**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 |
| V0.1 | 2018.5 |  | 肖勇、张小波、卓定飞、敬威 | 初版 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[第一章 文档说明 5](#_Toc516040731)

[第二章 基于OneNET平台的NB设备接入及应用开发流程总体综述 6](#_Toc516040732)

[2.1 NB设备接入OneNET平台流程图 6](#_Toc516040733)

[2.2 应用开发流程图 6](#_Toc516040734)

[第三章 NB设备接入OneNET平台 8](#_Toc516040735)

[3.1 终端设备接入OneNET平台前的准备工作 8](#_Toc516040736)

[3.1.1 SDK移植到MCU 9](#_Toc516040737)

[3.1.2 SDK移植到NB通信模组 9](#_Toc516040738)

[3.1.3 SDK移植到NB芯片 10](#_Toc516040739)

[3.2 终端设备接入OneNET平台步骤 11](#_Toc516040740)

[3.2.1 OneNET平台创建产品及设备 12](#_Toc516040741)

[3.2.2 终端设备软硬件初始化 15](#_Toc516040742)

[3.2.3 终端创建设备及资源 15](#_Toc516040743)

[3.2.4 登录OneNET平台 16](#_Toc516040744)

[3.2.5 平台订阅&发现设备资源 17](#_Toc516040745)

[第四章 第三方应用开发 18](#_Toc516040746)

[4.1第三方应用接入OneNET平台 18](#_Toc516040747)

[4.1.1第三方应用平台接入验证程序 19](#_Toc516040748)

[4.1.2 OneNET平台配置第三方应用平台 19](#_Toc516040749)

[4.2 OneNET平台数据推送 20](#_Toc516040750)

[4.1.1 明文消息 21](#_Toc516040751)

[4.1.2 密文消息 22](#_Toc516040752)

[4.1.3 消息相关字段说明 22](#_Toc516040753)

[4.1.4 加密算法详述 23](#_Toc516040754)

[4.2 API接口 24](#_Toc516040755)

[4.2.1创建设备 25](#_Toc516040756)

[4.2.2查看单个设备信息 26](#_Toc516040757)

[4.2.3删除设备 27](#_Toc516040758)

[4.2.4读设备资源 27](#_Toc516040759)

[4.2.5写设备资源 28](#_Toc516040760)

[4.2.6下发命令 29](#_Toc516040761)

[4.2.7获取资源列表 29](#_Toc516040762)

[4.2.8订阅 30](#_Toc516040763)

[4.2.9离线命令 31](#_Toc516040764)

[4.2.10 触发器 35](#_Toc516040765)

[4.2.11批量查询设备状态 40](#_Toc516040766)

[4.2.12批量查询设备最新数据 40](#_Toc516040767)

[4.2.13查看数据点 41](#_Toc516040768)

[第五章 接入实例 46](#_Toc516040769)

[5.1 MCU侧工作流程说明 47](#_Toc516040770)

[5.1.1创建设备（dev） 47](#_Toc516040771)

[5.1.2向设备添加资源 48](#_Toc516040772)

[5.1.3登录请求 49](#_Toc516040773)

[5.2 OneNET平台侧数据收发流程说明 50](#_Toc516040774)

[5.2.1数据接收 50](#_Toc516040775)

[5.2.2指令下发 51](#_Toc516040776)

[第六章 资源下载 57](#_Toc516040777)

[6.1 数据推送SDK下载 57](#_Toc516040778)

[6.2 NB-IoT API下载 57](#_Toc516040779)

[6.3 NB-IoT 开发板资料下载 57](#_Toc516040780)

[第七章 NB-IoT接入协议说明 58](#_Toc516040781)

[7.1 基于NB-IoT的LWM2M协议 58](#_Toc516040782)

[7.1.1 LWM2M协议逻辑实体与逻辑接口 59](#_Toc516040783)

[7.1.2 LWM2M协议栈 60](#_Toc516040784)

[7.2基于NB-IoT的CoAP协议 61](#_Toc516040785)

[7.2.1 CoAP协议栈 61](#_Toc516040786)

[7.2.2 块传输 63](#_Toc516040787)

[7.2.3 安全传输 65](#_Toc516040788)

[第八章 常见问题 66](#_Toc516040789)

