ 联通

☐ 其他

设备接入方式：

☒ 公开协议    ☐ 私有协议(RGMP)

联网方式：

☐ wifi    ☐ 移动蜂窝网络    ☒ NB-IoT

设备接入协议：

LWM2M

LWM2M基本功能介绍：

1. 低功耗；
2. 广覆盖；
3. 低成本；
4. 强连接；

确定    取消

图3-8 创建产品页面

 创建产品成功！

创建产品成功了，可以添加设备了哦！

立即添加设备    暂不添加

图3-9 创建产品成功页面



图3-10 添加设备页面

### 3.2.2 终端设备软硬件初始化

终端设备在上电以后，需要完成设备的软硬件初始化。硬件初始化包括各个硬件模块的上电、处理器的寄存器初始化、工作模式初始化等等。软件初始化包括基础通信套件初始化、应用接口初始化、应用程序初始化等等。

### 3.2.3 终端创建设备及资源

终端设备在应用程序中创建设备（dev），在设备中配置好接入机地址、endpoint name（也即鉴权信息 IMEI、IMSI 等信息）、lifetime 以及回调函数（读、写和执行函数）后，应用程序中创建完成的设备会在基础通信套件中创建同样的设备，在设备登录 OneNET 平台成功后，基础通信套件中的设备会上传到平台。

目前 OneNET 平台支持 LWM2M 和 IPSO 定义的资源模型，用户需根据传感器类型从这两种标准中选择适合的资源模型。资源模型为 Object/Instance/Resource 三层结构。

- Object（对象）：表示某类传感器类型。
- Instance（实例）：同一类传感器的数量。
- Resource（属性）：传感器某些特性描述。

终端设备在应用程序中创建完设备（dev）后，还需要创建 object 以及对应的 instance 和 resource。同样的，应用程序中创建的设备资源会在基础通信套件中创建同样的设备资源。

下面以Object ID=3311（IPSO Light Control），Instance=0/1/2（三个实例），Resource ID=5851/5850（Dimmer/Switch，详见表 3-3 描述）为例描述 Object/Instance/Resource三者的关系,如图3-11所示。

表 3-3 Resource ID=5851/5850 描述

Resource ID	Description	Type	Access
5851	Dimmer	Integer 0-100	R,W
5850	On/Off	Boolean	R,W

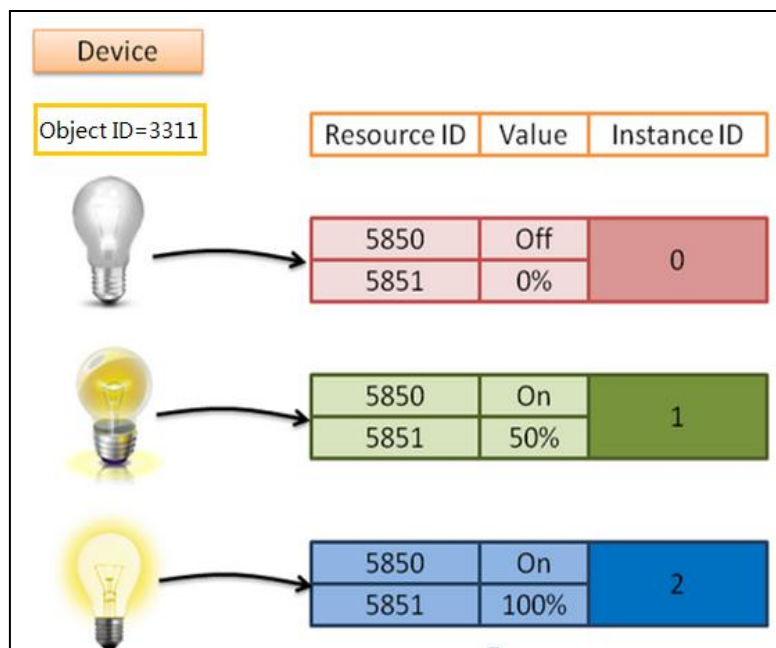


图3-11 Object/Instance/Resource三者的关系

### 3.2.4 登录 OneNET 平台

终端设备的基础通信套件完成初始化以及设备资源创建完成后，可向 OneNET 平台上报登录请求的注册码，服务器在收到登录请求的注册码后，会验证注册码中的参数，返回登录结果。如果参数错误或者登录超时，平台会返回登录失败。如果验证通过，平台会返回登录成功。

若终端设备登录平台成功，设备名称左边的圆形图标会由灰色（离线状态）



变成绿色（在线状态），如图 3-12 所示。

<p><span style="color: green;">●</span> NB-test</p> <p>设备ID:2447977</p> <p>创建时间:2017-09-12 10:24:56</p>	公开	   
---	----	---

图3-12 平台侧设备状态显示

### 3.2.5 平台订阅&发现设备资源

终端设备登录平台成功后，OneNET 平台会向设备下发 Observer 消息和 Discover 消息。终端设备收到这两条消息后，基础通信套件会自动处理，无需用户另行处理。

例如，用户创建了Object ID=3200（IPSO Digital Input）的对象，在设备成功登录到平台后，平台会向该对象下发Observer消息和Discover消息。终端收到的消息如图3-13所示。

```
+MIPLOBSERVE:0,55572,1,3200,0,-1
+MIPLDISCOVER:0,55573,3200
```

图3-13 平台下发的Observer消息和Discover消息格式

其中，Observer 消息是平台传递的观测请求消息，基础通信套件收到该消息后会自动维护相关观测记录。Discover 消息是通知基础通信套件需要获取指定 Object 的属性，基础通信套件自动反馈相关属性。初次上传到平台的 Object 及其属性如图 3-14 所示。

序号	对象名称	实例个数	属性个数
1	Digital Input	1	1

序号	实例名称	属性名	属性类型	属性值	时间	操作
1	Digital Input_0	Digital Input State	boolean	null	null	<a href="#">读</a> <a href="#">写</a> <a href="#">执行</a> <a href="#">详情</a>
		Digital Input Counter	integer	null	null	<a href="#">读</a> <a href="#">写</a> <a href="#">执行</a> <a href="#">详情</a>
		Digital Input Counter Reset	opaque	null	null	<a href="#">读</a> <a href="#">写</a> <a href="#">执行</a> <a href="#">详情</a>
		Application Type	string	null	null	<a href="#">读</a> <a href="#">写</a> <a href="#">执行</a> <a href="#">详情</a>

图3-14 平台侧显示的Object及其属性

## 第四章 第三方应用开发

第三方应用与 OneNET 平台的业务逻辑关系如图 4-1 所示，第三方应用与 OneNET 平台之间通过 HTTPS 请求/应答的方式实现数据交互。OneNET 平台为第三方应用提供封装好的 API 接口，应用平台通过调用这些 API 接口完成对 OneNET 平台的读写执行以及设备管理请求，然后 OneNET 平台会将相应的指令请求发送到终端设备。OneNET 平台收到终端设备响应的数据及设备信息后，会将数据及设备信息推送到应用平台，完成应答。

下面将分别介绍第三方应用接入 OneNET 平台的步骤、OneNET 平台提供的的数据推送功能以及 OneNET 平台为第三方应用封装好的 API 接口。



图4-1 第三方应用与OneNET平台的逻辑关系

### 4.1 第三方应用接入 OneNET 平台

第三方应用与 OneNET 平台实现数据交互的前提是已经建立连接，第三方应用接入 OneNET 平台的开发流程如图 4-2 所示，包括三个步骤。

- (1) 在 OneNET 平台创建了 NB-IoT 设备，该设备有对应的真实设备且能正常上报数据。
- (2) 用户已经开发并部署了 OneNET 平台验证接入的程序。
- (3) 在 OneNET 平台正确配置了第三方应用数据接收的地址（URL）、Token 以及其他参数。

在确保上述第（1）步已经实现的基础上，下面将分别讲述第（2）步和第（3）步的实现过程。

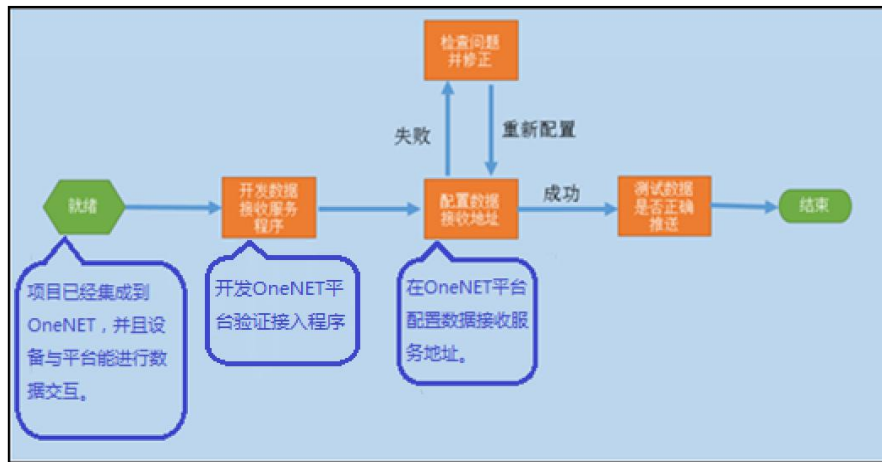


图4-2 数据推送的开发流程

#### 4.1.1 第三方应用平台接入验证程序

第三方应用平台接入验证程序工作流程如下所述：当 OneNET 平台完成对第三方开发平台的配置后，会向第三方开发平台发起 URL&Token 验证。接入验证程序对平台发送的验证参数进行验证，验证通过后，第三方开发平台会向 OneNET 平台返回验证消息，当 OneNET 平台收到验证消息后说明第三方开发平台与 OneNET 平台已建立连接。

目前 OneNET 平台为第三方应用平台提供接入验证程序的 SDK，有 [Java](#)，[php](#)，[Nodejs](#) 和 [C#](#)四个版本。

SDK 下载地址：<https://open.iot.10086.cn/doc/art432.html#118> ->“数据推送”

#### 4.1.2 OneNET 平台配置第三方应用平台

在 OneNET 平台对第三方应用平台的配置界面如图 4-3 所示。

- **URL**：第三方开发平台的地址；
- **Token**：可以自定义，用于第三方开发平台接入平台握手验证；

**消息加解密方式**：有明文模式和安全模式。当启用安全模式后，平台端需要用户自定义 AESKey 或者由平台随机生成，在应用程序中使用该 AESKey 解密消息。

**推送时间限制(秒)和推送数据限制(个)**：设置“推送时间限制”、“推送数据限

制”可以实现批量推送数据，减少服务器压力。当满足“推送时间限制”和“推送数据限制”要求后，OneNET 将给第三方推送数据。如果没有设置“推送时间限制”和“推送数据限制”，将默认数据实时推送。



URL : 必须以http://开头

[验证说明](#)

Token : 必须以英文或数字，长度为3-32字符

Token将用于接入验证，可自行定义。

消息加解密方式：请根据业务需求，选择加解密类型，启用后将立即生效。

☒ 明文模式  
明文模式下，不适用消息体加解密功能，安全系数较低

☐ 安全模式（推荐）  
安全模式下消息包为纯密文，需要开发者加密和解密，安全系数高

[更多高级设置](#)

推送时间限制(秒)：请输入小于60的整数

推送数据限制(个)：请输入小于1000的整数

设置【推送时间限制】、【推送数据限制】可以实现批量推送数据，减少服务器压力，当满足【推送时间限制】或者【推送数据限制】要求后，OneNET将给第三方推送数据。推送设置响应数据只包括：公开协议产品下设备数据点、私有协议产品下设备传感器数据。如果没有设置【推送时间限制】或【推送数据限制】，将默认数据实时推送

提交

图4-3 第三方开发平台的配置界面

当用户成功在 OneNET 平台配置了第三方开发平台服务器地址后，点击“提交”按钮后，OneNET 平台会向第三方应用平台发起 URL&Token 验证请求，验证通过且收到第三方返回的验证消息后，则第三方应用平台与 OneNET 平台已建立连接。之后，OneNET 平台在收到设备数据时，会将设备数据推送到已配置的第三方应用平台。

用户数据接收服务程序接收到 OneNET 平台发送的“URL&Token”验证请求，请求的格式形如：<http://ip:port/test?msg=xxx&nonce=xxx&signature=xxx>，蓝色字体部分为用户在 OneNET 平台配置的 URL，常规的验证流程如下：

