OneNet平台对接说明

目录

[平台对接说明【必读】 1](#_Toc70093934)

[一、ONENET平台对接 1](#_Toc70093935)

[1、ONENET的平台的注册和准备 1](#_Toc70093936)

[1.1 ONENET注册 2](#_Toc70093937)

[1.2 注册页面 2](#_Toc70093938)

[1.3 登录后选择产品服务 ->基础服务 3](#_Toc70093939)

[1.4 进去后点击NB-IOT物联网套件的 立即使用 按钮 4](#_Toc70093940)

[1.5 添加产品 4](#_Toc70093941)

[1.6 添加设备 7](#_Toc70093942)

[2、欧孚设备对接ONENET平台 9](#_Toc70093943)

[2.1 关于指令下行 11](#_Toc70093944)

[3、欧孚设备数据协议 13](#_Toc70093945)

[4、对设备通信协议的补充说明 13](#_Toc70093946)

[4.1十六进制介绍 13](#_Toc70093947)

[4.2对于协议中字段定义的说明 13](#_Toc70093948)

[4.3多字节解析说明 14](#_Toc70093949)

[4.4关于A3和A2协议解析的说明 14](#_Toc70093950)

[5、ONENET上行数据接口和下行接口的实现 17](#_Toc70093951)

[5.1 上下行概念说明 17](#_Toc70093952)

[5.4 调用ONENET接口下行代码 17](#_Toc70093953)

[二、设备Wifi数据解析【必读】 20](#_Toc70093954)

[三、上行数据解析和下行指令配置说明【必读】 20](#_Toc70093955)

平台对接说明【必读】

1、最新版本下载地址：<http://aiday.com.cn/Help/api/Device/>

**2、文档已经详细说明，仔细阅读就可以完成对接（原则上欧孚不提供对接技术支持）**

一、ONENET平台对接

1、ONENET的平台的注册和准备

客户自行申请移动ONENET平台账户，并在ONENET平台设备按照如下填写相关的信息。移动ONENET平台具体请参考 <https://open.iot.10086.cn/doc/>

01、设备数据通过ONENET接入自己的服务器后因定位wifi和基站解析需要使用到第三方服务解析，推荐使用高德地图提供的【智能硬件行业解决方案<https://lbs.amap.com/smart/smarthard>】需要注册高德开发者账号做企业认证【企业认证可提高解析次数】。

02、通过ONENET提供的接口下发指令到设备 需申请应用管理-> 新增应用>获取APP key 及App Secret【具体请详细阅读ONENET提供的相关文档】