# 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功底的嵌入式开发人员，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。

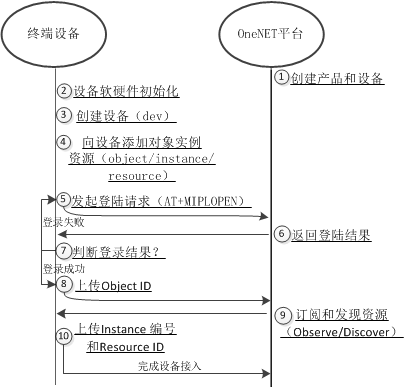
注意：建议以公司名义先注册OneNET平台账号

# OneNET平台NB设备开发流程总体综述

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。

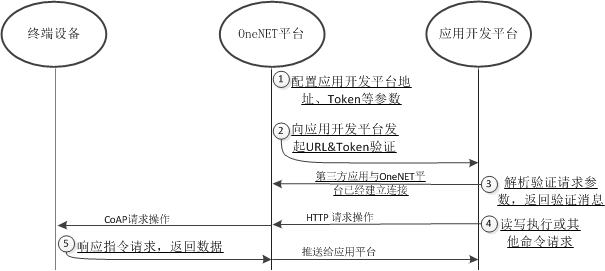
注意：流程图中的数字标号从小到大表示流程的步骤。

## NB设备接入OneNET平台流程图



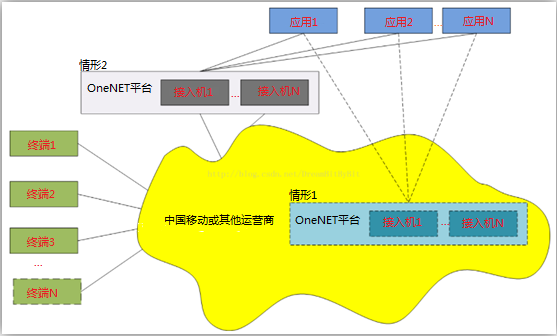
NB设备接入OneNET平台流程图

## 应用开发流程图

应用开发流程图

# 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。



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。

## 终端设备接入OneNET平台前的准备工作

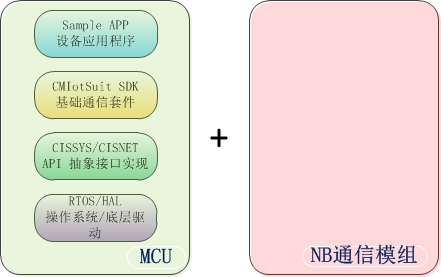
在设备接入OneNET平台之前，设备侧需完成由OneNET平台提供的基础通信套件SDK的移植工作。目前，按SDK集成在终端的方式可将终端分为如下三类：

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”的SOC架构中，SDK移植到NB芯片中。

### 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。

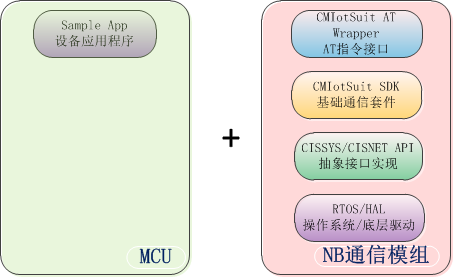


SDK移植到MCU中的软硬件架构图

### 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。



SDK移植到NB通信模组中的软硬件架构图

|  |  |  |
| --- | --- | --- |
| **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 | |

已完成移植接入OneNET平台的SDK模组

### 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3-2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。



SDK移植到NB芯片中的软硬件架构图

|  |  |  |
| --- | --- | --- |
| **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 |

已完成移植接入OneNET平台的SDK NB芯片

## 终端设备接入OneNET平台步骤



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。

### 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，点击首页右上角的“[**注册**](https://open.iot.10086.cn/reg)”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](https://open.iot.10086.cn/login)”，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。



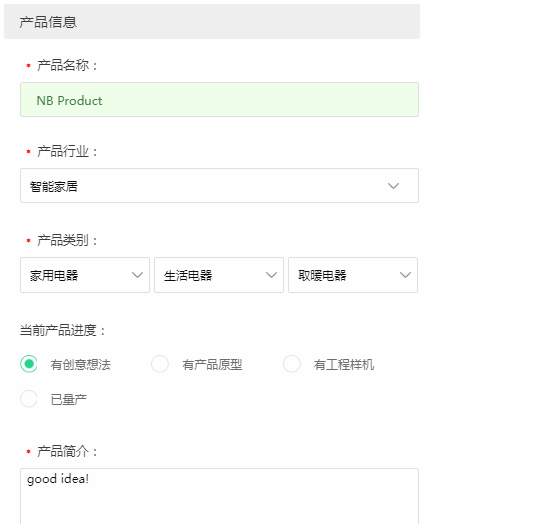
用户账号注册页面

#### 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“**N**B-IoT****”,设备接入协议选择**“**LWM2M**”**。

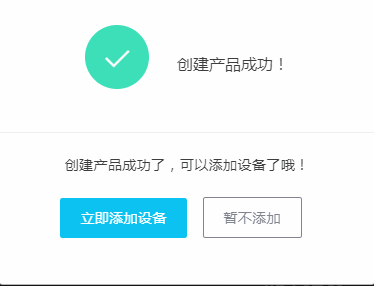
产品创建完成后，页面弹出“**创建产品成功**”，如图3-9所示。点击“**立即添加设备**”，显示如图3-10所示界面。依次填写**“设备名称”、IMEI和IMSI**，并根据需要在“**是否开启自动订阅**”选项选择“**是**”或者“**否**”。如果用户开启自动订阅功能，终端设备创建的资源信息会自动更新到平台上对应“虚拟设备”的资源列表下。用户需要将“虚拟设备”里面的IMEI和IMSI更新到终端设备里面，用于平台对终端设备鉴权。