(1) 在请求中获取参数“nonce”、“msg”的值，将“token”（此为页面配置参数 token 的值）、“nonce”、“msg”的值计算 MD5(token+nonce+msg) 加密值，并且编码为 Base64 字符串值。

(2) 将上一步中 Base64 字符串值通过 URL Decode 计算后的值与请求参数

“signature” 的值进行对比，如果相等则表示 token 验证成功。

(3) 如果 token 验证成功，向 OneNET 平台返回 “msg” 参数值，否则返回其他值。

说明：如果用户不想验证 Token 的有效性，可以忽略以上步骤，并直接向 OneNET 平台返回 “msg” 参数值。

## 4.2 OneNET 平台数据推送

平台以 HTTPS POST 请求形式向第三方应用平台注册地址推送数据，推送数据相关信息以 JSON 串的形式置于 HTTPS 请求中的 body 部分。

第三方应用平台在接收数据时，会接收到数据的明文消息或者密文消息。明文消息中以“type”的数值区分不同的消息，而密文消息与明文消息的区别是“msg”部分经过了加密处理。下面将分别对明文消息、密文消息以及消息相关字段和密文的加密算法进行说明。

### 4.2.1 明文消息

明文消息有两种消息类型，分别为数据点消息和设备上下线消息。而数据点消息又分为单条数据点消息和数据点消息批量形式，数据点消息批量形式中的不同数据点是以不同时间戳区分的。

#### (1) 数据点消息(type=1)

```
{
  "msg": {
    "type": 1,
    "dev_id": 2016617,
    "ds_id": "datastream_id",
    "at": 1466133706841,
    "value": 42
  },
  "msg_signature": "message signature",
  "nonce": "abcdefgh"
}
```

#### (2) 数据点消息批量形式(type=1)

```
{
  "msg": [
```

```
{
  "type": 1,
  "dev_id": 2016617,
  "ds_id": "datastream_id", //
  "at": 1466133706841,
  "value": 42
},
{
  "type": 1,
  "dev_id": 2016617,
  "ds_id": "datastream_id",
  "at": 1466133706842, "value": 43
},
...
],
"msg_signature": "message signature",
"nonce": "abcdefgh"
}
```

### (3) 设备上下线消息(type=2)

```
{
  "msg": {
    "type": 2,
    "dev_id": 2016617,
    "status": 0,
    "login_type": 10,
    "at": 1466133706841,
  },
  "msg_signature": "message signature",
  "nonce": "abcdefgh"
}
```

## 4.2.2 密文消息

密文消息中的“enc\_msg”是由明文消息中的“msg”采用 AES 加密算法而来。

```
{
  "enc_msg": "xxxx",
  "msg_signature": "message signature",
  "nonce": "abcdefgh"
}
```

### 4.2.3 消息相关字段说明

表 4-2 消息相关字段说明

字段	字段说明
type	标识数据类型, 当前版本范围[1,5]
dev_id	设备 ID
ds_id	公开协议中的数据流 ID, 其格式为 objectID_instanceID_resourceID
at	平台时间戳,单位 ms
value	具体数据部分,为设备上传至平台或触发的相关数据
status	设备上下线标识 0-下线, 1-上线
login_type	设备登陆协议类型 1: EDP, 2: nwx, 3: JTEXT, 5: JT808, 6: MODBUS, 7: MQTT, 8: gr20, 10: NB-IoT
cmd_type	命令响应的类型 1: 设备收到 cmd 的 ACK 响应信息, 2: 设备收到 cmd 的 Confirm 响应信息。
cmd_id	命令 ID
msg_signature	消息摘要
nonce	用于计算消息摘要的随机串
enc_msg	加密密文消息体,对明文 JSON 串(msg 字段)的加密

### 4.2.4 加密算法详述

平台基于 AES 算法提供加解密技术, 具体阐述如下:

- EncodingAESKey 即消息加解密 Key 的 BASE64 编码形式, 长度固定为 43 个字符, 从 a-z,A-Z,0-9 共 62 个字符中选取。由服务开启时填写, 后也可申请修改。
- AES 密钥计算为  $AESKey = Base64\_Decode(EncodingAESKey + "=")$ , EncodingAESKey 尾部填充一个字符的 "=", 用 Base64\_Decompose 生成 32 个字节的 AESKey。
- AES 采用 CBC 模式, 密钥长度为 32 个字节(256 位), 数据采用 PKCS#7 填充, 初始化 iv 向量取密钥前 16 字节; PKCS#7: K 为密钥字节数(采用 32), buf 为待加密的内容, N 为其字节数。Buf 需要被填充为 K 的



整数倍。在 buf 的尾部填充(K-N%K)个字节，每个字节的内容 是(K-N%K)。具体详见：[//tools.ietf.org/html/rfc2315](http://tools.ietf.org/html/rfc2315)

- BASE64 采用 MIME 格式，字符包括大小写字母各 26 个，加上 10 个数字，和加号“+”，斜杠“/”，一共 64 个字符，等号“=”用作后缀填充；
- 出于安全考虑，平台网站提供了修改 EncodingAESKey 的功能（在 EncodingAESKey 可能泄漏时进行修改，对应上第三方平台申请时填写的接收消息的加密对称密钥），所以建议保存当前的和上一次的 EncodingAESKey，若当前 EncodingAESKey 生成的 AESKey 解密失败，则尝试用上一次的 AESKey 的解密。
- 平台的加密消息部分为 `enc_msg=Base64_Encode( AES_Encrypt [random(16B)+msg_len(4B)+msg] )`，即以 16 字节随机字节串拼接 4 字节表示消息体长度的字节串(此处 4 字节长度表示为网络字节序),再加上消息本身的字节串作为 AES 加密的明文，再以 AES 算法对明文进行加密生成密文，最后对密文进行 BASE64 的编码操作生成加密消息体。
- 对加密消息体的解密流程为：1) 首先进行加密消息体的 BASE64 解码操作，`aes_msg=Base64_Decode(enc_msg)`；2) 对获取的解码内容以 AES 算法进行解密操作，获取明文部分，`plain_msg=AES_Decrypt(aes_msg)`，解密中使用的密钥由 EncodingAESKey 计算得来，使用的初始化 iv 向量为计算出的 aes 密钥的前 16 字节；3) 去掉 plain\_msg 的前 16 字节，再以前 4 字节取出消息体长度，根据消息体长度获取真实的消息部分（推荐以消息体长度获取真实消息，以兼容 plain\_msg 未来可能出现的结构变更）。表示加密传输的数据，后两字段与明文传输一致。

### 4.3 API 接口

第三方开发平台应用层通过 Rsetful API 的方式和 OneNET 平台进行交互对接，实现命令的下发、数据的读写以及相关业务的交互。

针对 LWM2M 协议，目前主要开发了以下 API 用来实现第三方开发平台向 OneNET 平台请求读写执行以及设备管理请求，具体见下表 4-2。下面分别对每一种 API 进行说明。



表 4-3 API 功能说明

API 类别		操作对象	API 名称
需要终端响应	只需平台响应		
	✓	设备	<a href="#">创建设备</a>
	✓		<a href="#">查看单个设备信息</a>
	✓		<a href="#">删除设备</a>
	✓		<a href="#">根据 imei 查询设备信息</a>
✓		资源	<a href="#">读设备资源</a>
✓			<a href="#">写设备资源</a>
✓		命令	<a href="#">下发命令-执行</a>
	✓	列表	<a href="#">获取资源列表</a>
	✓	订阅	<a href="#">订阅查看某个设备</a>
✓		离线命令	<a href="#">读数据</a>
✓			<a href="#">写数据</a>
✓			<a href="#">命令</a>
	✓		<a href="#">查看指定设备离线命令列表</a>
	✓		<a href="#">查看设定离线命令详情</a>
	✓		<a href="#">取消离线命令</a>
	✓		<a href="#">新增</a>
	✓	触发器	<a href="#">更新</a>
	✓		<a href="#">查看</a>
	✓		<a href="#">删除</a>
	✓		<a href="#">批量查询设备状态</a>
	✓	批量查询	<a href="#">批量查询设备最新数据</a>
	✓	数据点	<a href="#">查看数据点</a>

### 4.3.1 创建设备

表 4-4

HTTP 方法	POST
URL	http://api.heclouds.com /devices
HTTP 头部	api-key:xxxx-ffff-zzzzz, 必须为 <b>MasterKey</b> Content-Type:application/json
HTTP 内容	{ "title": "mydevice", //设备名 "desc": "some description", //设备描述 (可选)

	<pre> "tags": ["china","mobile"],//设备标签（可选，可为一个或多个） "protocol": "LWM2M", //接入协议 "location": {"lon": 106, "lat": 29, "ele": 370}, //设备位置{"纬度", "精度", "高度"}（可选） "private": true false, //设备私密性（可选，默认为 true） "auth_info": {"xxxxxxxxxxxx": "xxxxxxxxxxxxxxxx"}, //NB-IoT 设备：{"imei 码": "imsi 码"}, imei（不超过 17 位）和 imsi（不超过 16 位）都由数字或者字母组成 "obsv": true false, //是否订阅设备资源，默认为 true "other": {"version": "1.0.0", "manu": "china mobile"} //其他信息（可选，JSON 格式，可自定义） } </pre>
HTTP 响应 响应消息内容	<pre> {   "errno": 0, //无错误   "error": "succ",   "data": {     "device_id": "233444"//平台分配唯一 ID   } } </pre>
说明	<p>（1）响应消息中 <b>errno</b> 表示错误码，<b>error</b> 表示错误原因，如果创建设备失败，则没有 <b>device_id</b> 字段；</p> <p>（2）NB CoAP 设备 <b>auth_info</b> 中 <b>imei</b>（不超过 17 位）和 <b>imsi</b>（不超过 16 位）均由数字或者字母组成。</p>

### 4.3.2 查看单个设备信息

表 4-5

HTTP 方法	GET
URL	http://api.heclouds.com/devices/<device_id>
HTTP 头部	api-key:xxxx-ffff-zzzzz, 必须为可查看该设备的 Key Content-Type:application/json
请求返回	<pre> {   "errno": 0,   "error": "succ",   "data": {     "online": true false,     "protocol": "HTTP"     "title": "my device1",     "desc": "some description,ex:url",     "create_time": "xx-xx-xx 10:22:22",     "obsv": true, //NB CoAP 协议设备才有该字段值     "private": true false, //设备关联的图像或二进制数据   } } </pre>

	<pre> "binary":[{     "index":"FJWOPN9023899",//二进制数据索引     "at":"2014-10-23 20:22:22",//上传时间     "size":2333(字节),//二进制数据大小     "desc":"binary description" }] "tags":["Tag1","Tag2"], "location":{"ele":370000,"lat":17.609991828964787,"lon":177.0340299 68273}, "datastreams"://数据流 [ {     "create_time":"2014-10-23 20:22:22"     "id":"datastream_id1",     "unit":"celsius",     "unit_symbol":"C",     "uuid":" 231a5aa9-9de4-5f2e-9e0f-015181c98429" }]//end data_streams }         </pre>
--	---

### 4.3.3 删除设备

表 4-6

HTTP 方法	DELETE
URL	http://api.heclouds.com/ devices /<device_id>
HTTP 头部	api-key:xxxx-ffff-zzzzz //可为设备级别的 Key Content-Type:application/json
HTTP 内容	无
请求返回	<pre> {     "errno": 0,     "error": "succ" }         </pre>
说明	响应消息中 <b>errno</b> 表示错误码， <b>error</b> 表示错误原因。

### 4.3.4 根据 IMEI 查询设备信息

表 4-7

HTTP 方法	GET
URL	http://api.heclouds.com/ devices /getbyimei

HTTP 头部	api-key:xxxx-ffff-zzzzz //必须为 masterkey
请求返回	<pre>{   "errno": 0,   "error": "succ",   "data": {     "id": 1232, //设备 id     "online": true false,     "title": "my device1",     "desc": "some description ",     "create_time": "2018-05-12 10:22:22",     "private": true false,   } }</pre>
说明	响应消息中 <b>errno</b> 表示错误码， <b>error</b> 表示错误原因。