1.1 ONENET注册

**ONENET注册地址：**[**https://** **open.iot.10086.cn/**](https://www.ctwing.cn/)

注册 账号和密码



1.2 注册页面



然后点击登录



1.3 登录后选择产品服务 ->基础服务

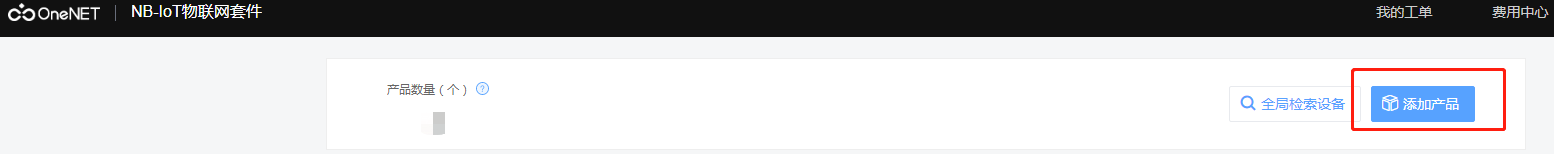


1.4 进去后点击NB-IOT物联网套件的 立即使用 按钮

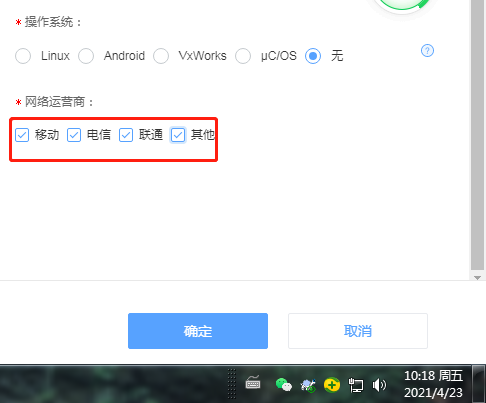


1.5 添加产品

点击产品中心 进入添加产品的页面。下图就是添加产品的页面和关键要领