创建产品页面



创建产品页面



创建产品成功页面



添加设备页面

### 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。

### 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。

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

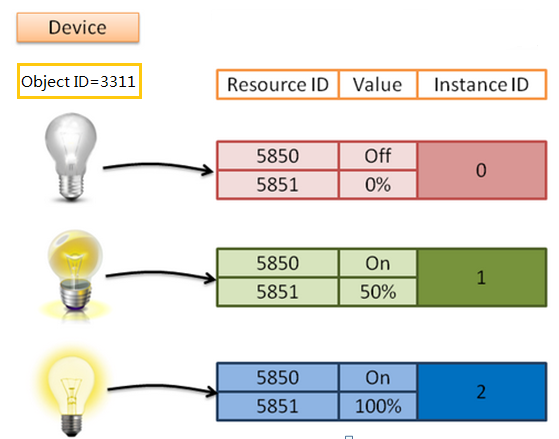
Resource ID=5851/5850描述

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。



Object/Instance/Resource三者的关系

### 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。



平台侧设备状态显示

### 平台订阅&发现设备资源

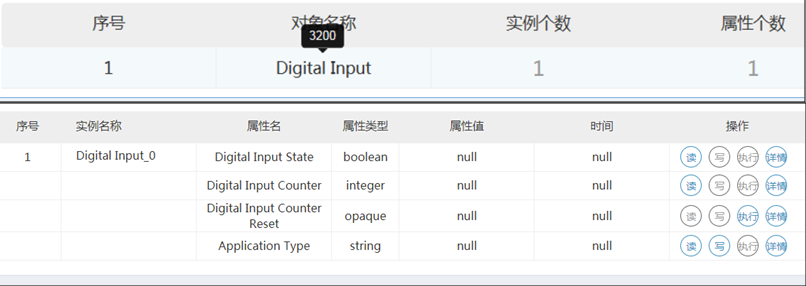
终端设备登录平台成功后，OneNET平台会向设备下发Observer消息和Discover消息。终端设备收到这两条消息后，基础通信套件会自动处理，无需用户另行处理。

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。



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。



平台侧显示的Object及其属性

# 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平台提供的数据推送功能以及OneNET平台为第三方应用封装好的API接口。



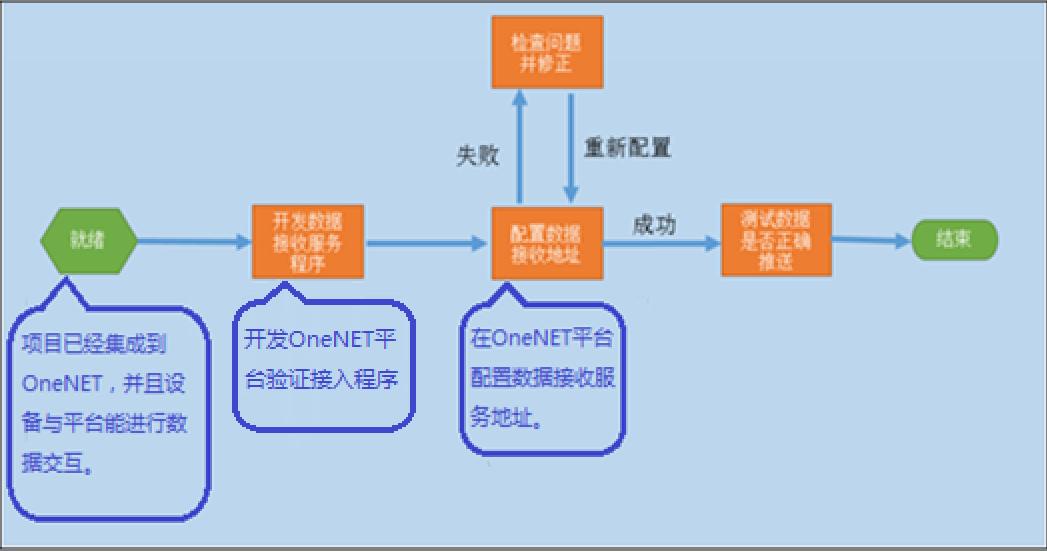
第三方应用与OneNET平台的逻辑关系

## 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。



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：见文档中心

### 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。**

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数据后，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。



第三方开发平台的配置界面

## 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。

### 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"

                     }

### 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"

                    }

### 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 |
| 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 |

消息相关字段说明

### 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\_Decode生成32个字节的AESKey。
* AES采用CBC模式，秘钥长度为32个字节（256位），数据采用PKCS#7填充 ,初始化iv向量取秘钥前16字节； PKCS#7：K为秘钥字节数（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
* （4）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)，解密中使用的秘钥由EncodingAESKey计算得来，使用的初始化iv向量为计算出的aes秘钥的前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。

## 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4-3。下面分别对每一种API进行说明。

|  |  |
| --- | --- |
| 操作对象 | API名称 |
| 设备 | [创建设备](#_创建设备) |
| 查看单个设备信息 |
| 删除设备 |
| 资源 | [读设备资源](#_读设备资源) |
| [写设备资源](#_写设备资源) |
| 命令 | [下发命令-执行](#_下发命令) |
| 列表 | [获取资源列表](#_获取资源列表) |
| 订阅 | [订阅查看某个设备](#_订阅) |
| 离线命令 | 读数据 |
| 写数据 |
| 命令 |
| 查看指定设备离线命令列表 |
| 查看设定离线命令详情 |
| 取消离线命令 |
| 触发器 | 新增 |
| 更新 |
| 查看 |
| 删除 |
| 批量查询 | 批量查询设备状态 |
| 批量查询设备最新数据 |
| 数据点 | [查看数据点](#_查看数据点) |

API接口列表

### 4.2.1创建设备

|  |  |
| --- | --- |
| HTTP方法 | POST |
| URL | http://api.heclouds.com /devices |
| HTTP头部 | api-key:xxxx-ffff-zzzzz，必须为MasterKey  Content-Type:application/json |
| HTTP内容 | {      "title": "mydevice",  //设备名      "desc": "some description", //设备描述（可选）      "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ture）  "auth\_info": {"xxxxxxxxxxxx":"xxxxxxxxxxxxxx"}, //NBIOT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）  } |
| HTTP响应响应消息内容 | {  "errno": 0, //无错误  "error":“succ”,  "data":  {  "device\_id":"233444" //平台分配唯一ID  }  } |
| 说明 | （1）响应消息中errno表示错误码，error表示错误原因，如果创建设备失败，则没有device\_id字段；  （2）NB CoAP设备auth\_info中imei（不超过17位）和imsi（不超过16位）均由数字或者字母组成。 |