### 4.3.5 读设备资源

表 4-8

HTTP 方法	GET
URL	http://api.heclouds.com/nbiot
HTTP 头部	api-key:xxxx-ffff-zzzzz, 必须为 masterKey
HTTP 参数	<p>"imei": 121, // nbiot 设备的身份码, 和 ep_name 两者必填其一</p> <p>"ep_name": 121, // nbiot 设备的身份码, 和 imei 两者必填其一</p> <p>"obj_id": 1212, //设备的 object id, 对应到平台模型中为数据流 id, 必填</p> <p>"obj_inst_id": 1212, // nbiot 设备 object 下具体一个 instance 的 id ,对应到平台模型中数据点 key 值的一部分, 选填</p> <p>"res_id": 2122 // nbiot 设备的资源 id, 选填</p>
成功返回	<pre>{   "errno": 0,   "error": "succ",   "data": [ {     "obj_inst_id": 123,     "res": [       {         "res_id": 1234,         "val": Object //可为 boolean、string、long、double 类型数据       },       .....     ]   } ], }</pre>

	<pre>         .....     ] }</pre>
说明	<p>1、obj_instance_id 不存在的时候，resource_id 必不存在。</p> <p>2、响应消息中 errno 表示错误码，error 表示错误原因。</p>

### 4.3.6 写设备资源

表 4-9

HTTP 方法	POST
URL	http://api.heclouds.com/nbiot
HTTP 头部	api-key:xxxx-ffff-zzzzz, 必须为 masterKey Content-Type:application/json
HTTP 参数	<pre> "imei":121, // nbiot 设备的身份码, 和 ep_name 两者必填其一 "ep_name":121, // nbiot 设备的身份码, 和 imei 两者必填其一 "obj_id":1212, // 设备的 object id, 对应到平台模型中为数据流 id, 必填 "obj_inst_id": 1212, // nbiot 设备 object 下具体一个 instance 的 id , 对应到平台模型中数据点 key 值的一部分, 必填 "mode": 1 2 // write 的模式, 必填           </pre>
HTTP 内容	<pre> {     "data": [{         "res_id":123,         "val":Object //可为 boolean、string、long、double     },     {...},     ...     ] }</pre>
成功返回	<pre> {     "errno": 0,     "error": "succ", }</pre>
说明	<p>1、“mode”取值为 1, 表示 replace, 意为替换指定的 instance 或者 resource 的值 ;“mode”取值为 2, 表示 partial update, 意为只更新给定的 resource 或者 resource instance 的值。</p> <p>2、HTTP 内容部分必须存在。</p> <p>3、响应消息中 errno 表示错误码，error 表示错误原因。</p>

### 4.3.7 下发命令

表 4-10

HTTP 方法	POST
URL	http://api.heclouds.com/nbiot/execute
HTTP 头部	api-key:xxxx-ffff-zzzzz, 必须为 masterKey Content-Type:application/json
HTTP 参数	"imei":121, // nbiot 设备的身份码, 和 ep_name 两者必填其一 "ep_name":121, // nbiot 设备的身份码, 和 imei 两者必填其一 "obj_id":1212, // nbiot 设备的 object id, 对应到平台模型中为数据流 id, 必填; "obj_inst_id": 1212, // 设备 object 下具体一个 instance 的 id, 对应到平台模型中数据点 key 值的一部分, 必填; "res_id": 123 // nbiot 设备的资源 id, 必填。
HTTP 内容	{ "args": "ping"// 字符串 }
成功返回	{ "errno": 0, "error": "succ", }
说明	1、HTTP 内容部分选填。 2、响应消息中 <b>errno</b> 表示错误码, <b>error</b> 表示错误原因。

#### 4.3.8 获取资源列表

表 4-11

HTTP 方法	GET
URL	http://api.heclouds.com/nbiot/resources
HTTP 头部	api-key:xxxx-ffff-zzzzz, 必须为 masterKey
HTTP 参数	"imei":121, // nbiot 设备的身份码, 必填 "obj_id":3200 //可选
成功返回	{ "errno": 0, "error": "succ", "data": { "total_count":123, "item": [ { "obj_id":3200, "instances": [

	<pre> {   "inst_id":0,   "resources":[5500,5050] }, {.....}, ..... ] }, {.....}, ..... ] } </pre>
说明	<p>1、HTTP 内容部分选填。</p> <p>2、响应消息中 <b>errno</b> 表示错误码，<b>error</b> 表示错误原因。</p>

### 4.3.9 订阅

表 4-12

HTTP 方法	POST
URL	http://api.heclouds.com/nbiot/observe
HTTP 头部	api-key:xxxx-ffff-zzzzz, 必须为 masterKey
HTTP 参数	<p>"imei":小于 17 位数字组合, // nbiot 设备的身份码, 必填</p> <p>"cancel":true false, //true 为取消订阅, false 为订阅, 必填</p> <p>"obj_id":xxxx, // nbiot 设备的 object id, 对应到平台模型中为数据流 id, 必填</p> <p>"obj_inst_id": xxxx, //设备 object 下具体一个 instance 的 id , 对应到平台模型中数据点 key 值的一部分, 选填</p> <p>"res_id": 123 // nbiot 设备的资源 id, 选填</p> <p>"pmin":11, //上传数据的最小时间间隔 int 类型, 默认为 0, 此时有数据就上传</p> <p>"pmax":123, //上传数据的最大时间间隔, int 类型, 可选</p> <p>"gt":12, //当数据大于该值上传, double 类型, 可选</p> <p>"lt":233, // 当数据小于该值上传, double 类型, 可选</p> <p>"st":12 //当两个数据点相差大于或者等于该值上传, double 类型, 可选</p>
成功返回	<pre> {   "errno": 0,   "error": "succ", } </pre>
说明	<p>1、pmin 和 pmax 都存在时, pmax&gt;=pmin 且需大于 0。</p> <p>2、lt &lt; gt, 并且 lt + 2*st &lt; gt。</p> <p>3、如果有 gt、lt、st, 则 res_id 必填。</p> <p>4、cancel 为非 true false 将被默认为 false。</p> <p>5、响应消息中 <b>errno</b> 表示错误码，<b>error</b> 表示错误原因。</p>

### 4.3.10 离线命令

在设备离线状态下，我们可以调用离线命令对设备进行操作。其原理是用户设置命令开始生效时间和命令过期时间，在此时间段内，OneNET 平台对离线命令进行缓存，当终端设备上线时，平台下发缓存的命令到终端设备。如果超过此有效时间段，平台将清除离线命令缓存。

下面分别对各个离线命令进行说明。

#### 4.3.10.1 读数据/Read

表 4-13

HTTP 方法	GET
URL	http://api.heclouds.com /nbio/offline
HTTP 头部	api-key:xxxx-ffff-zzzzz //必须为 MasterKey
HTTP 参数	"imei":121, // nbio 设备的身份码，必填 "obj_id":1212, //设备的 object ID 对应到平台模型中为数据流 id，必填 "obj_inst_id": 1212, // nbio 设备 object 下具体一个 instance 的 id，对应到平台模型中数据点 key 值的一部分，选填 "res_id": 2122, // nbio 设备的资源 id，选填 "valid_time": "2018-03-08T17:30:00", //命令开始生效时间，必填且大于当前时间 "expired_time": "2018-03-09T17:30:00", //命令过期时间,必填且大于 valid_time "retry":3 // [3-10]之间，表示失败重试次数（等待下一次设备 update 或者上线）,必填
HTTP 响应消息内容	<pre>{   "errno": 0,   "error": "succ",   "data": {     "uuid": "42742677-adc3-54ca-83a1-5aaaf71482f8" //离线命令 uuid   } }</pre>
说明	1、响应消息中 <b>errno</b> 表示错误码， <b>error</b> 表示错误原因。

#### 4.3.10.2 写数据/Write

表 4-14

HTTP 方法	POST
URL	http://api.heclouds.com /nbio/offline
HTTP 头部	api-key:xxxx-ffff-zzzzz, //必须为 MasterKey Content-Type:application/json
HTTP 参数	"imei":121, // nbio 设备的身份码，必填



	<p>"obj_id":1212, //设备的 object id, 对应到平台模型中为数据流 id, 必填</p> <p>"obj_inst_id": 1212, // nbio 设备 object 下具体一个 instance 的 id , 对应到平台模型中数据点 key 值的一部分, 必填</p> <p>"mode": 1 2 // write 的模式, 必填</p> <p>"valid_time": "2018-03-08T17:30:00", //命令开始生效时间, 必填大于当前时间</p> <p>"expired_time": "2018-03-09T17:30:00", //命令过期时间, 必填且大于 valid_time</p> <p>"retry":3 // [3-10] 之间, 表示失败重试次数 (等待下一次设备 update 或者上线), 必填</p>
HTTP 内容	<pre>{   "data": [{     "res_id":123,     "type":1, //可选, 目前只支持为 1, 此时数据为十六进制字符串     "val":Object //可为 boolean、string、long、double   },   {...},   ... ]</pre>
HTTP 响应消息内容	<pre>{   "errno": 0,   "error": "succ",   "data": {     "uuid": "42742677-adc3-54ca-83a1-5aaaf71482f8" //离线命令 uuid   } }</pre>
说明	<p>1、响应消息中 <b>errno</b> 表示错误码, <b>error</b> 表示错误原因。</p>

#### 4.3.10.3 执行/Execute

表 4-15

HTTP 方法	POST
URL	http://api.heclouds.com/nbio/execute/offline
HTTP 头部	api-key:xxxx-ffff-zzzzz //必须为 masterKey
HTTP 参数	<p>"imei":121, //设备模组 IMEI, 必填</p> <p>"obj_id":1212, // nbio 设备的 object id, 对应到平台模型中为数据流 id, 必填</p> <p>"obj_inst_id": 1212, //设备 object 下具体一个 instance 的 id, 对应到平台模型中数据点 key 值的一部分, 必填</p> <p>"res_id": 123 // nbio 设备的资源 id, 必填</p> <p>"valid_time": "2018-03-08T17:30:00", //命令开始生效时间戳, 必填且大于当前时间</p> <p>"expired_time": "2018-03-09T17:30:00", //命令过期时间戳, 必填且大于</p>

	valid_time “retry”:3 // [3 10]之间, 表示失败重试次数 (等待下一次设备 update 或者上线), 必填
HTTP 内容	<pre>{   "args": "ping"//字符串 }</pre>
成功返回	<pre>{   "errno": 0,   "error": "succ",   "data": {     "uuid": "42742677-adc3-54ca-83a1-5aaaf71482f8"//离线命令 uuid   } }</pre>
说明	1、HTTP 内容部分选填。 2、响应消息中 <b>errno</b> 表示错误码, <b>error</b> 表示错误原因。

#### 4.3.10.4 查看指定设备离线命令列表

表 4-16

HTTP 方法	GET
URL	http://api.heclouds.com/nbiot/offline/history
HTTP 头部	api-key:xxxx-ffff-zzzzz //必须为 masterKey
URL 参数	imei="121212", //设备的 imei 号 start="2016-08-05T08:00:00", //指定开始时间, 必选 end="2016-08-06T08:00:00", //指定结束时间, 可选 page=12, //当前页数 per_page=10, //每一页多少条 sort=DESC ASC //时间排序方式, DESC:降序, ASC 升序, 默认降序
成功返回	<pre>{   "errno": 0,   "error": "succ"   "data": {     {       "count": 100,       "items": [         {           "cmd_uuid": "f6869ecb-3dc1-5374-9be0-4fb961f8af3c ",           "type": "READ",           "create_time": "2017-08-28T11:34:58",           "valid_time": "2017-08-28T11:34:58", </pre>

	<pre> "expired_time": "2017-08-28T11:34:58", "send_time": "2017-08-28T11:34:58", "send_status": 5, //命令下发成功 "confirm_time": "2017-08-28T11:34:58", "confirm_status": "SUCCESS" {...} ] } } </pre>
备注	<p>1、send_time/confirm_time/sid 可能不存在</p> <p>2、send_status 字段值代表意义如下：</p> <ul style="list-style-type: none"> <li>‘1’：命令等待（wait）</li> <li>‘2’：命令取消（cancel）</li> <li>‘3’：命令已发往设备（send）</li> <li>‘4’：命令过期（expired）</li> <li>‘5’：命令下发成功（success）</li> <li>‘6’：命令下发失败（failed）</li> <li>‘7’：命令响应超时（timeout）</li> <li>‘8’：其他未知错误（undefined）</li> </ul>
说明	<p>1、响应消息中 <b>errno</b> 表示错误码，<b>error</b> 表示错误原因。</p>