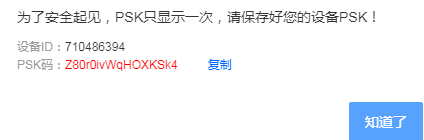






1.6 添加设备

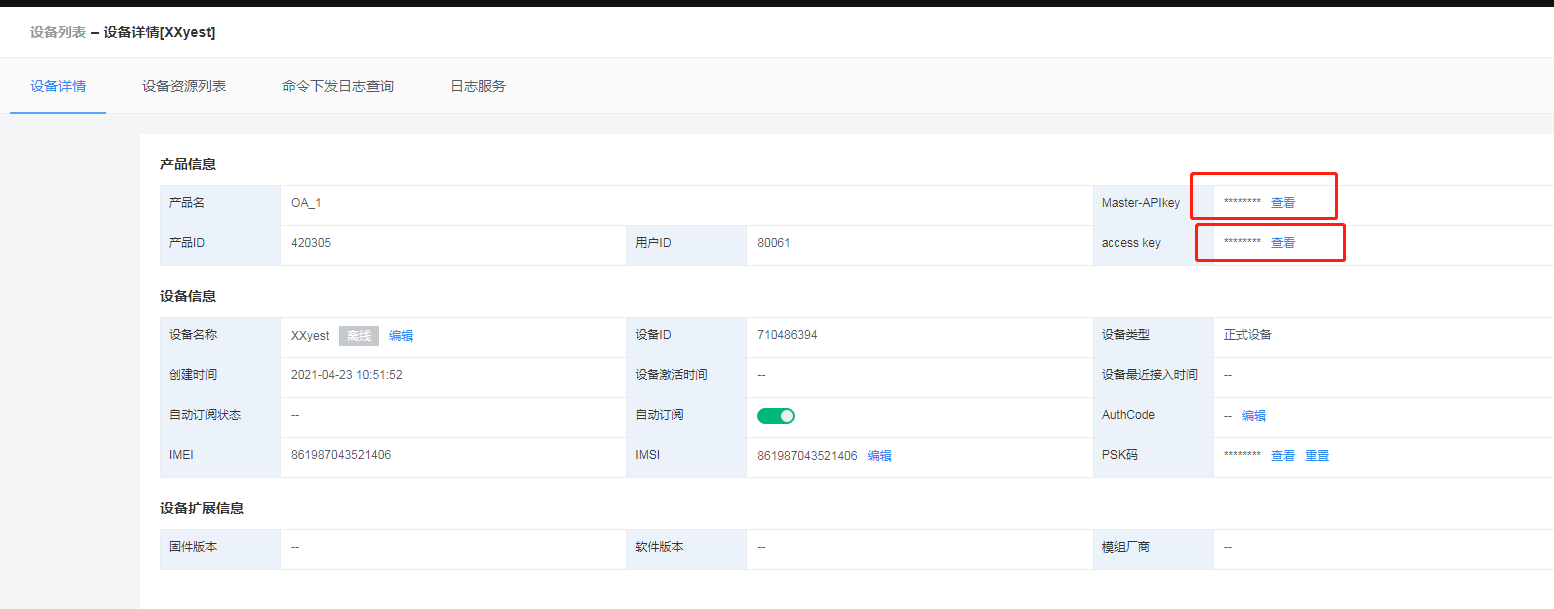




添加完成后注意保存设备ID，状态显示为离线

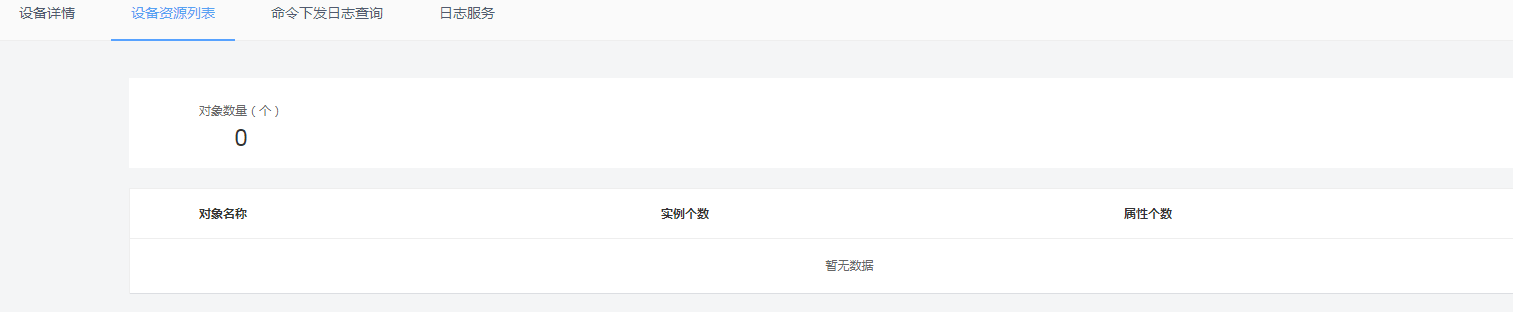
****

点击详情后 进入下面界面

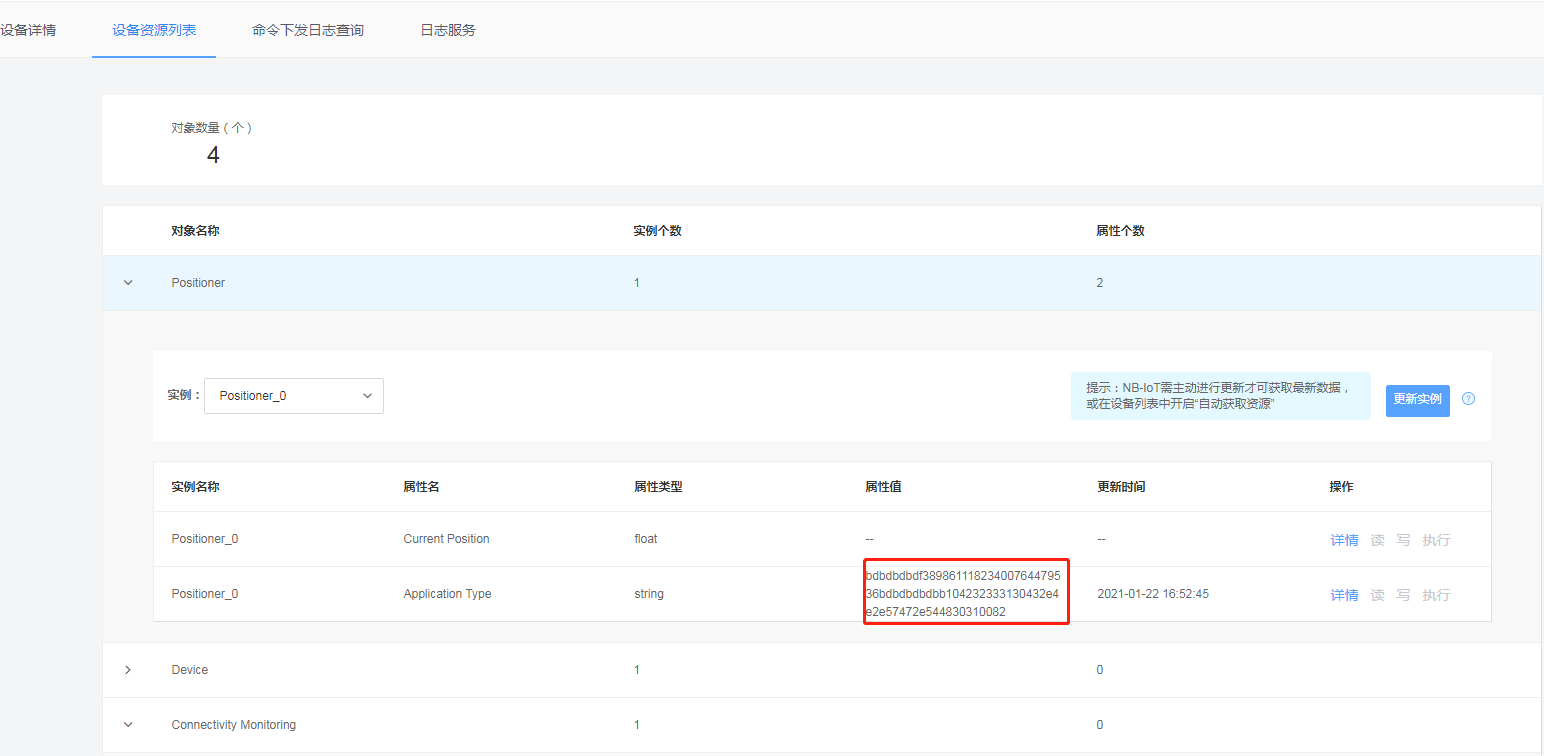


保留Master-APIKey 和access-key

设备没有和平台通信时 资源列表 是这样