### 4.2.2查看单个设备信息

|  |  |
| --- | --- |
| 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 | {  "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*  "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":177.034029968273},  "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  }//end data  } |

### 4.2.3删除设备

|  |  |
| --- | --- |
| 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 | {  "errno": 0,  "error":“succ”  } |
| 说明 | 响应消息中errno表示错误码，error表示错误原因。 |

### 4.2.4读设备资源

|  |  |
| --- | --- |
| 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 | "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，选填  "res\_id": 2122 // nbiot设备的资源id，选填 |
| 成功返回 | {  "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  },  ………  ]  },  ………  ]  } |
| 说明 | 1. obj\_instance\_id不存在的时候，resource\_id必不存在。   2、响应消息中errno表示错误码，error表示错误原因。 |

### 4.2.5写设备资源

|  |  |
| --- | --- |
| 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 | "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 |
| HTTP内容 | {  "data": [{  "res\_id":123,  "val":Object //可为boolean、string、long、double  },  {…},  …  ]  } |
| 成功返回 | {  "errno": 0,  "error": "succ",  } |
| 说明 | 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。 2. HTTP内容部分必须存在。 3. 响应消息中errno表示错误码，error表示错误原因。 |

### 4.2.6下发命令

|  |  |
| --- | --- |
| 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errno表示错误码，error表示错误原因。 |

### 4.2.7获取资源列表

|  |  |
| --- | --- |
| 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":[  {  "inst\_id":0,  "resources":[5500,5050]  },  {…..},  …..  ]  },  {…….},  ……  ]  }  } |
| 说明 | 1. HTTP内容部分选填。 2. 响应消息中errno表示错误码，error表示错误原因。 |

### 4.2.8订阅

|  |  |
| --- | --- |
| 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 | "imei":121, // nbiot设备的身份码，必填  "cancel":true|false, //true为取消订阅，false为订阅，必填  "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，选填  "res\_id": 123 // nbiot设备的资源id，选填  "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  "pmax":123,//上传数据的最大时间间隔，int类型，可选  "gt":12, //当数据大于该值上传，double类型，可选  "lt":233，// 当数据小于该值上传，double类型，可选  "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 |
| 成功返回 | {  "errno": 0,  "error": "succ",  } |
| 说明 | 1、pmin和pmax都存在是，pmax>=pmin且需大于0。  2、lt < gt，并且lt + 2\*st < gt。  3、如果有gt、lt、st，则res\_id必填。  4、cancel为非true|false将被默认为false。  5、响应消息中errno表示错误码，error表示错误原因。 |

### 4.2.9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，平台将清处离线命令缓存。

下面分别对各个离线命令进行说明。

#### 4.2.9.1读数据/Read

|  |  |
| --- | --- |
| HTTP方法 | GET |
| URL | http://api.heclouds.com /nbiot/offline |
| 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":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，选填  "res\_id": 2122 ,// nbiot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响应响应消息内容 | {  "errno": 0,  "error":"succ",  "data":  {  "uuid":"42742677-adc3-54ca-83a1-5aaaf71482f8" //离线命令uuid  }  } |
| 说明 | 1、响应消息中errno表示错误码，error表示错误原因。 |

#### 4.2.9.2写数据/Write

|  |  |
| --- | --- |
| HTTP方法 | POST |
| URL | http://api.heclouds.com /nbiot/offline |
| 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，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, // 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  “valid\_time”: "2018-03-08T17:30:00", //命令开始生效时间，必填大于当前时间  “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内容 | {  "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  },  {…},  …  ]  } |
| HTTP响应响应消息内容 | {  "errno": 0,  "error":"succ",  "data":  {  "uuid":"42742677-adc3-54ca-83a1-5aaaf71482f8" //离线命令uuid  }  } |
| 说明 | 1、响应消息中errno表示错误码，error表示错误原因。 |

#### 4.2.9.3执行/Execute

|  |  |
| --- | --- |
| HTTP方法 | POST |
| URL | http://api.heclouds.com/nbiot/execute/offline |
| HTTP头部 | api-key:xxxx-ffff-zzzzz //必须为masterKey |
| HTTP参数 | "imei":121, //设备模组IMEI，必填  "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  “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  “expired\_time”: "2018-03-09T17:30:00", //命令过期时间戳，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内容 | {  "args": "ping" //字符串  } |
| 成功返回 | {  "errno": 0,  "error": "succ",  "data":  {  "uuid": "42742677-adc3-54ca-83a1-5aaaf71482f8" //离线命令uuid  }  } |
| 说明 | 1. HTTP内容部分选填。 2. 响应消息中errno表示错误码，error表示错误原因。 |

#### 4.2.9.4查看指定设备离线命令列表

|  |  |
| --- | --- |
| 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 | {  "errno": 0,  "error": "succ"  "data":  {  "count": 100,  "items":  [  {  "cmd\_uuid":"f6869ecb-3dc1-5374-9be0-4fb961f8af3c ",  "type":"READ",  "create\_time":"2017-08-28 11:34:58",  "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"  {…}  ]  }  } |
| 备注 | 1、send\_time/confirm\_time/sid可能不存在  2、send\_status字段值代表意义如下：  ‘1’：命令等待（wait）  ‘2’：命令取消（cancel）  ‘3’：命令已发往设备（send）  ‘4’：命令过期（expired）  ‘5’：命令下发成功（success）  ‘6’：命令下发失败（failed）  ‘7’：命令响应超时（timeout）  ‘8’：其他未知错误（undefined） |
| 说明 | 1、响应消息中errno表示错误码，error表示错误原因。 |