#### 4.3.10.5 查看指定离线命令详情

表 4-17

HTTP 方法	GET
URL	http://api.heclouds.com/nbiot/offline/history/<uuid>
URL 参数	imei="imei"
HTTP 头部	api-key:xxxx-ffff-zzzzz //必须为 masterKey
成功返回	<pre> {   "errno": 0,   "error": "succ"   "data":   {     "type": "READ",     "args": {       // type 类型接口 http 参数     }   }   "content": {     //type 类型接口 http 内容   },   "create_time": "2017-08-28 11:34:58", </pre>

	<pre> "valid_time": "2017-08-28 11:34:58", "expired_time": "2017-08-28 11:34:58", "send_time": "2017-08-28 11:34:58", "send_status": 5, //命令下发成功 "confirm_time": "2017-08-28 11:34:58", "confirm_status": "SUCCESS", //命令响应结果, 如果不是 success, 则没有 cmd_result "cmd_result": { //响应内容     // type 类型接口成功返回的 json 结果 } </pre>
备注	<p>send_status 字段值代表意义如下:</p> <ul style="list-style-type: none"> <li>'1': 命令等待 (wait)</li> <li>'2': 命令取消 (cancel)</li> <li>'3': 命令已发往设备 (send)</li> <li>'4': 命令过期 (expired)</li> <li>'5': 命令下发成功 (success)</li> <li>'6': 命令下发失败 (failed)</li> <li>'7': 命令响应超时 (timeout)</li> <li>'8': 其他未知错误 (undefined)</li> </ul>
说明	<p>1、响应消息中 <b>errno</b> 表示错误码, <b>error</b> 表示错误原因。</p>

#### 4.3.10.6 取消离线命令

表 4-18

HTTP 方法	PUT
URL	http://api.heclouds.com/nbiot/offline/cancel/<uuid>
URL 参数	imei="imei"
HTTP 头部	api-key:xxxx-ffff-zzzzz //必须为 masterKey
成功返回	<pre> {   "errno": 0,   "error": "success", } </pre>
备注	只有当命令处于 wait 状态才能取消, 也就是命令 valid_time 没到之前
说明	<p>1、响应消息中 <b>errno</b> 表示错误码, <b>error</b> 表示错误原因。</p>

#### 4.3.11 触发器

触发器的含义: 当指定范围内的数据点满足触发条件的要求时, 会向 url 参数指定的地址发送 post 请求。触发器有三种工作触发模式, 分别为新增、更新



#### 4.3.11.2 更新

表 4-20

HTTP 方法	PUT
URL	http://api.heclouds.com/triggers/<trigger_id>
HTTP 头部	api-key:xxxx-ffff-zzzzz //需要 master Key Content-Type:application/json
HTTP 内容	{ "title": "wen du jian kong", "url": "http://xx.bb.com", "type": "> > = < < = in out exp change frozen live", "threshold": 100 }
请求返回	{ "errno": 0, "error": "succ" }
备注	<p>1、若要修改触发条件，必须同时设置 type 和 threshold。</p> <p>2、Type 字段说明：</p> <p>（1）type 为 &gt; &gt; = &lt; &lt; = 时，threshold 必须为数值。</p> <p>（2）type 为 inout 时，threshold 设置为 {“lolmt”:40,“uplmt”:52}，表示数据流的值首次进入或离开闭区间[40,52]时触发。</p> <p>（3）type 为 exp 时,threshold 设置为字符串类型的条件表达式，\$val[0]表示第一个数据流的当前值，\$val[1]为第二个…。例如，第一个数据流上报数据点格式为：{“temperature”: 22,“humidity”: 56},若需设置温度大于 30，且湿度小于 33 时触发告警，则 threshold 中条件表达式可设置为： “\$val[0][‘temperature’] &gt; 30 &amp;&amp; \$val[0][‘humidity’] &lt; 33”。</p> <p>（4）type 为 change 时，threshold 参数不用传递；当上传的值有改变时触发告警。</p> <p>（5）type 为 frozen 时，threshold 为数值，指定多少秒内未上报数据触发告警，同时被监控对象进入 frozen 状态。</p> <p>（6）type 为 live 时，threshold 要传递；(要传值大于 0 的数字)被监控对象在 frozen 状态下收到上报的数据点时，触发告警。</p>



说明

 1、响应消息中 **errno** 表示错误码，**error** 表示错误原因。

## (2) 批量查看

表 4-22

HTTP 方法	GET
URL	http://api.heclouds.com/triggers
URL 参数	title = name //指定触发器名称 page = 1 //指定页码, 可选 per_page = 10 //指定每页输出个数, 可选, 默认 10, 最多 100
HTTP 头部	api-key:xxxx-ffff-zzzzz
请求返回	<pre>{   "errno": 0,   "error": "succ",   "data": {     "total_count": 600,     "per_page": 30,     "page": 1,     "triggers": [{       "id": trigger_id1,       "title": "wen du jian kong",       "url": "xx.bb.com",       "type": "&gt; &gt;=&lt; &lt;=&lt; ==in out exp change frozen live",       "threshold": 100,       //       "invalid": true, //触发器是否已失效, 详见备注说明       "create_time": XXXXXXXX     }]   } }</pre>
备注	<p>1、invalid 字段说明：触发器是否已失效，若为“true”说明触发器失效。（触发器失效是指触发器所依赖的数据流或者设备被删除了，此时触发器永远不会有效触发事件）。</p> <p>2、Type 字段说明：</p> <p>（1）type 为 &gt; &gt;=&lt; &lt;=&lt; == 时，threshold 必须为数值。</p> <p>（2）type 为 inout 时，threshold 设置为 {“lolmt”:40,“uplmt”:52}，表示数据流的值首次进入或离开闭区间[40,52]时触发。</p> <p>（3）type 为 exp 时,threshold 设置为字符串类型的条件表达式，\$val[0]表示第一个数据流的当前值，\$val[1]为第二个…。例如，第一个数据流上报数据点格式为：{“temperature”: 22,“humidity”: 56},若需设置温度大于 30，且湿度小于 33 时触发告警，则 threshold 中条件表达式可设置为：“\$val[0][‘temperature’] &gt; 30 &amp;&amp; \$val[0][‘humidity’] &lt; 33”。</p>



	<p>(4) type 为 change 时, threshold 参数不用传递; 当上传的值有改变时触发告警。</p> <p>(5) type 为 frozen 时, threshold 为数值, 指定多少秒内未上报数据触发告警, 同时被监控对象进入 frozen 状态。</p> <p>(6) type 为 live 时, threshold 要传递; (要传值大于 0 的数字)被监控对象在 frozen 状态下收到上报的数据点时, 触发告警。</p>
说明	1、响应消息中 <b>errno</b> 表示错误码, <b>error</b> 表示错误原因。

#### 4.3.11.4 删除触发器

表 4-23

HTTP 方法	DELETE
URL	http://api.heclouds.com/triggers/<trigger_id>
HTTP 头部	api-key:xxxx-ffff-zzzzz //需要 master Key Content-Type:application/json
HTTP 内容	无
请求返回	<pre>{   "errno": 0,   "error": "succ" }</pre>
说明	1、响应消息中 <b>errno</b> 表示错误码, <b>error</b> 表示错误原因。

#### 4.3.12 批量查询设备状态

表 4-24

HTTP 方法	GET
URL	http://api.heclouds.com/devices/status
URL 参数	devIds=1221,12233,1123 //设备 id 用逗号隔开, 限制 1000 个设备
HTTP 头部	api-key:xxxx-ffff-zzzzz //可为设备级别的 Key Content-Type:application/json
HTTP 内容	无
请求返回	<pre>{   "errno": 0,   "error": "succ",   "data": {     "total_count": 121,     "devices": [       {         "id": 12323, </pre>

	<pre> "title": "daf", "online": false }, {....}, ..... ] } } </pre>
说明	1、响应消息中 <b>errno</b> 表示错误码， <b>error</b> 表示错误原因。

### 4.3.13 批量查询设备最新数据

表 4-25

HTTP 方法	GET
URL	http://api.heclouds.com/ devices /datapoints
UTL 参数	devIds=1221,12233,1123 //设备 id 用逗号隔开, 限制 500 个设备
HTTP 头部	api-key:xxxx-ffff-zzzzz //可为设备级别的 Key Content-Type:application/json
HTTP 内容	无
请求返回	<pre> {   "errno": 0,   "error": "succ",   "data":   {     "devices": [       {         "id": 12323,         "title": "daf",         "datastreams": [           {             "id": "temperature",             "at": "2017-02-12 10:22:22",             "value": 12           },           {....},           .....         ]       },       {....},       .....     ]   } } </pre>

	<pre>       }     } </pre>
说明	1、响应消息中 <b>errno</b> 表示错误码， <b>error</b> 表示错误原因。

#### 4.3.14 查看数据点

表 4-26

HTTP 方法	GET
URL	http://api.heclouds.com/devices/<device_id>/datapoints
URL 参数	<p>接口参数包括必选的模式选项和可选的公共参数。</p> <p>可选的公共参数：</p> <ol style="list-style-type: none"> <li>1.datastream_id:涉及的数据流: datastream_id=&lt;datastream_id&gt;,多个数据流之间用逗号分隔；</li> <li>2.start:表示提取数据点的开始时间，格式为 2015-01-10T08:00:35</li> <li>3.end:表示提取数据点的结束时间，格式为 2015-01-10T08:00:35</li> <li>4.limit:限定本次请求最多返回的数据点的数量，取值&gt;0，&lt;=6000</li> <li>5.cursor:指定本次请求继续从 cursor 位置开始提取数据</li> </ol>
HTTP 头部	api-key:xxxx-ffff-zzzzz //需要设备级别的 key
请求返回	<pre> {   "errno": 0,   "error": "succ",   "data": {     "cursor": "XXAABBCCDD",     "datastreams": [       {         "id": "temperature",         "datapoints": [           { "at": "xxxx-xx-xx 10:22:22", "value": 42 },           { "at": "xxxx-xx-xx 10:22:22", "value": 84 }         ],         "statistic": { //数据流统计信息           "at": "xxxx-xx-xx 10:22:22", //求最值时返回的一个最值时间点           "value": 42         }       },       {         "id": "key",         "datapoints": [           { "at": "xxxx-xx-xx 10:22:22", "value": { ... } },           { "at": "xxxx-xx-xx 10:22:22", "value": { ... } }         ]       }     ]   } } </pre>

	<pre>         ]       }, {...}]     }   } } </pre>
备注	<p>使用方法说明：</p> <p>(1) 不携带任何参数，直接调用。则返回本设备所存在的所有数据流中最新的数据。如果这个设备有三个数据流，则返回这三个数据流中每个数据流中最后一条数据。<a href="http://api.heclouds.com/devices/564280/datapoints">http://api.heclouds.com/devices/564280/datapoints</a> 表示查询设备 id 为 564280 对应的所有数据流的最新数据点。</p> <p>(2) 不携带数据流 id 参数，携带 limit 参数时，会返回该设备每个数据流最多 limit 条数据。比如：</p> <p><a href="http://api.heclouds.com/devices/564280/datapoints?limit=10">http://api.heclouds.com/devices/564280/datapoints?limit=10</a>，假设说设备 564280 有三个数据流，那么这三个流中每个流都会最多返回 10 个数据点。</p> <p>(3) 要查看某一条数据流数据，在上述 URL 中增加参数 datastream_id。</p> <p>(4) 其中 datastream_id 等于 obj_id_obj_inst_id_res_id，如 obj_id:3200, obj_inst_id:0, res_id:5501，那么这个 datastream_id 就为 3200_0_5501。那么如下所示：<a href="http://api.heclouds.com/devices/564280/datapoints?datastream_id=3200_0_5501">http://api.heclouds.com/devices/564280/datapoints?datastream_id=3200_0_5501</a>，datastream_id=后面的数据流的名称不加双引号，会返回数据流 3200_0_5501 里时间最新的一个数据点。</p> <p>(5) 用 <a href="http://api.heclouds.com/devices/564280/datapoints?datastream_id=3200_0_5501&amp;limit=10">http://api.heclouds.com/devices/564280/datapoints?datastream_id=3200_0_5501&amp;limit=10</a> 会返回数据流“温度 2”中最近的 10 个数据点。</p> <p>要查看某一条数据流在某个时间范围内的数据，可以在增加 start 和 end 参数。<a href="http://api.heclouds.com/devices/564280/datapoints?datastream_id=3200_0_5501&amp;start=2013-05-12T17:12:33&amp;end=2013-06-12T17:22:33">http://api.heclouds.com/devices/564280/datapoints?datastream_id=3200_0_5501&amp;start=2013-05-12T17:12:33&amp;end=2013-06-12T17:22:33</a></p> <p>表示查询本设备对应的数据流在 2013 年 5 月 12 日 17 点 12 分 33 秒到 17 点 22 分 33 秒的数据点。注意如果 start 参数存在，end 参数不存在，表示取 start 后的所有数据；如果 start 不存在，end 存在，设备云会忽略 end 参数。注意：日期时间格式必须为示例中的格式：如 2013-05-12T17:12:33，日期和时间之间用大写字母 T 分隔。start 和 end 之间的时间间隔最大为 1 年，超过一年会忽略。</p> <p>(7) 如果指定了 start 参数，则可能返回的数据点的数目会很多，此时默认会返回最多 100 个数据点。可以使用 limit 参数，设定返回多少个数据点，最大为 6000 个数据点。当实际的数据点数目多于 limit 限定的数目时，返回的 json 串中会有一个 cursor 字段，下一次请求的命令行中可以携带此 cursor 字段表示接着遍历本数据流。此 cursor 字段标识上次取出数据点后下一个数据点的位置。</p>