设备和平台通信后在positioner 会看到这些 设备上报的数据



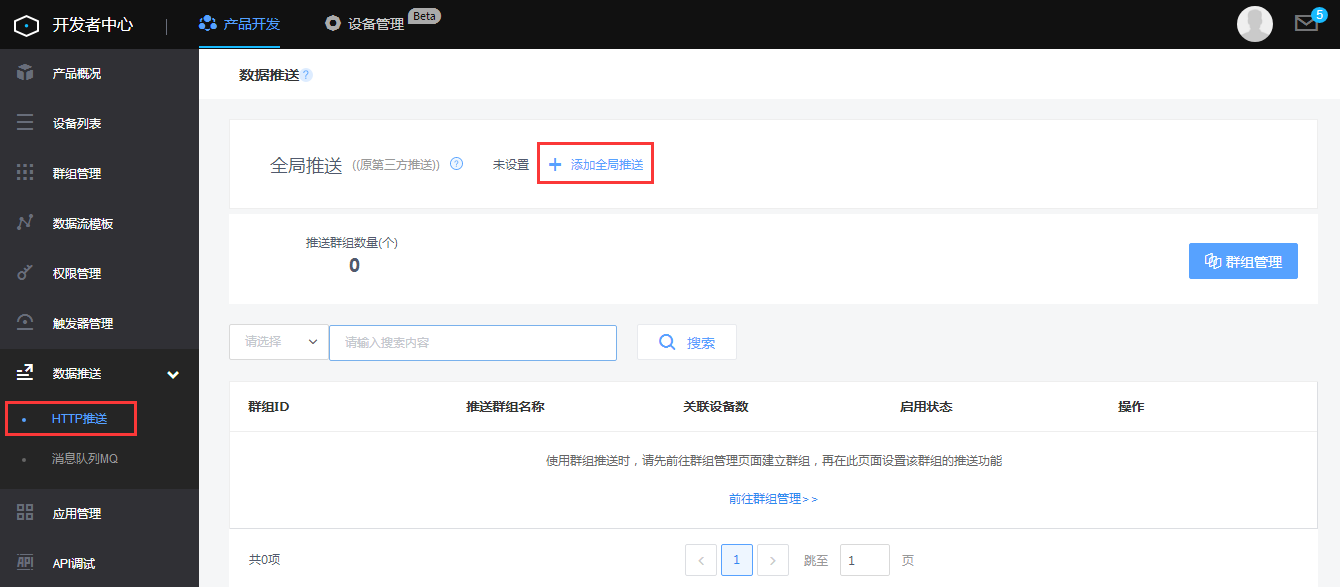
这样 就完成了设备和onenet平台的通信了

2、欧孚设备对接ONENET平台

1：点击页面左侧 数据推送 选择http 推送

2： 启用全局推送 点击添加全局推送

3：填写推送地址 和token



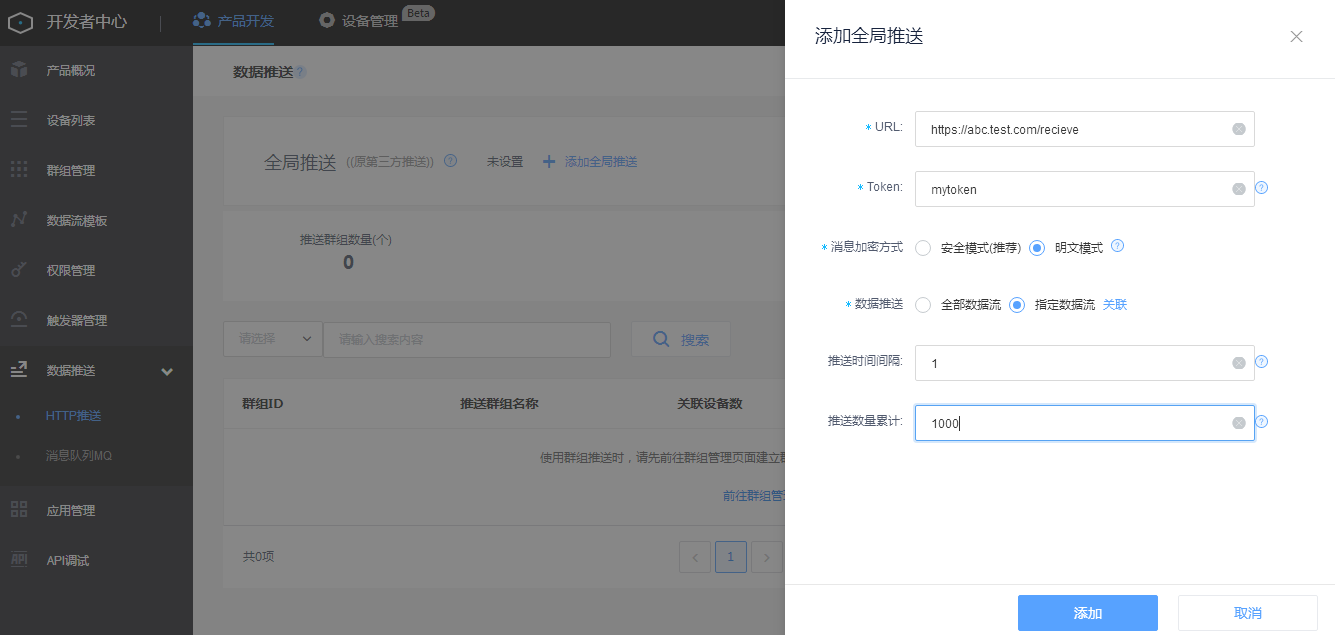
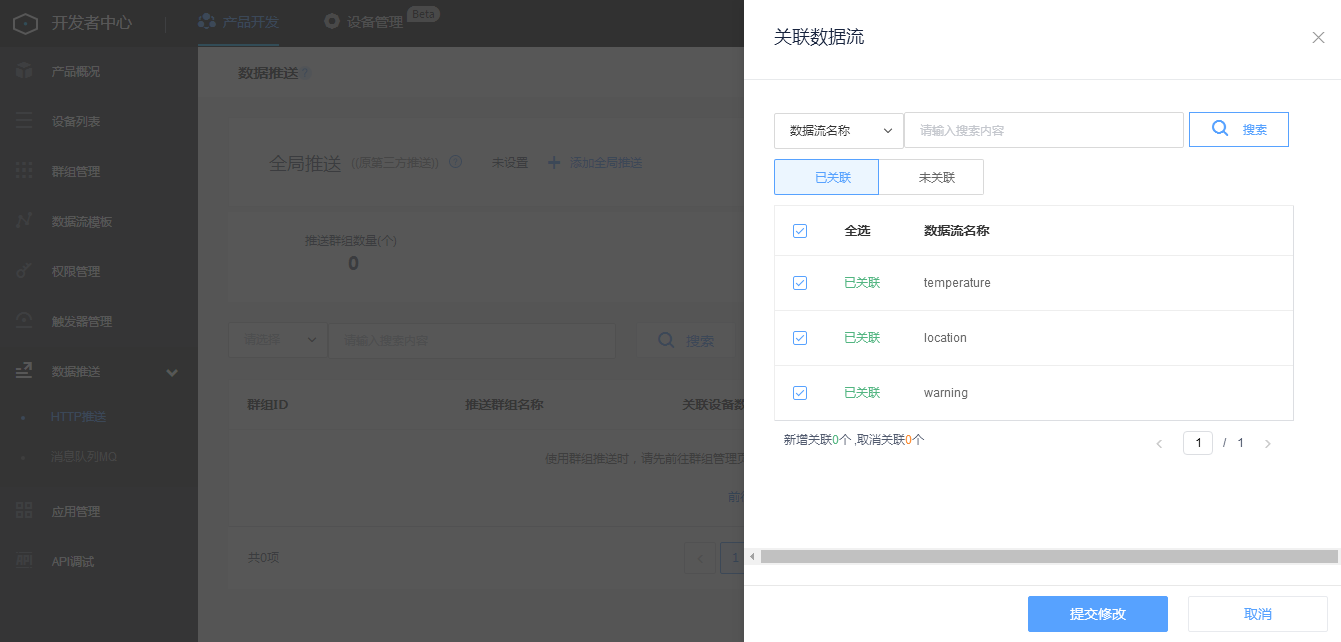
URL：用于接收数据的服务器URL地址；