#### 4.2.9.5查看指定离线命令详情

|  |  |
| --- | --- |
| 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 | {  "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",  "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  }  }} |
| 备注 | send\_status字段值代表意义如下：  ‘1’：命令等待（wait）  ‘2’：命令取消（cancel）  ‘3’：命令已发往设备（send）  ‘4’：命令过期（expired）  ‘5’：命令下发成功（success）  ‘6’：命令下发失败（failed）  ‘7’：命令响应超时（timeout）  ‘8’：其他未知错误（undefined） |
| 说明 | 1、响应消息中errno表示错误码，error表示错误原因。 |

#### 4.2.9.6取消离线命令

|  |  |
| --- | --- |
| 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 | {  "errno": 0,  "error": "success",  } |
| 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 | 1、响应消息中errno表示错误码，error表示错误原因。 |

### 4.2.10 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、更新和查看。另外对触发器API中ds\_id和dev\_ids参数说明如下：

* 在请求参数中单独指定了ds\_id，不包括其他参数，那么当前项目下所有设备的数据流中数据流名称符合ds\_id的数据都会进入触发判断逻辑；
* 在请求参数中指定了ds\_id和dev\_ids，那么只有这些指定的设备的数据流会进入触发判断逻辑。

#### 4.2.10.1新增

|  |  |
| --- | --- |
| HTTP方法 | POST |
| URL | http://api.heclouds.com/triggers |
| HTTP头部 | api-key:xxxx-ffff-zzzzz //必须为Master Key  Content-Type:application/json |
| HTTP内容 | {  "title":"wen du jian kong", //可选  "ds\_id":"3300\_0\_5505",//可选,数据流名称（id）。这个值是{obj\_id}\_{obj\_inst\_id}\_{res\_id}组成  "dev\_ids":["1027","23456","15523"], //可选，设备ID  "url":"http://xx.bb.com",  "type":">|>=|<|<=|==|inout|in|out|exp|change|frozen|live", //根据type不同，见以下备注说明  "threshold ":100  } |
| 请求返回 | {  "errno": 0,  "error":“succ”,  "data":  {  "trigger\_id":233444  }  } |
| 备注 | Type字段说明：  （1）type为>|>=|<|<=|==时，threshold必须为数值。  （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。  （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’] > 30 && $val[0][‘humidity’]< 33 ”。  （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。  （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。  （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。 |
| 说明 | 1、响应消息中errno表示错误码，error表示错误原因。 |

#### 4.2.10.2更新

|  |  |
| --- | --- |
| 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 | 1、若要修改触发条件，必须同时设置type和threshold。  2、Type字段说明：  （1）type为>|>=|<|<=|==时，threshold必须为数值。  （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。  （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’] > 30 && $val[0][‘humidity’]< 33 ”。  （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。  （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。  （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。 |
| 说明 | 1、响应消息中errno表示错误码，error表示错误原因。 |

#### 4.2.10.3查看

（1）单个查看

|  |  |
| --- | --- |
| HTTP方法 | GET |
| URL | http://api.heclouds.com/triggers/<trigger\_id> |
| HTTP头部 | api-key:xxxx-ffff-zzzzz //需要master Key  Content-Type:application/json |
| 请求返回 | {  "errno": 0,  "error":“succ”,  "data":{  "id":trigger\_id1,  "title":"wen du jian kong",  "url":"xx.bb.com",  "type":">|>=|<|<=|==|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”:"xx-xx-xx 10:22:22"  }  } |
| 备注 | 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）。  2、Type字段说明：  （1）type为>|>=|<|<=|==时，threshold必须为数值。  （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。  （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’] > 30 && $val[0][‘humidity’]< 33 ”。  （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。  （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。  （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。 |
| 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

|  |  |
| --- | --- |
| 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 | {  "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":">|>=|<|<=|==|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  }]  }  } |
| 备注 | 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）。  2、Type字段说明：  （1）type为>|>=|<|<=|==时，threshold必须为数值。  （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。  （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’] > 30 && $val[0][‘humidity’]< 33 ”。  （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。  （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。  （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。 |
| 说明 | 1、响应消息中errno表示错误码，error表示错误原因。 |

#### 4.2.10.4删除触发器

|  |  |
| --- | --- |
| 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 | {  "errno": 0,  "error":“succ”  } |
| 说明 | 1、响应消息中errno表示错误码，error表示错误原因。 |

### 4.2.11批量查询设备状态

|  |  |
| --- | --- |
| HTTP方法 | GET |
| URL | http://api.heclouds.com/ devices /status |
| UTL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 | {  "errno": 0,  "error":“succ”,  "data":  { "total\_count":121,  "devices":[  {  "id":12323,  "title":"daf",  "online":false  },  {…..},  …..  ]  }  } |
| 说明 | 1、响应消息中errno表示错误码，error表示错误原因。 |