#### 4.3.14.1 查看数据点--示例一

表 4-27

请求 URL	<a href="http://api.heclouds.com/devices/564280/datapoints?datastream_id=3200_0_55012&amp;start=2015-11-30T17:12:33&amp;end=2015-12-">http://api.heclouds.com/devices/564280/datapoints?datastream_id=3200_0_55012&amp;start=2015-11-30T17:12:33&amp;end=2015-12-</a>
--------	---

	01T17:22:33
响应结果	<pre>{   "errno": 0,   "data": {     "count": 26,     "datastreams": [       {         "datapoints": [           {             "at": "2015-12-01 17:06:23.535",             "value": "39"           },           {             "at": "2015-12-01 17:06:34.488",             "value": "39"           },           {             "at": "2015-12-01 17:06:35.935",             "value": "39"           },           {             "at": "2015-12-01 17:09:35.918",             "value": "20"           },           {             "at": "2015-12-01 17:09:55.915",             "value": "30"           },           {             "at": "2015-12-01 17:10:24.981",             "value": "35"           },           {             "at": "2015-12-01 17:10:53.406",             "value": "38"           },           {             "at": "2015-12-01 17:11:24.815",             "value": "48"           },           {             "at": "2015-12-01 17:11:58.126",             "value": "58"           },           {             "at": "2015-12-01 17:12:22.925",             "value": "68"           },           {             "at": "2015-12-01 17:12:32.173",             "value": "78"           },           {             "at": "2015-12-01 17:13:36.797",             "value": "38"           },           {             "at": "2015-12-01 17:13:58.436",             "value": "48"           },           {             "at": "2015-12-01 17:14:03.500",             "value": "18"           },           {             "at": "2015-12-01 17:14:11.332",             "value": "78"           },           {             "at": "2015-12-01 17:14:15.983",             "value": "18"           },           {             "at": "2015-12-01 17:15:51.763",             "value": "18"           },           {             "at": "2015-12-01 17:15:56.768",             "value": "19"           },           {             "at": "2015-12-01 17:16:01.089",             "value": "13"           },           {             "at": "2015-12-01 17:16:04.720",             "value": "23"           },           {             "at": "2015-12-01 17:16:26.593",             "value": "13"           },           {             "at": "2015-12-01 17:16:30.998",             "value": "33"           },           {             "at": "2015-12-01 17:16:35.592",             "value": "43"           },           {             "at": "2015-12-01 17:16:40.401",             "value": "23"           },           {             "at": "2015-12-01 17:17:41.498",             "value": "aa"           },           {             "at": "2015-12-01 17:18:08.904",             "value": "10"           }         ]       }     ]   },   "id": "3200_0_5501" }, {   "error": "succ" }</pre>

#### 4.3.14.2 查看数据点--示例二

表 4-28

请求 URL	<a href="http://api.heclouds.com/devices/564280/datapoints?datastream_id=3200_0_5501&amp;start=2015-11-30T17:12:33&amp;end=2015-12-01T17:22:33&amp;limit=5">http://api.heclouds.com/devices/564280/datapoints?datastream_id=3200_0_5501&amp;start=2015-11-30T17:12:33&amp;end=2015-12-01T17:22:33&amp;limit=5</a>
响应结果	<pre>{   "errno": 0,   "data": {     "cursor": "25971_564280_1448961024981",     "count": 5,     "datastreams": [       {         "datapoints": [           {             "at": "2015-12-01 17:06:23.535",             "value": "39"           },           {             "at": "2015-12-01 17:06:34.488",             "value": "39"           },           {             "at": "2015-12-01 17:06:35.935",             "value": "39"           }         ]       }     ]   } }</pre>

	<pre>{   "at": "2015-12-01 17:09:35.918",   "value": "20" }, {   "at": "2015-12-01 17:09:55.915",   "value": "30" }], {   "id": "3200_0_5501" } } }, {   "error": "succ" }</pre>
--	--

#### 4.3.14.3 查看数据点--示例三

表 4-29

请求 URL	<pre>http://api.heclouds.com/devices/564280/datapoints? datastream_id=3200_0_5501&amp;start=2015-11-30T17:12:33&amp;end=2015-12- 01T17:22:33&amp;limit=5&amp;cursor=25971_564280_1448961024981</pre>
响应结果	<pre>{   "errno": 0,   "data": {     "cursor": "25971_564280_1448961152173",     "count": 5,     "datastreams": [       {         "datapoints": [           {             "at": "2015-12-01 17:10:24.981",             "value": "35"           },           {             "at": "2015-12-01 17:10:53.406",             "value": "38"           },           {             "at": "2015-12-01 17:11:24.815",             "value": "48"           },           {             "at": "2015-12-01 17:11:58.126",             "value": "58"           },           {             "at": "2015-12-01 17:12:22.925",             "value": "68"           }         ]       }     ]   },   "id": "3200_0_5501" } }, {   "error": "succ" }</pre>

## 第五章 接入实例

本章以 OneNET 平台开发的搭载有 M5310 模组的 NB-IoT 开发板为例，详细讲解终端设备接入 OneNET 平台的流程。NB-IoT 开发板接入 OneNET 平台及数据交互需要 OneNET 平台、M5310 模组以及 MCU 相互协作完成，OneNET 平台、M5310 模组以及 MCU 工作流程如图 5-1 所示。

### (1) OneNET 平台：

- 创建产品和设备；
- 响应登陆请求；
- 订阅和发现资源（Observe/Discover）；
- 下发指令（Read/Write/Execute）；

### (2) M5310 模组：

- 初始化基础通信套件及驻网；
- 创建对象及资源；
- 登陆请求；
- 上传对象资源；
- 响应平台指令；

### (3) MCU 侧：

- 创建设备（dev）；
- 向设备添加对象资源（object/instance/resource）；
- 发起登陆请求（AT+MIPLOPEN）；
- 响应模组指令；

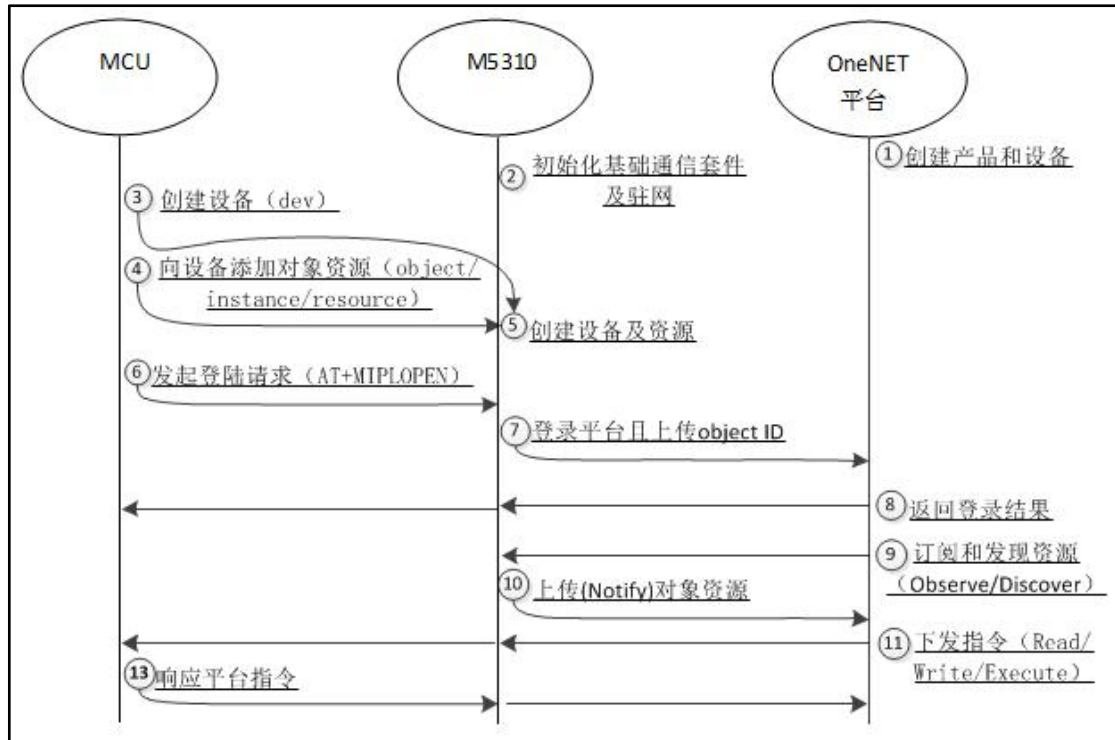


图5-1 M5310开发板接入OneNET平台及数据交互流程

## 5.1 MCU 侧工作流程说明

对用户而言，用 M5310 开发板接入 OneNET 平台及数据交互过程只需要关注 MCU 侧和平台侧的工作流程，M5310 模组侧的工作流程是模组自动完成，无需用户操作。MCU 侧的工作流程具体包括创建设备（dev）、向设备添加资源（Object/Instance/Resource）、登录请求、响应模组（即平台）指令。

### 5.1.1 创建设备（dev）

MCU 侧创建设备(dev)包括向设备添加 endpoint\_name(包括 IMEI 和 IMSI)、URI（重庆接入机地址："coap://183.230.40.40:5683"）、lifetime（设备在平台生存周期，最长 24 小时）、回调函数（包括读写执行函数）。创建设备函数如下：

```

nbiot_device_create( &dev,
                    endpoint_name,
                    uri,
                    life_time,
                    write_callback,
                    read_callback,

```



```
execute_callback );
```

在该函数中使用 AT+MIPLCONF 指令在模组中创建设备，其使用格式为：

AT+MIPLCONF=<size>,<config>,<index>,<flag>

其中，

<size>指示<config>部分总数据长度，按照 ASCII 计数；

<config>具体的设备配置数据，满足配置结构体规范；

<index>配置数据分片参数；

<flag>配置数据流结束符。

### 5.1.2 向设备添加资源

向设备添加资源为如下函数：

```
nbiot_resource_add( dev,  
                    Object ID,  
                    Instance Index,  
                    Resource ID,  
                    数据类型结构体 );
```

其中，dev 是创建设备函数中所创建的设备；Object ID 是参考 IPSO 定义的对象 ID；Instance Index 是对象实例的编号（从 0 开始编号）；Resource ID 是参考 IPSO 定义的资源 ID；数据类型结构体是所定义资源的所有可能的数据类型。

在该函数中使用两个 AT 指令：

（1）使用 AT+MIPLADDOBJ 指令在模组中添加对象（object），其使用格式为：AT+MIPLADDOBJ=<ref>,<objectid>,<instanceid>;

其中，

<ref>表示 OneNET 通信实例的引用 ID；

<objectid>为具体 Object 的 ID；

<instanceid>为实例的序号，从 0 开始编号。

例如，创建一个 Object 为 3200，instance 为 0 的对象，AT 指令如下：

AT+MIPLADDOBJ=0,3200,0;

（2）使用 AT+MIPLNOTIFY 指令将资源添加到对象下面，其使用格式为：

AT+MIPLNOTIFY=<ref>,<objectid>,<instanceid>,<resourceid>,<valuetype>,<value>,<flag>,[<ackid>];