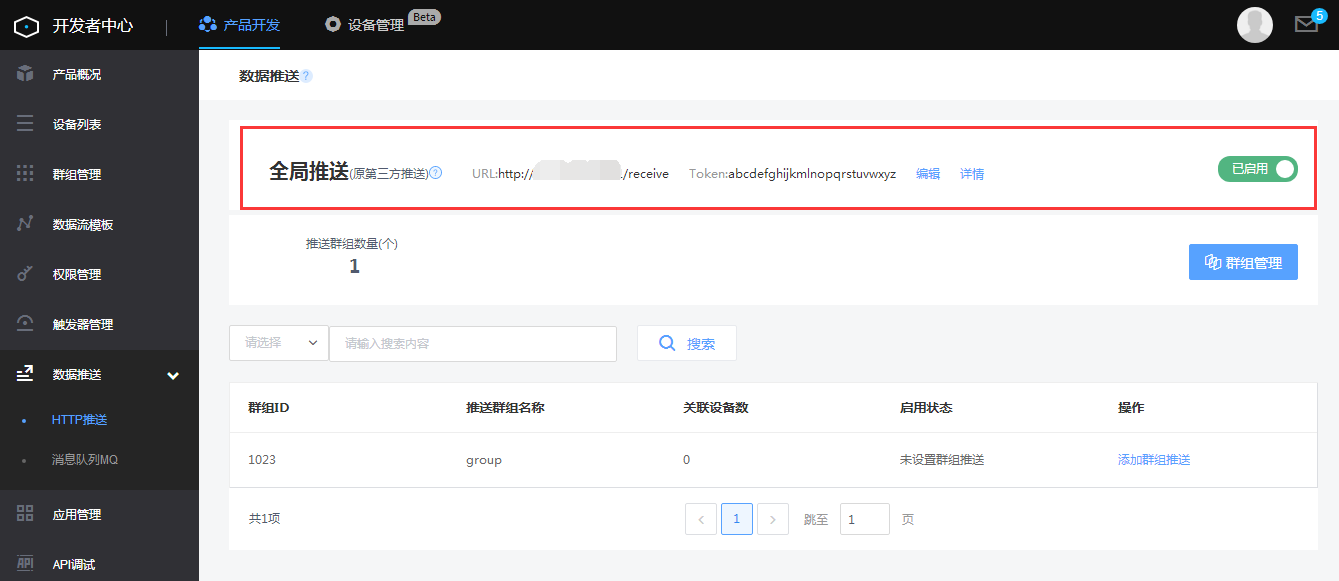
Token：自定义token,可用于验证请求是否来自OneNET；

选择指定数据流 要保存好数据流ID 指令下行时 有用 按下面操作

指定数据流：点击**关联**，选择需要推送新增数据点的数据流。如下图所示：







部署应用服务器接收服务，点击**添加**，成功后可以看到该服务处于**已启用**状态，如下图所示：（前提是 url 已在本地服务器上部署好 用户在配置页面完成配置并点击“提交”时，OneNET平台会向填写URL地址发送HTTP **GET**请求进行URL验证）