### 4.2.12批量查询设备最新数据

|  |  |
| --- | --- |
| 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 | {  "errno": 0,  "error":“succ”,  "data":  {  "devices":[  {  "id":12323,  "title":"daf",  " datastreams ":[  {  "id":"temperature",  "at":"2017-02-12 10:22:22",  "value":12  },  {…..},  ……  ]  },  {…..},  …..  ]  }  } |
| 说明 | 1、响应消息中errno表示错误码，error表示错误原因。 |

### 4.2.13查看数据点

|  |  |
| --- | --- |
| HTTP方法 | GET |
| URL | http://api.heclouds.com/devices/<device\_id>/datapoints |
| URL参数 | 接口参数包括必选的模式选项和可选的公共参数。  可选的公共参数：  1.datastream\_id:涉及的数据流：datastream\_id=<datastream\_id>,多个数据流之间用逗号分隔；  2.start:表示提取数据点的开始时间，格式为2015-01-10T08:00:35  3.end:表示提取数据点的结束时间，格式为2015-01-10T08:00:35  4.limit:限定本次请求最多返回的数据点的数量，取值>0，<=6000  5.cursor:指定本次请求继续从cursor位置开始提取数据 |
| HTTP头部 | api-key:xxxx-ffff-zzzzz //需要设备级别的key |
| 请求返回 | {  "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":{…}}  ]  }，{…}]  }  } |
| 备注 | 使用方法说明：  （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。http://api.heclouds.com/devices/564280/datapoints 表示查询设备id为564280对应的所有数据流的最新数据点。  （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：  http://api.heclouds.com/devices/564280/datapoints?limit=10，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。  （3）要查看某一条数据流数据，在上述URL中增加参数datastream\_id。  （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：[http://api.heclouds.com/devices/564280/datapoints?datastream\_id=](http://api.heclouds.com/devices/564280/datapoints?datastream_id=3200_0_5501)  [3200\_0\_5501](http://api.heclouds.com/devices/564280/datapoints?datastream_id=3200_0_5501)，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。  （5）用<http://api.heclouds.com/devices/564280/datapoints?datastream_id=>  3200\_0\_5501&limit=10会返回数据流”温度2”中最近的10个数据点。  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。http://api.heclouds.com/devices/564280/datapoints?datastream\_id=  3200\_0\_5501&start=2013-05-12T17:12:33&end=2013-06-12T17:22:33  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。  （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。 |

### 4.2.14 查看数据点--示例一：

|  |  |
| --- | --- |
| 请求URL | http://api.heclouds.com/devices/564280/datapoints? datastream\_id=3200\_0\_55012&start=2015-11-30T17:12:33&end=2015-12-01T17:22:33 |
| 响应结果 | {"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":"2015-12-0117:18:08.904","value":"10"}],  "id":"3200\_0\_5501"}]},"error":"succ"} |

### 4.2.15 查看数据点--示例二：

|  |  |
| --- | --- |
| 请求URL | <http://api.heclouds.com/devices/564280/datapoints?datastream_id=>  3200\_0\_5501&start=2015-11-30T17:12:33&end=2015-12-01T17:22:33&limit=5 |
| 响应结果 | {"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"},  {"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"} |

### 4.2.16 查看数据点--示例三：

|  |  |
| --- | --- |
| 请求URL | http://api.heclouds.com/devices/564280/datapoints? datastream\_id=3200\_0\_5501&start=2015-11-30T17:12:33&end=2015-12-01T17:22:33&limit=5&cursor=25971\_564280\_1448961024981 |
| 响应结果 | {"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"} |

# 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：

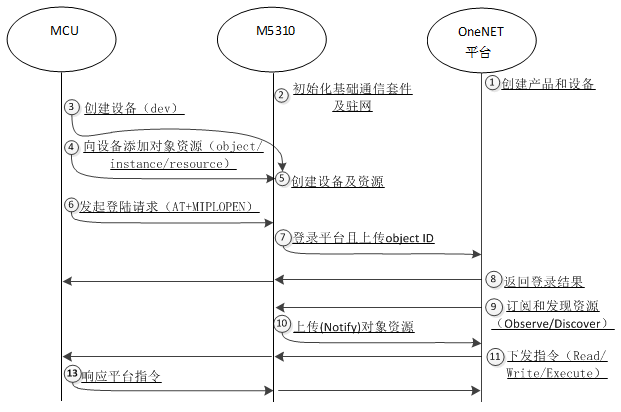
* 创建产品和设备；
* 响应登陆请求；
* 订阅和发现资源（Observe/Discover）；
* 下发指令（Read/Write/Execute）；

（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；



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：

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；

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>]；

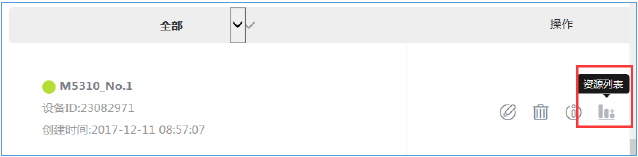
其中，<ref>：OneNET通信实例的引用ID；

<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。

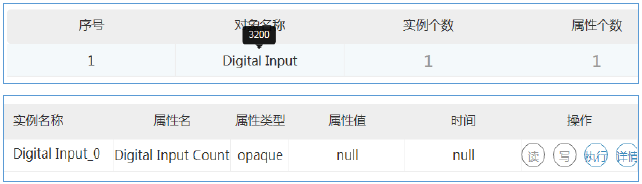
终端设备登录成功后，平台会下发 Observer消息和Discover消息。模组收到这两条消息之后，会自动处理，无需用户另行处理 。其中，Observer消息是平台传递的观测请求消息，模组收到该消息后会自动维护相关观测记录。Discover消息是上报通知模组需要获取指定Object的属性，上报后模组自动反馈相关属性。