其中，

<ref>: OneNET 通信实例的引用 ID;

<objectid>: 具体 Object 的 ID;

<instanceid>: 实例的序号, 从 0 开始编号;

<resourceid>: 资源 ID;

<valuetype>: OneNET 支持的数据类型包括, string, opaque, integer, float, bool 和 hex\_str;

<value>: 具体数值, 其大小不超过 1024Bytes;

<flag>: '1'表示所有已订阅的对象实例资源已添加完成, 模组将会更新到服务器;  
'0'表示还有对象实例资源没添加;

<ackid>: 选填, 如果设置大于 0, 则 OneNET 会返回 ACK。

### 5.1.3 登录请求

终端通过如下函数向 OneNET 平台发起登录请求:

```
nbiot_device_connect(dev,timeout);
```

其中 dev 是已经创建好的设备, 该设备包含对象/实例/资源以及与设备相关的各种配置参数。Timeout 是终端设备登录平台, 等待平台响应的超时时间, 超过这个时间, 终端设备则认为登录平台失败。

函数中使用 AT+MIPLOPEN 实现登录平台请求, 其使用格式为:

```
AT+MIPLOPEN=<ref>[,<timeout>];
```

其中, <ref>: OneNET 通信实例的引用 ID;

<timeout>: 可选, 连接平台超时时间。

终端设备登录成功会收到平台返回:+MIPLOPEN:0,1, 登录失败会收到平台返回:+MIPLOPEN:0,0。

终端设备登录成功后, 平台会下发 Observer 消息和 Discover 消息。模组收到这两条消息之后, 会自动处理, 无需用户另行处理。其中, Observer 消息是平台传递的观测请求消息, 模组收到该消息后会自动维护相关观测记录。Discover 消息是上报通知模组需要获取指定 Object 的属性, 上报后模组自动反馈相关属性。

登陆成功后, 可以在 OneNET 平台对应设备的资源列表中看到订阅的 Object

实体，如图 5-2。



图5-2 平台显示的设备状态

进入资源列表，可以看到登录时订阅的 Object/Resource 信息，如图 5-3。

序号	对象名称	实例个数	属性个数
1	Digital Input	1	1

实例名称	属性名	属性类型	属性值	时间	操作
Digital Input_0	Digital Input Count	opaque	null	null	<a href="#">读</a> <a href="#">写</a> <a href="#">执行</a> <a href="#">详情</a>

图5-3 资源列表

## 5.2 OneNET 平台侧数据收发流程说明

### 5.2.1 数据接收

用户上报数据到 OneNET 平台使用 AT+MIPLNOTIFY 指令，与 4.1.2 节讲解的用法一样，其格式为：AT+MIPLNOTIFY=<ref>,<objectid>,<instanceid>,<resourceid>,<valuetype>,<value>,<flag>,[<ackid>]。终端上报数据流程如图 5-4 所示：

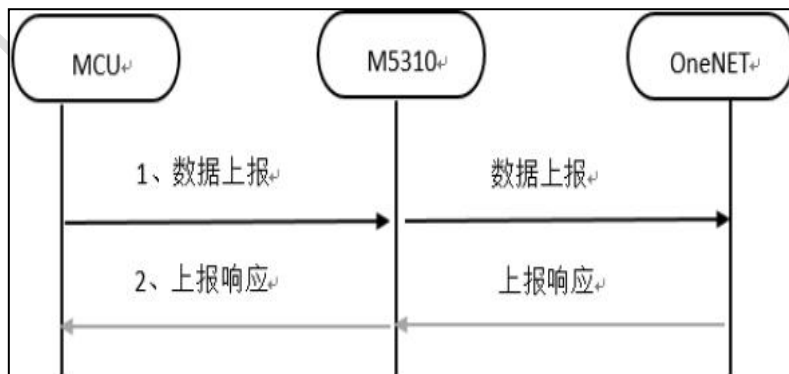


图5-4 终端上报数据流程

用户向 OneNET 上报的数据格式分两种。比如，向 object=3200 和 resource=5505 上传数据，分两种格式上传以及 OneNET 平台响应的数据格式如下：

(1) 不带 ackid 上报数据：

上报数据：AT+MIPLNOTIFY=0,3200,0,5505,1,"E365AAE447B",1

平台响应：OK

(2) 带 ackid 上报数据：

上报数据：AT+MIPLNOTIFY=0,3200,0,5505,1,"E365AAE447B",1,278

平台响应包括两部分：OK；

+MIPLNOTIFY:0,278; （其中，278 为 ackid）

在带 ackid 情况下，平台收到终端上传的数据后，会回复 ACK 消息，并且携带 ackid 值。

**另外，需要注意的是：**

(1) <ackid>被设置为大于 1 的情况下，平台侧收到数据会执行 ACK 回复操作；如果缺省，平台侧不会有 ACK 回复消息。<ackid>仅在<flag>位参数为 1 时有效。并且，在一次设备存活周期内，ackid 不能出现重复。

(2) 由于模组 M5310 的 buffer 资源限制，每次上报数据 Payload 用户数据部分不超过 1000Bytes。

### 5.2.2 指令下发

对终端设备的管理，除了可以通过 OneNET 平台直接读\写\执行外，还可以通过第三方应用平台调用 API 接口实现，将指令下发到 OneNET 平台，再通过 OneNET 平台将指令下发到终端设备。另外，API 接口除了可以对终端设备读\写\执行外，还具有获得设备信息、批量查询设备状态等功能。

需要说明的是，OneNET 平台或者第三方应用平台下发数据后，平台如果在 10 秒内未收到回复，会尝试多次下发，在超过门限次数（2 次）后，将会反馈失败。所以，MCU 应在收到模组转发的平台操作指示后数秒钟内（推荐 3s）上报对应操作结果，否则可能导致操作超时失败。

以下主要讲述第三方应用平台通过 OneNET 平台管理终端设备的流程，包括

读\写\执行三个指令的流程，如图 5-5 所示为管理指令在 OneNET 平台、M5310 模组和 MCU 之间的流程。

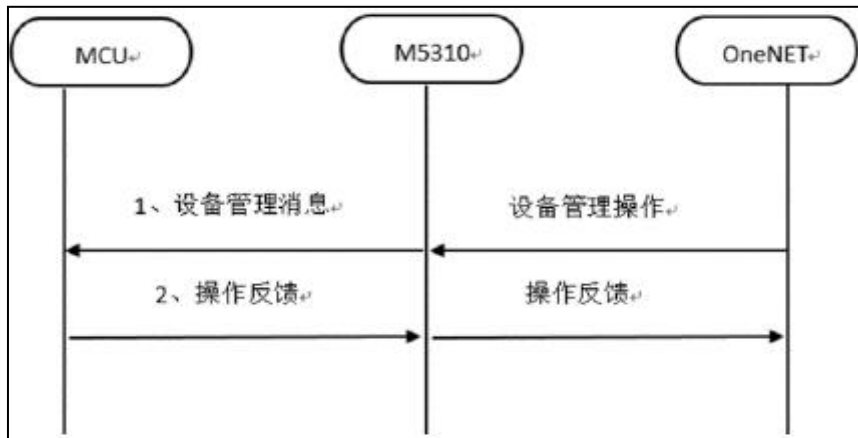


图5-5 指令流程

### 5.2.2.1 Read 操作流程

#### (1) 应用侧

第三方应用调用读设备 API 获得设备资源，使用 Fiddler 4 模拟第三方应用调用读操作 API（具体 API 格式请参见 3.2.4 节），需填写的设备相关信息有：imei=8658200300139788、obj\_id=3200、obj\_inst\_id=0、api-key= BBHR1rTSI=8oa6JEVz=IXP=uIT0=等参数，如图 5-6。填写完毕，点击执行，返回该设备下 obj\_id=3200, obj\_inst\_id=0 下面的 resource ID 以及其 value，如图 5-7 所示。



图5-6 Fiddler调用读设备API



图5-7 终端返回数据

## (2) 终端侧

调用 API 成功后，在终端会收到平台下发的以“+MIPLREAD”开头的读指令，如图 5-8 所示。

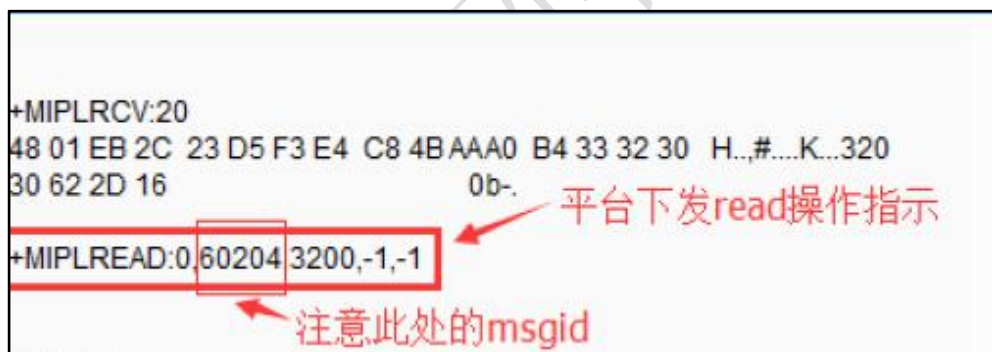


图5-8 平台下发读指令

终端调用 AT+MIPLREAD 指令响应平台的读指令，如图 5-9 所示。



图5-9 终端响应平台读指令

通过 API 参数的不同设置，还能实现读取指定 Object 下指定 instance 下的指定 resource 的数据；读取指定 Object 下所有 instance 的所有 resource 的数据等作。

**注意：**每次 Read 操作后，终端执行回复响应上报用户数据不超过 1000Bytes。



### 5.2.2.2 Write 操作流程

#### (1) 应用侧

第三方应用向终端设备写数据，使用 Fiddler 4 调用写操作 API（具体 API 格式请参见 3.2.5 节），需填写的设备相关信息有：imei=8658200300139788、obj\_id=3200、obj\_inst\_id=0、mode=2、api-key= BBHR1rTSI=8oa6JEVz=IXP=uIT0= 等参数，如图 5-10 所示。填写完毕，点击执行，收到该设备的响应，如图 5-11 所示。



图5-10 Fiddler调用写设备API

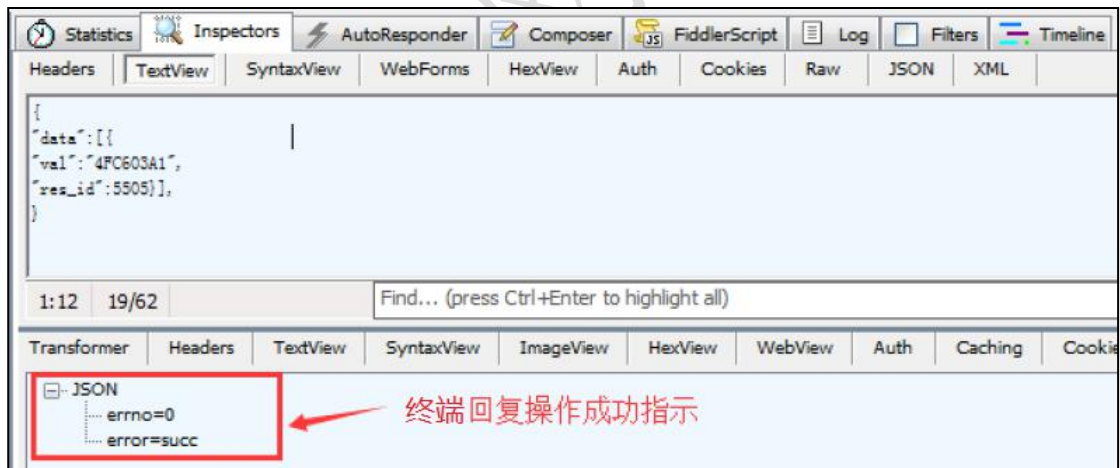


图5-11 终端返回写成功消息

#### (2) 终端侧

调用 API 成功后，在终端会收到平台下发的以“+MIPLWRITE”开头的读指令，如图 5-12 所示。



图5-12 平台下发写指令

终端调用 AT+MIPLWRITE 指令响应平台的写指令，如图 5-13 所示。



图5-13 终端响应平台写指令

注意：每次从平台使用 Write 操作下发的 COAP 包 Payload 用户数据部分不超过 1000Bytes。

### 5.2.2.3 Execute 操作流程

#### (1) 应用侧

第三方应用调用执行 API 向设备对应的 resource 发送字符串指令，使用 Fiddler 4 模拟第三方应用调用执行操作 API（具体 API 格式请参见 3.2.6 节），需填写的设备相关信息有：imei=8658200300139788、obj\_id=3200、obj\_inst\_id=0、res\_id=5505、api-key= BBHR1rTSI=8oa6JEVz=IXP=uIT0=等参数，如图 5-14 所示。填写完毕，点击执行，收到该设备的响应，如图 5-14 所示。