请求形式示例如下：

http://url?msg=xxx&nonce=xxx&signature=xxx

这部分比较长 参见 <https://open.iot.10086.cn/doc/nb-iot/book/manual/httppush/develop-manual.html>

在这个页面上 提供了SDK 开发文档 <https://open.iot.10086.cn/doc/nb-iot/book/manual/httppush/sdk.html>

这样就可以在自己的平台上接收数据了

public String OneNetReceive()

{

Stream req = Request.InputStream;

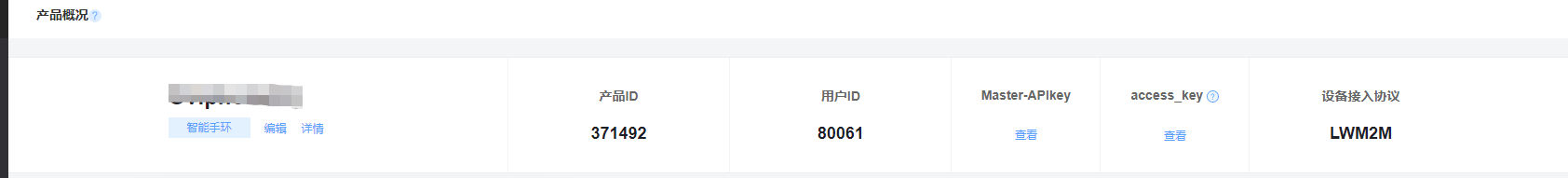
req.Seek(0, System.IO.SeekOrigin.Begin);

string body = new StreamReader(req).ReadToEnd();

….. // 随意解析即可

}

2.1 关于指令下行



在平台上 设备通信后这么下行指令





点击上图确定后 就可以在设备上观察结果了

在自己服务器上这么下行指令 文档参考

<https://open.iot.10086.cn/doc/nb-iot/book/application-develop/list/11cache-comm-issue.html>

由于采用https 下行 部分包装做几点说明

url "https://api.heclouds.com/nbiot/?imei="+dev.imei+"&obj\_id=3311&obj\_inst\_id=1&mode=2";



1：Onenet写命令 就是 写资源 这点区别于下面的命令下发 这点要注意

2： 如果还不清楚，可以调用 即时命令 资源发现 读取设备对应的obj\_inst\_id 这样也可下发

代码中要用到 headers.Add("api-key",APIKey);

3 后面附录一段下行代码

参数说明

设备下行指令需使用平台提供的接口下发，有固定的格式定义.

我们大部分会设置产品上报周期的指令

设备协议里是这么定义的

BDBDBDBD170101000000173B000000000000000000000000000000000000000000A0

17 就是下行指令 后面01 是启用 再后面0100 是小端优先 实际是0001 就是1分钟一次

后面0000就是0点0分 173B就是23点59分 后面可以再跟着设置

但这个发给平台也是不认的，平台指令下行规定的格式是这样的

3、欧孚设备数据协议

相关协议文件请到该链接：<http://aiday.com.cn/Help/API/Device/Protocal/>【欧孚通信NBIOT设备COAP协议】和【欧孚通信设备通信协议】

4、对设备通信协议的补充说明

4.1十六进制介绍

**十六进制**（简写为*hex*或下标16）在[数学](https://baike.baidu.com/item/%E6%95%B0%E5%AD%A6/107037)中是一种逢16进1的[进位制](https://baike.baidu.com/item/%E8%BF%9B%E4%BD%8D%E5%88%B6/6509260)。一般用数字0到9和字母A到F（或a~f）表示，其中:A~F表示10~15，这些称作**十六进制数字**。

详情可查阅该链接：

<https://baike.baidu.com/item/%E5%8D%81%E5%85%AD%E8%BF%9B%E5%88%B6/4162457?fr=aladdin>

4.2对于协议中字段定义的说明

U8,i8,x8 都是单字节 对应的类型不同 分别对应 无符号短整型 ，短整型，和二进制位存的数据类型

U16,i16,x16 都是双字节 对应的类型不同 分别对应 无符号短整型 ，短整型，和二进制位存的数据类型

U32,i32 都是四字节 对应的类型不同 分别对应 无符号长整型 ，长整型 用于存放时间戳

U64 8字节长

Float 4字节长 运用存放有小数点的数据

对于协议中字段定义的说明

4.3多字节解析说明

由于数据传输分为两种模式

**大端模式：**字数据的高字节存储在低地址中，而字数据的低字节则存放在高地址中。

**小端模式：**与大端存储模式相反，低地址中存放的是字数据的低字节，高地址存放的是字数据的高字节。

欧孚设备协议现使用的是小端模式，也就是一个数如果真实值 0x3256在实际通信时，指令格式中会显示为0x5632。多字节也是这样前后数据对调.