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。



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。



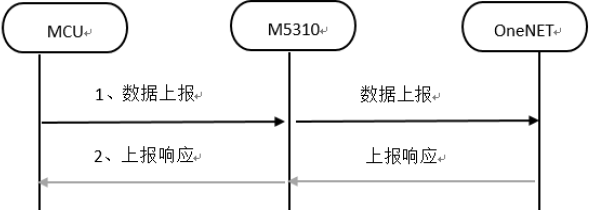
资源列表

## 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：



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。

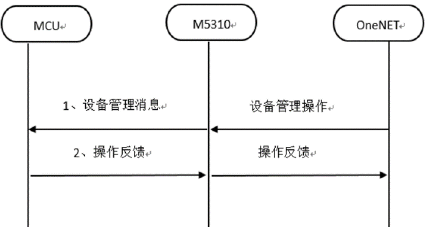
（2）由于模组M5310 的buffer资源限制，每次上报数据Payload用户数据部分不超过1000Bytes。

### 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。



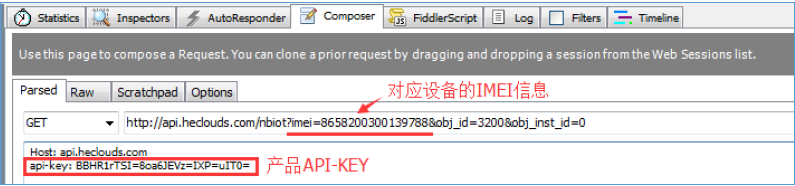
指令流程

#### 5.2.2.1 Read操作流程

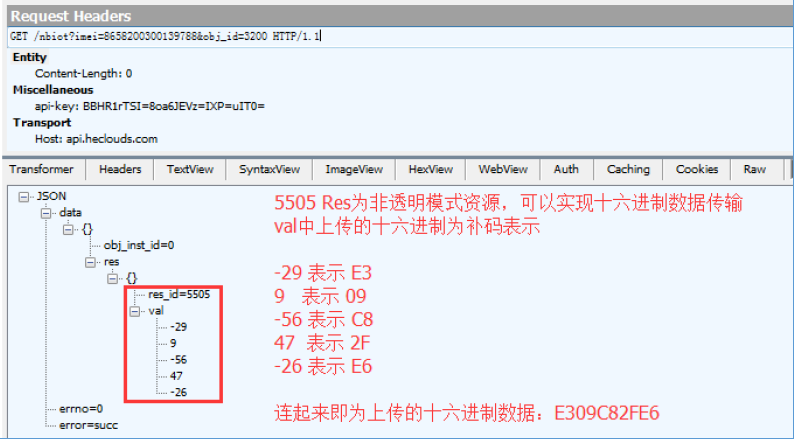
（1）应用侧

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。



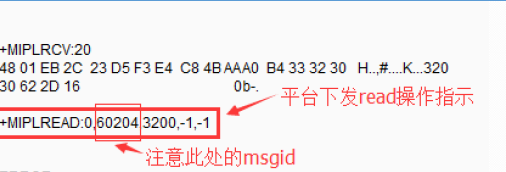
Fiddler调用读设备API



终端返回数据

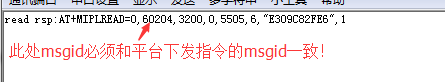
（2）终端侧

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。



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。



终端响应平台读指令

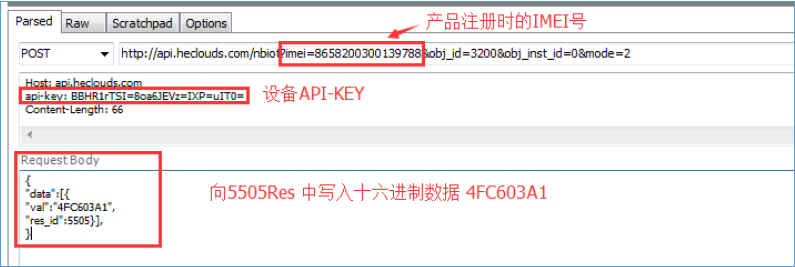
通过API参数的不同设置，还能实现读取指定Object下指定instance下的指定resource的数据；读取指定Object下所有instance的所有resource的数据等作。