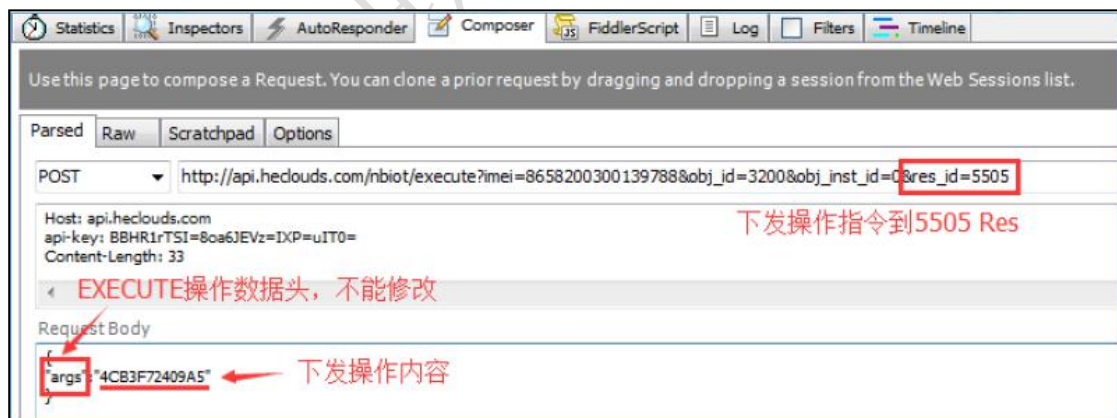


图5-14 Fiddler调用执行命令API





## 第六章 资源下载

### 6.1 数据推送 SDK 下载

OneNET 平台提供数据推送 SDK，目前包括 Nodejs、Java、PHP 和 C# 版本，下载地址：<https://open.iot.10086.cn/doc/art432.html#118>

### 6.2 NB-IoT API 下载

OneNET 平台提供 NB-IoT API SDK，目前只有 JAVA 版本，下载地址：<https://open.iot.10086.cn/doc/art432.html#118>

### 6.3 NB-IoT 开发板资料下载

OneNET 平台提供基于 M5310 的 NB-IoT 开发板，开发板相关资料下载地址：<https://open.iot.10086.cn/bbs/thread-19650-1-1.html>

## 第七章 NB-IoT 接入协议说明

NB-IoT（Narrow Band Internet of Things）属于 LPWAN 技术的一种，是一种为物联网而设计的基于蜂窝的窄带无线技术。NB-IoT 标准发展历程如图 7-1 所示。



图 7-1 NB-IoT 标准发展历程

NB-IoT 源起于物联网对广覆盖、大连接、低功耗和低成本的需求，所以 NB-IoT 应具备以下特点：

- （1）广覆盖：在同样的频段下，NB-IoT 比现有的网络增益提高 20dB，相当于提升了 100 倍覆盖区域的能力；
- （2）大连接：轻松支持大量设备联网需求，具备支撑海量连接的能力，NB-IoT 一个扇区能够支持 10 万个连接；
- （3）低功耗：聚焦小数据量、小速率应用特别对于一些不能经常更换电池的设备和场合，理论上 NB-IoT 终端模块的待机时间可长达 10 年；
- （4）低成本：预期的单个接连模块不超过 1 美元；

基于 NB-IoT 的以上特点，其典型的应用场景有远程抄表、烟感器、智慧井盖、智能路灯等等。

### 7.1 基于 NB-IoT 的 LWM2M 协议

OneNET 平台采用基于 NB-IOT 的 LWM2M 协议和 CoAP 协议实现 UE 与平台的通信，其中 LWM2M 协议为应用层协议，CoAP 协议为传输层协议。

LWM2M 协议是 OMA 组织制定的轻量化的 M2M 协议，主要面向基于蜂窝的窄带物联网（Narrow Band Internet of Things, NB-IoT）场景下物联网应用，聚焦于低功耗广覆盖（LPWA）物联网（IoT）市场，是一种可在全球范围内广泛应用的新兴技术。具有覆盖广、连接多、速率低、成本低、功耗低、架构优等特点。

### 7.1.1 LWM2M 协议逻辑实体与逻辑接口

(1) LwM2M 定义了三个逻辑实体：

- LWM2M Server：接入机，平台服务器接口；
- LWM2M client：客户端，负责执行服务器 的命令和上报执行结果；
- Bootstrap server：引导机，负责 配置 LWM2M 客户端。

(2) 这三个逻辑实体之间有 4 个逻辑接口：

- Bootstrap: Bootstrap Server 通过这个接口来配置 Client - 比如说 LWM2M server 的 URL 地址；
- Device Discovery and Registration: 客户端注册到服务器并通知服务器客户端所支持的能力；
- Device Management and Service Enablement: LWM2M Server 发送指令给 Client 并受到回应.；
- Information Reporting: LWM2M Client 来上报其资源信息的，比如传感器温度。

这三个逻辑实体与四个逻辑接口之间的关系如图 7-2 所示。

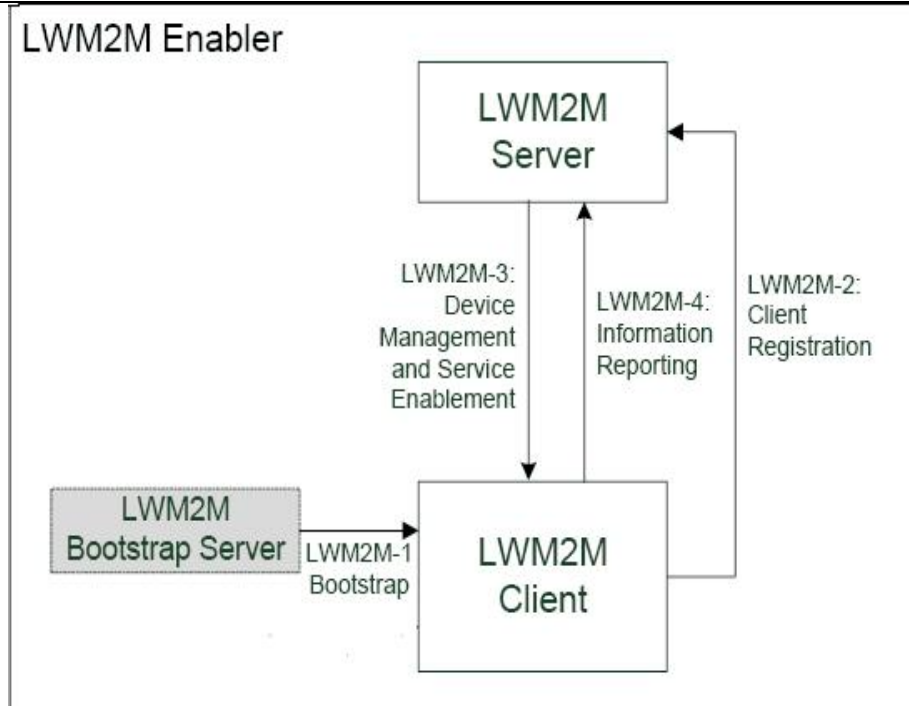


图7-2 逻辑实体与逻辑接口之间的关系

## 7.1.2 LWM2M 协议栈

Lightweight M2M 协议栈结构如图 7-3 所示。

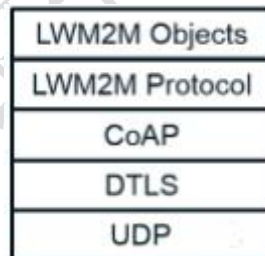


图7-3 Lightweight M2M 协议栈结构

### 7.1.2.1 LWM2M Objects

每个 object（对象）对应客户端的某个特定功能实体。LWM2M 规范定义了一些标准 Objects，比如：

urn:oma:lwm2m:oma:2; (LWM2M Server Object，其中‘2’为 object ID)。

urn:oma:lwm2m:oma:3; (LWM2M Access Control Object，其中‘3’为 object ID)。

除了 LWM2M 定义的 object，IPSO 组织也定义了一些常用传感器的 object，比如：

urn:oma:lwm2m:ext:3311; (IPSO Light Control, , 其中‘3311’为 object ID)。

每个 object 可以有多个 resource, 每个 resource 代表一项 object 属性或者功能。比如 object ID 为 3311 的传感器的部分 resource 描述如表 7-1:

序号	Resource ID	描述	类型	权限
1	5851	Dimmer	Integer 0-100	R/W
2	5850	On/Off	Boolean	R/W

表 7-1 resource 描述

### 7.1.2.2 LWM2M Protocol

LWM2M Protocol 定义了一些逻辑操作, 比如 Read, Write, Execute, Create 和 Delete 等操作。通过这些逻辑操作, 实现云平台与终端设备的数据交互。

## 7.2 基于 NB-IoT 的 CoAP 协议

CoAP(Constrained Application Protocol)协议是 IETF 提出的一种面向网络的协议, 采用了与 HTTP 类似的特征, 核心内容为资源抽象、REST 式交互以及可扩展的头选项等。为了克服 HTTP 对于受限环境的劣势, CoAP 既考虑到数据报长度的最优化, 又考虑到提供可靠通信。

CoAP 协议具有以下特点:

- (1) 消息模型, 以消息为数据通信载体, 通过交换网络消息来实现设备间数据通信;
- (2) 对云端设备资源操作都是通过请求与响应机制来完成, 类似 HTTP, 设备端可通过 4 个请求方法 (GET, PUT, POST, DELETE) 对服务器端资源进行操作;
- (3) 协议包轻量级, 最小长度仅为 4B;
- (4) 支持可靠传输, 通过确认和数据重传确保数据可靠到达;
- (5) 支持 IP 多播, 即可以同时向多个设备发送请求;
- (6) 非长连接通信, 适用于低功耗物联网场景。

### 7.2.1 CoAP 协议栈

CoAP 协议栈如图 7-4 所示, CoAP 由 UDP 作为承载, 遵循 UDP 基本的协议报文格式, UDP 数据内容部分按照 CoAP 协议报文格式进行写入传输。





- (1) **Success 2.xx** 代表客户端请求被成功接收并被成功处理；
- (2) **Client Error 4.xx** 代表客户端请求有错误，比如参数错误等；
- (3) **Server Error 5.xx** 代表服务器在执行客户端请求时出错。

The diagram illustrates the structure of a CoAP message, showing how it is encapsulated within UDP and IP headers. The layers are as follows:

- CoAP Header (4 bytes):**
  - Ver (2 bits)
  - T (2 bits)
  - TKL (4 bits)
  - Code (8 bits)
  - Message ID (16 bits)
  - Token (if any, TKL bytes)
  - Options (if any...)
  - Payload (if any...)
- UDP Header (8 bytes):**
  - Source Port (16 bits)
  - Destination Port (16 bits)
  - Length (16 bits)
  - Checksum (16 bits)
- IP Header (20 bytes):**
  - Source IP Address (32 bits)
  - Destination IP Address (32 bits)
  - Protocol (8 bits)
  - TTL (8 bits)
  - Length (16 bits)

The diagram shows the CoAP message being encapsulated within a UDP datagram, which is then encapsulated within an IP packet. The total size of the message is 24 bytes (4 bytes for CoAP header, 8 bytes for UDP header, and 12 bytes for the CoAP payload).