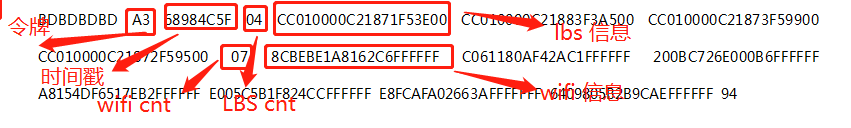
注意：解析数据时对照协议只要是多字节就要注意这个问题尤其是定位数据的时间戳和wifi、基站的信号强度等.

4.4关于A3和A2协议解析的说明

A3和A2大致相同 A3多了一个时间戳

下面举一个例子 简单说明一下

BDBDBDBDA3 68984C5F04CC010000C21871F53E00CC010000C21883F3A500CC010000C21873F59900CC010000C21872F59500078CBEBE1A8162C6FFFFFFC061180AF42AC1FFFFFF200BC726E000B6FFFFFFA8154DF6517EB2FFFFFFE005C5B1F824CCFFFFFFE8FCAFA02663AFFFFFFF6409805B2B9CAEFFFFFF94



图中对各个字段都加了标注 lbs cnt 基站个数 lbs 信息 基站信息 wifi cnt wifi个数 wifi 信息

LBS: CC010000C21871F53E00 wifi:8CBEBE1A8162C6FFFFFF

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | MCC | MNC | LAC | Cell-id | rssi |
| 显示值 | CC01 | 0000 | C218 | 71F5 | 3E00 |
| 实际值 | 01CC | 0000 | 18C2 | F571 | 003E |
| 十进制 | 460 | 0 | 6188 | 5233 | 62 |

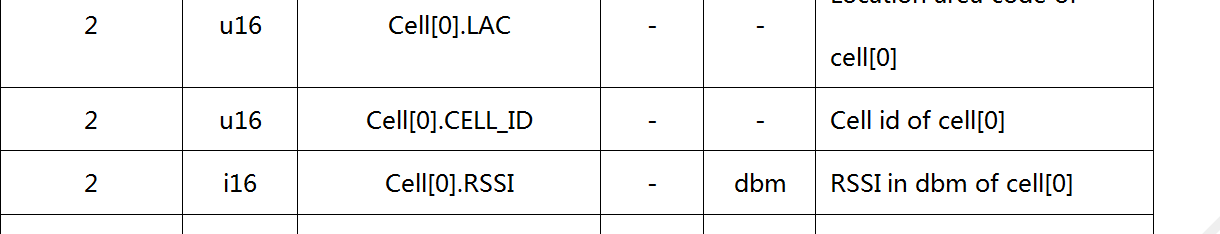
|  |  |  |
| --- | --- | --- |
|  | Mac地址 | 信号强度 |
|  | 8C:BE:BE:1A:81:62 | C6 FFFFFF |
|  | 8C:BE:BE:1A:81:62 | FFFFFF C6 |
|  | 8C:BE:BE:1A:81:62 | -58 |

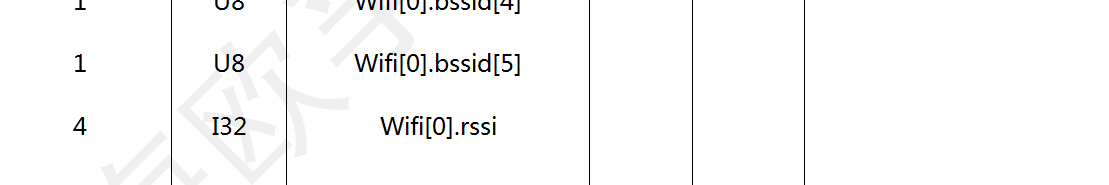
有些版本收到wifi的信号强度可能就是FFFFFFC6 这是以前未统一引起的

时间戳 ：是从1970-01-01 00:00:00 到发送时间的秒数rssi 003E wifi信号强度是 C6 都是补码

同时也可理解u和i的区别

对应设备协议文档规定： wifi的rssi 协议规定是4字节的整形,实际值只有0到-100多，正常解析显示值高位即可





调用的高德接口说明：



<addkey="AmapLbsApiUrl" value="http://apilocate.amap.com/position?accesstype=0&amp;imei={0}&amp;cdma=0&amp;bts={1}&amp;nearbts={2}&amp;output=json&amp;key="/>

<add key="AmapWifiApiUrl" value=" http://apilocate.amap.com/position?accesstype=1&amp;imei={0}&amp;mmac={1}&amp;macs={2}&amp;output=json&amp;key="/>

<add key="AmapMixWifiLbsUrl" value=" http://apilocate.amap.com/position?accesstype=1&amp;imei={0}&amp;mmac={1}&amp;macs={2}&amp;bts={3}&amp;nearbts={4}&amp;output=json&amp;key="/>

4.5关于float 或者double的转换

C# BitConverter.ToDouble(byte[],0);

java Double.longBitsToDouble(toLong(bytes[])) 去转换

5、ONENET上行数据接口和下行接口的实现

5.1 上下行概念说明

数据上行：指设备上行数据到移动平台。然后到自己的服务器

数据下行: 指平台下发指令到设备.