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。

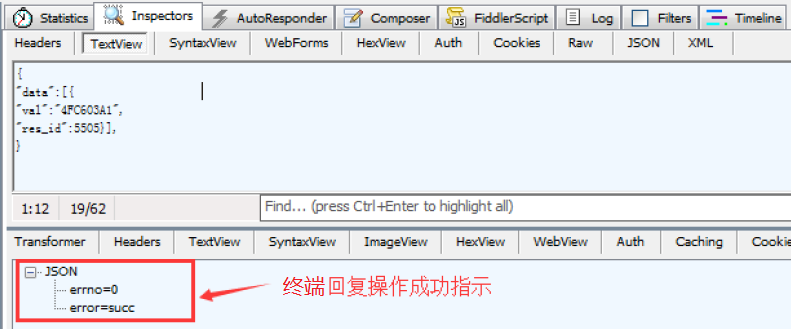
#### 5.2.2.2 Write操作流程

（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。



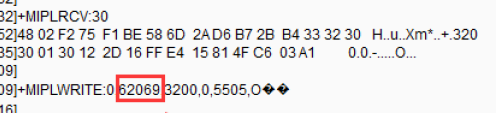
Fiddler调用写设备API



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。



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。



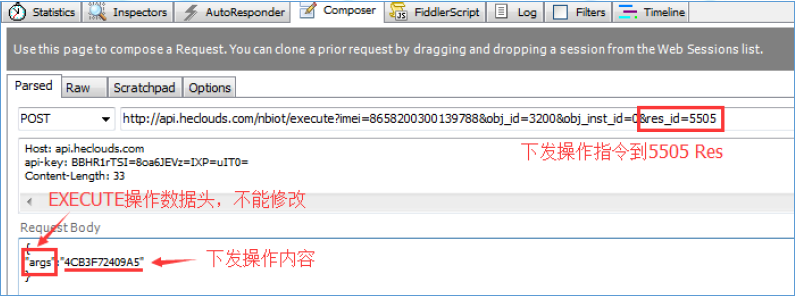
终端响应平台写指令

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。

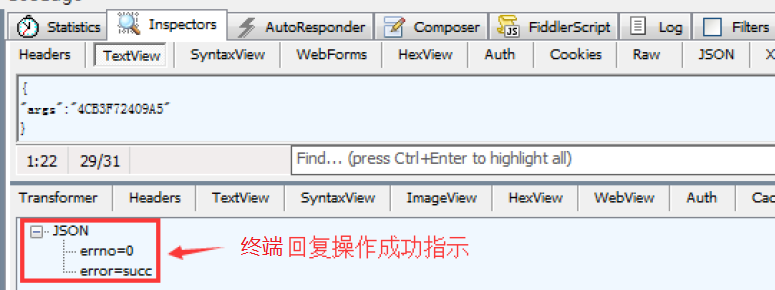
#### 5.2.2.3 Execute操作流程

（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5-15所示。



Fiddler调用执行命令API



终端返回执行成功消息

（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5-16所示。



平台下发执行指令

终端调用AT+MIPLEXECUTE指令响应平台的写指令，如图5-17所示。



终端响应平台执行指令

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

# 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

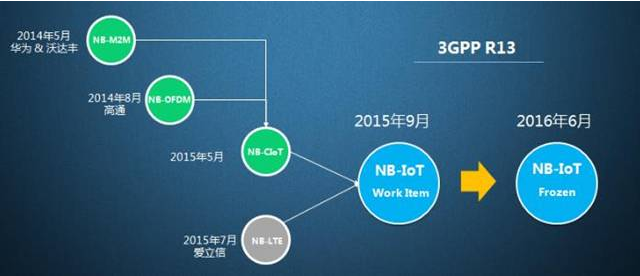
## 6.3 NB-IoT 开发板资料下载

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>

# 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。



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

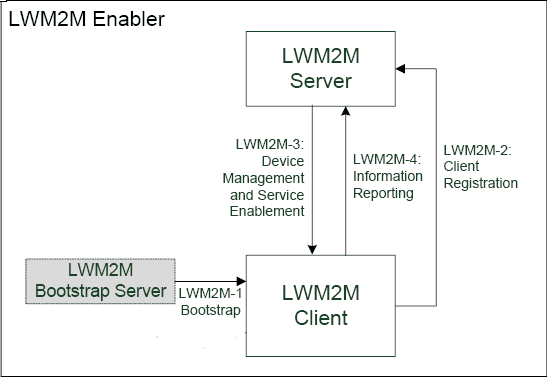
（1）LwM2M定义了三个逻辑实体：

* 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Clinet - 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。

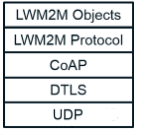
这三个逻辑实体与四个逻辑接口之间的关系如图7-2所示。



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。



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 |

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；

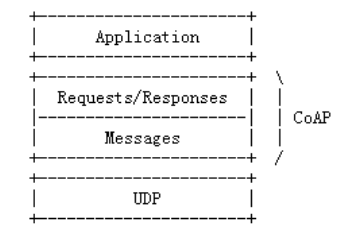
（4）支持可靠传输，通过确认和数据重传确保数据可靠到达；

（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。



CoAP协议栈

#### 7.2.1.1 CoAP 资源请求/响应模型

Requests：请求方法与HTTP协议类似，分别为： GET, PUT, POST和 DELETE。

Responses：响应内容也与HTTP协议类似，主要有以下3类：

（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。

#### 7.2.1.2 CoAP 消息报文定义

如下图7-5所示为CoAP消息报文示意图，CoAP消息报文是通过在UDP上传输消息完成，各部分说明如下：



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。

1. 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。

1. Payload

实际携带数据内容， 若有携带数据， 前面加payload 标志 OxFF。

### 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：

|  |  |  |
| --- | --- | --- |
| Number | Name | Reference |
| 23 | Block2 | RFC 7959 |
| 27 | Block1 | RFC 7959 |
| 28 | Size2 | RFC 7959 |

新增option说明

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

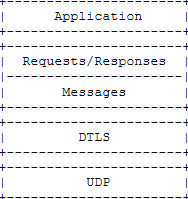
#### 新增加的2个response codes如表7-3所示：

|  |  |  |
| --- | --- | --- |
| Code | Description | Reference |
| 2.31 | Continue | RFC 7959 |
| 4.08 | Request Entity Incomplete | RFC 7959 |

新增response code说明

### 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。



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。

# 常见问题

（根据用户在接入工作过程中以周为单位进行新增）