图7-5 CoAP消息报文示意图

### (1) CoAP 消息报文头部 (Header)

头部固定为 4 个 Bytes，包含版本号 (Ver)、报文类型 (T)、CoAP 标识符长度 (TKL)、功能码/响应码 (Code) 以及报文编号 (Message ID)。各部分说明如下：

Ver: 版本号占 2 位，取值为 01B，用于指示 CoAP 协议的版本号。

T: CoAP 协议定了 4 种不同形式的报文，CON 报文，NON 报文，ACK 报文和 RST 报文。

TKL: CoAP 协议中具有两种功能相似的标识符，一种为 Message ID (消息编号)，一种为 Token (标识符)。其中每个报文均包含消息编号，但是标识符对于报文来说是非必须的。

Code: Code 在 CoAP 请求报文和响应报文中具有不同的表现形式，Code 占一个字节，它被分成了两部分，前 3 位一部分，后 5 位一部分，为了方便描述它被写成了 c.dd 结构。其中 0.XX 表示 CoAP 请求的某种方法，而 2.XX、4.XX 或 5.XX 则表示 CoAP 响应的某种具体表现。

Message ID: 消息编号。

### (2) Token (可选)

标识符具体内容，通过 TKL 指定 Token 长度。通过 token，客户端收到响应后，取出 Token，就可以知道该响应是针对之前哪个请求回复的。

### (3) Option (可选，0 个或者多个)

请求消息 与回应消息都可以 0~多个 options。 主要用于描述请求或者响应对应的各个属性，类似参数或者特征描述。

### (4) 1111 1111

CoAP 报文和具体负载之间的分隔符。

### (5) Payload

实际携带数据内容， 若有携带数据， 前面加 payload 标志 0xFF。

## 7.2.2 块传输

CoAP 协议的特点是传输的内容小巧精简，但是在某些情况下不得不传输较大的数据。在这种情况下可以使用 CoAP 协议中的选项设定分块传输的大小，服



务器和客户端即可完成分片和组装。扩展的块传输协议在 COAP 基础协议上增加了 3 个 options，2 个 response codes 用于块传输大小协商及控制。当请求消息或者响应消息里面有出息 block1/block2 option 时，代表这是块传输通信。需要根据 option block 里面 M 标识，决定是否继续进行块传输。

#### 7.2.2.1 新增 option 说明

新增加的 3 个 options 如表 7-2 所示：

表 7-2 新增 option 说明

Number	Name	Reference
23	Block2	<a href="#">RFC 7959</a>
27	Block1	<a href="#">RFC 7959</a>
28	Size2	<a href="#">RFC 7959</a>

Option Block2：主要用于服务器端响应时，分块传输，比如设备端发起资源发现时，由于服务器上资源较多，就要启动分块传输。

Option Block1：主要用于客户端发出请求时，分块传输，比如需要上传一个大的数据包到服务器上。

Option Size2：代表服务器端响应资源总的大小。

Option Block 由三部分组成：

(1) NUM：占用 0~3Byte，代表当前块编号，以 0 开始，如我们要传 10 个数据块，则数据块的编号为 0~9。

(2) M：占用 1bit，如果设置为 1 代表还有剩下数据块未传输。如果设置为 0，代表数据块都已传输出去。

(3) SZX：占用 2bit，取值范围 0~6，对应每个 block 大小为  $2^{(SZX+4)}$ ，即范围为 (16~1024) Bytes。

#### 7.2.2.2 新增 response code 说明

新增加的 2 个 response codes 如表 7-3 所示：

表 7-3 新增 response code 说明

Code	Description	Reference
2.31	Continue	<a href="#">RFC 7959</a>
4.08	Request Entity Incomplete	<a href="#">RFC 7959</a>

### 7.2.3 安全传输

COAP 使用 DTLS 来做安全传输层，该层运行于 UDP 之上，如图 7-6 所示。

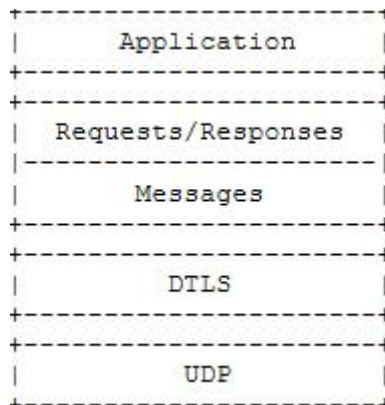


图7-6 DTLS分层图

当前考虑使用 DTLS 时，需要考虑设备终端资源受限情况，有些资源有限设备无法运行 DTLS 安全加密算法。做安全加密，需要根据应用场景需要，对应只上报数据，且数据敏感度不高场景，可以不考虑加入安全层。

## 第八章 常见问题 FAQ

用户在基于 OneNET 平台进行 NB 产品开发的过程中,可能会遇到各种各样的问题。开发过程中遇到的问题,一方面可以参考 OneNET 文档中心的 NB-IoT 文档或者本文档前几个章节的内容,另一方面用户可以参考本章节总结的常见问题。希望这些文档能对用户在开发过程中有一定的指导和帮助。

本章将分三个小节对用户开发 NB 产品过程中可能遇到的问题进行梳理,分别为:终端设备接入 OneNET 平台的常见问题、API 调用的常见问题和 OneNET 平台的能力。

### 8.1 终端设备接入 OneNET 平台的常见问题

序号	问题描述	原因分析	解决方法
1	怎样在 OneNET 平台创建 NB-IoT 产品		在产品创建页面“联网方式”处选择 NB-IoT,不能选择 wifi 和移动蜂窝网络。
2	怎样创建设备资源		按照 IPSO 或者 LWM2M 定义的资源模型创建资源,资源模型为: ObjectID_InstanceID_ResourceID
3	SDK 初始化模组失败	SDK 在启动的时候会首先对当前模组的工作模式进行配置,使其能够附着进 NB 网络中,配置的方式是通过一系列 AT 指令,需要注意的是模组处理每条 AT 指令都需要一定的时间。	建议 AT 指令间可以延迟几秒,否则会造成初始化过程失败。
4	终端登陆不上 OneNET	(1) 终端是否已经集成了 OneNET SDK 通信套件; (2) 终端附着不进 NB 网络; (3) NB 核心网和公网不通;	(1) 如果终端没有集成 OneNET SDK 通信套件,请先完成移植 SDK; (2) 终端查看当前 NB 网络的信号强度和信噪比,如果很差,需要运营商优化网络;其次,保证物联卡处于激活状态; (3) 运营商需要开放 NB 核心网,

		(4) 鉴权失败, 返回 4.03。	保证核心网和 OneNET 平台之间的连通性 (此情况只针对电信和联通); (4) 终端侧需要检查用于鉴权的 IMEI 和 IMSI 是否与 OneNET 平台上注册设备的 IMEI 和 IMSI 一致, 平台通过设备的 endpoint name 鉴权, endpoint name 的格式是 imei;imsi 中间为分号
5	OneNET SDK 基础通信套件是否支持数据分片传输		支持。数据分片是集成了 OneNET SDK 基础通信套件模组或者 NB 芯片自动完成的, 无需用户操作。
6	LWM2M 协议与 IPSO 定义的 object 区别		各自定义的 object, OneNET 平台都支持, 用户一般用 IPSO 定义的 object, 模组用 LWM2M 定义模型。
7	平台是否保存连续上传的同一条数据	CoAP 协议通过 message id 和 token 来过滤重复消息。	对于上报的数据包, 因为每次 token 相同, 如果 message id 也不变, 会被当作重复消息被过滤, 重复消息过滤的时间窗口是 247 秒。

## 8.2 API 调用的常见问题

序号	问题描述	原因分析	解决方法
1	读/写/执行返回 TIME_OUT	TIME_OUT 是因为平台没有在超时时间内 (默认 25s, 可设置范围 5~40s) 收到设备的响应。	(1) 网络连接 session 被核心网回收, NB-IOT 设备通过核心网连接到平台, 如果设备在一段时间内 (各地情况不一, 通常为几分钟) 没有上行和下行的活动, 核心网会回收连接, 此时平台下发的消息无法到达设备; (2) 网络问题, NB-IOT 的网络不稳定, 尽管 CoAP 有重传机制, 仍然有可能在 25 秒内无法完成平台到设备的请求响应的全过程。
2	执行接口 (Execute) 无法下发二进制数据		Lwm2m 协议中, Execute 操作的参数为字符串, 不支持二进制, 要下发二进制, 可以使用 Write 操作, 请参照 API 文档写资源接口。
3	API 调用失败	(1) 在调用 API 之前,	(1) 请参考 4.1 节 “第三方应用

		<p>需保证 OneNET 平台与应用平台已建立连接；</p> <p>(2) 检查 SDK 是否是 NB-IoT 协议的 API；</p> <p>(3) 检查 API 指令的调用方式（如 POST、GET 等）、语法以及配置参数。</p>	<p>接入 OneNET 平台”；</p> <p>(2) NB-IoT API 下载地址： <a href="https://open.iot.10086.cn/doc/art432.html#118">https://open.iot.10086.cn/doc/art432.html#118</a></p> <p>(3) 请参考 API 格式正确调用。</p>
4	第三方应用怎么接入 OneNET 平台？		<p>用户在 OneNET 产品页面“第三方开发平台”配置应用服务器地址（URL）、Token 等参数后，OneNET 平台向应用服务器发送 URL&amp;Token 接入校验，应用回复 msg（验证消息）给平台，完成接入。（注意：只有一次接入需要校验，以后每次推送数据都不需要校验）</p> <p>附：第三方应用平台接入 OneNET 平台的 SDK 下载地址： <a href="https://open.iot.10086.cn/doc/art432.html#118">https://open.iot.10086.cn/doc/art432.html#118</a>-----“数据推送部分”</p>
5	数据推送到第三方应用，应用回复什么？		每次推送数据成功后都是回复 200（表示应用收到数据）。
6	平台缓存离线命令条数		每个设备下最多缓存 10 条，每增加一条就会按照时间先后依次覆盖最早的历史命令。
7	平台缓存离线命令时间		可以缓存一年。
8	平台保存数据时间		可以缓存一年。
9	平台停止推送数据原因	平台累计推送失败 2000 次，停止推送，并会以短信方式通知用户。	可以在平台端手动启用重新推送（并且只能在平台手动启用），并重新计数。

### 8.3 使用 OneNET 平台遇到的常见问题

序号	问题描述	原因分析	解决方法
1	高并发能力		支撑高并发应用及终端接入，保

			证可靠服务; 提供高达 99.9% 的 SLA 服务可用性。
2	数据安全存储及传输		<p>(1) OneNET 平台通过可信云安全认证, 平台具有分布式结构和多重数据保障机制, 提供安全的数据存储;</p> <p>(2) 提供传输加密, 支持基于公钥的 DTLS 加密 (使用 coap 和 5684 端口连接), 后续会支持基于 PSK 和 X.509 证书的 DTLS 加密。保证用户数据 360 度全方位安全。</p>
3	多协议接入		支持多种行业及主流标准协议的设备接入, 如 LWM2M (NB-IOT)、MQTT、Modbus、EDP、HTTP、JT/T808 以及 TCP 透传等。
4	SDK 多语言版本		<p>(1) 平台提供 NB-IoT API JAVA SDK;</p> <p>(2) 数据推送 SDK 有四个版本: Nodejs、Java、PHP 和 C#</p>
5	平台是否支持自定义资源模型		<p>(1) 平台不显示资源列表, 但是会显示二进制数据流;</p> <p>(2) 非 IPSO 和 LWM2M 模型的 Object ID 和 Resource ID, 平台无法判断资源的数据类型, 只能按照二进制处理, 用户在应用层需要自己恢复出原始数据。</p>
6	平台是否支持 DTLS 加密传输		<p>支持。</p> <p>(1) 当前支持基于公钥的 DTLS 加密 (5684 端口连接), 后续会支持基于 PSK 和 X.509 证书的加密;</p> <p>(2) 当前也支持不使用加密方式的南向接入 (5683 端口连接)。</p>
7	Integer/Float/string 数据显示为二进制格式	没有使用 IPSO 定义的标准资源模型	对于非 IPSO 模型的 Object ID 和 Resource ID, 平台无法判断资源的数据类型, 只能按照二进制处理。
8	平台下发失败, 返回 timeout	<p>(1) 模组休眠;</p> <p>(2) 核心网老化时间</p>	(1) 终端进入 PSM 状态后, 下行数据不可达, 只有主动上传数据

		限制； (3) 网络延迟或者网络丢包。	才能唤醒自身； (2) 老化时间过后，终端在核心网侧映射的端口会被释放掉，核心网没法完成下行转发； (3) 由于网络时延大或者丢包，在超时时间（默认 25s，可通过 API 设置超时时间：5s~40s）内无法完成应用平台的指令下发以及终端设备的响应。
--	--	------------------------	---

## 8.4 OneNET 平台 FOTA 功能

请参考《FOTA 升级使用说明书》。

下载地址：<https://open.iot.10086.cn/doc/art431.html#118> FOTA 部分。