收到上述消息后，请立即返回下述下述代码

return "ok";

数据下行接口 最好参照OneNet的sdk开发

5.4 调用ONENET接口下行代码

指令下行必须现在平台上添加应用和设备后才能取到接口开发时所需的app key 和App Secret。

public static string CreatePostHttpResponse(string url, Dictionary<string, string> headers, string data, string imei)

{

// https://api.heclouds.com/nbiot/?imei="+dev.imei+"&obj\_id=3311&obj\_inst\_id=1&mode=2"

// headers.Add("api-key",APIKey)

// data 就是BDBDBDBD。。。。。

string html = "";

try

{

HttpWebRequest request = null;

request = WebRequest.Create(url) as HttpWebRequest;

request.ProtocolVersion = HttpVersion.Version10;

if (headers != null && headers.Count > 0)

{

foreach (var item in headers)

{

request.Headers.Add(item.Key, item.Value);

}

}

request.Method = "POST";

request.ContentType = "application/x-www-form-urlencoded";

request.UserAgent = "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36";

byte[] strBytes = System.Text.Encoding.UTF8.GetBytes(data);

using (Stream stream = request.GetRequestStream())

{

stream.Write(strBytes, 0, strBytes.Length);

}

var response = request.GetResponse() as HttpWebResponse;

using (StreamReader responseStream = new StreamReader(response.GetResponseStream()))

{

html = responseStream.ReadToEnd();

}

response.Close();

//response.Dispose();

}

catch (Exception ex)

{

OnHelper.OnLogHelper.WriteNbLog(ex.Message, imei);

}

return html;

}

//A3的时间解析 采用协议最后使用的例子的utc时间 546f0d5f

import java.text.SimpleDateFormat;

import java.util.Calendar;

import java.util.Date;

public static void main(String[] args) {

String utcTime = "5F0D6F54";

Date date=new Date();

date.setTime(Long.parseLong(utcTime,16)\*1000);

SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMddHHmmss");

System.out.println(sdf.format(date));

}

下面再举一个java 关于GPS的示例

报文：DBDBDBDB037d9f84ac81815c40e766926b1d8936404e4541749d695f0b

DBDBDBDB03 7d9f84ac81815c40 e766926b1d893640 4e 45 41 749d695f 0b

public static void main(String[] args) {

//报文7d9f84ac81815c40 实际值405c8181ac849f7d

Double.longBitsToDouble(Long.parseLong("405c8181ac849f7d", 16))); //114.02353966666665

//报文e766926b1d893640 实际值4036891d6b9266e7

Double.longBitsToDouble(Long.parseLong("4036891d6b9266e7", 16))) ;//22.535605166666667

HexToStr(data.Substring(“4e”)); //N

HexToStr(data.Substring(“45”)); //E

HexToStr(data.Substring(“41”)); //AA表示数据"OK"，V表示一个警告

//报文 749d695f实际值 5f699d74

Date date=new Date();

date.setTime(Long.parseLong(“5f699d74",16)\*1000);

SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMddHHmmss");

System.out.println(sdf.format(date)); //2020-09-22 14:45:08

}

二、设备Wifi数据解析【必读】

1、设备传输的WIfi定位数据是Wifi MAC地址和信号强度，请百度“Wifi定位原理”了解。

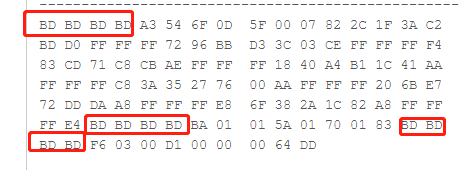
2、Wifi或基站数据解析：以高德wifi数据库为例（若使用其他家Wifi数据库解析请自行探索）提供的【智能硬件解析方案】，该方案使用需要在高德开放平台上提交申请（网址是：<https://lbs.amap.com/smart/smarthard>）；

另注册企业开发者账号可以增加调用接口数，高德Wifi定位数据协议请参考文档：

三、上行数据解析和下行指令配置说明【必读】

1、考虑到设备现在采用的低功耗，报文往往会并包发送(即一条数据中存在多组BDBDBDBD的情况.).

示例如下图：



2、给设备下行指令时则不存在并包发送问题，参考我们协议编码后仅一组BDBDBDBD转成base64编码内容即可.

3、下行指令接口中超时时间建议设置为成1-2个心跳时间，这样确保设备上行数据时平台指令不至于失效造成下发指令无效.