**新大陆物联网云平台**

**TCP透传—设备接入协议**

**修订记录**

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **修订版本** | **修改描述** | **作者** |
| 2019-04-30 | V1.0 | 添加 | wh |

# 说明

便于用户能够快速地接入设备到NLECloud平台，故编写此说明文档。

TCP透传具有较高自由度，它允许用户自定义协议，并根据用户定义的脚本完成与其他协议的交互。

# 接入流程

**第1步 创建项目及项目下的TCP透传设备**

登录NLECloud平台进入开发者中心，根据实际情况创建自己的项目及TCP透传设备如下图所示。

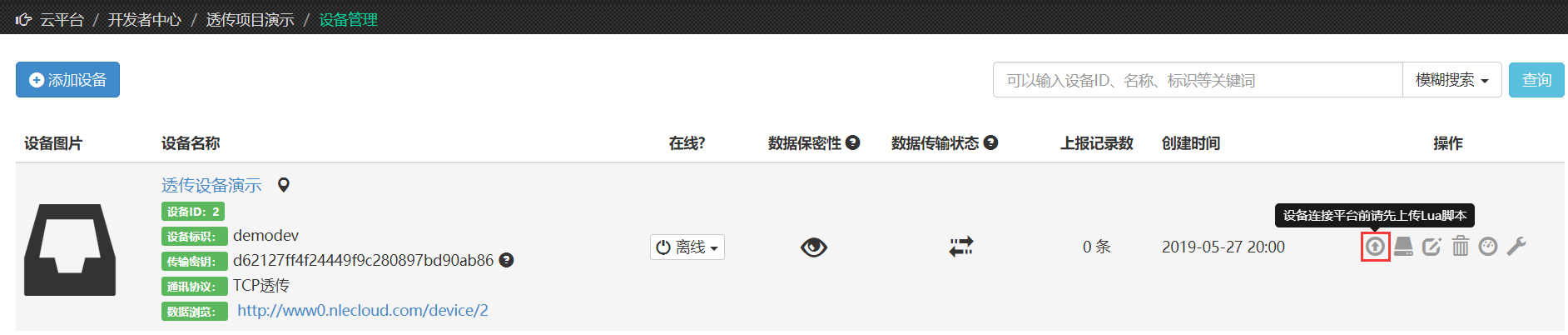
并在设备下添加相关的传感器件

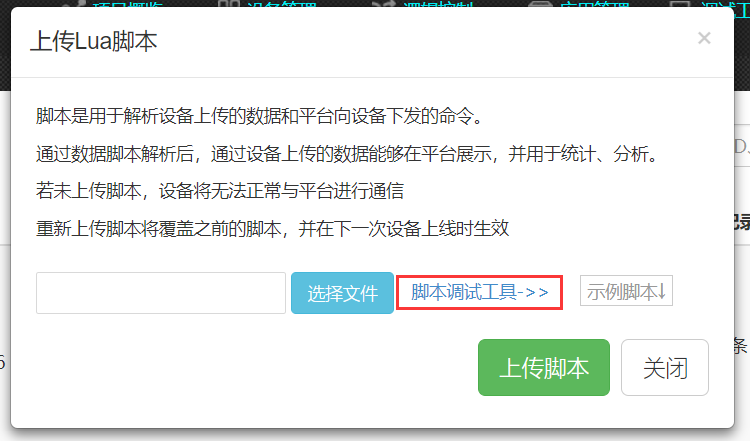


**第2步 编写Lua脚本**

[LUA脚本编写说明](#_LUA脚本编写说明)

**第3步 验证并上传解析脚本**







验证获得需要的结果后即可选择目标Lua脚本文件，点击上传脚本。

**第4步 设备发送握手连接请求**

在当设备建立TCP连接到指定端口(5300)后，需要发送握手连接请求报文接入云平台，请求报文格式如下：

**{**

**"**t**"**: 1,

**"**device**"**: **"**P123456789**"**,

**"**key**"**:**"**9861d43a0733415ab5424ee7d0f1c685**"**,

**"**ver**"**:**"**v1.1**"**

**}**

|  |  |  |  |
| --- | --- | --- | --- |
| **JSON键** | **JSON值** | **说明** | **消息示例** |
| t | 1 | 固体数字1，代表连接请求 |  |
| device | 设备标识 | 在平台上添加设备时的设备标识：   1. 新大陆网关：进入网关设置-》【参数设置】-》【系统参数】中的序列号 2. 新大陆农业网关：浏览器登录农业网关设置页面-》【设备状态】中的设备编号 3. 新大陆家居网关：进入平板的家居网关主界面，界面左上角的一行序列号 4. **其它的MCU/SOC/网关/手机等设备：可自行输入一个唯一的标识用于与平台连接** | PF12345678 |
| Key | 传输密钥 | 在平台上添加设备时自动生成的一串字符串（32位长度的字符串），该值在全局内具备唯一性； | 9861d43a0733415ab5424ee7d0f1c685 |
| ver | 客户端代码版本号 | 可以是自己拟定的一组客户端代码版本号值 | V1.1 |

握手成功且Lua脚本符合编写规范即成功接入平台进行正常通信。

# LUA脚本编写说明

用户必须在Lua脚本中实现：

数据上报解析函数device\_data\_analyze(data)

平台命令下发解析函数cmd\_send\_analyze(cmd)

* 1. **解析设备数据上报**

用户需实现Lua函数device\_data\_analyze(data)，以完成对设备上传数据的解析，返回值为json。

其中：

data为一个user\_data类型的值，提供了以下几个函数：

1. data: Size()

获取报文数据大小（字节数）。

@param 无

@return 返回报文数据大小（字节数）

@example local sz = dev:Size()

1. data:Byte(pos)

获取pos对应位置的报文数据（字节）。

@param pos number 指定的获取位置，取值范围[1,dev:size()]

@return 成功返回报文数据（int），否则返回nil

@example local data = dev:byte(1)

1. data:Bytes(pos,count)

获取从pos开始，count个报文数据。

@param pos number 指定的获取起始位置，取值范围[1,dev:size()]

count number 指定的获取数据总数，取值范围[1,dev:size()+1-pos]

@return 成功返回报文数据（string），否则返回nil

@example local datas = dev:bytes(1,dev:size())

json表示脚本解析后的报文，

数据上报需解析的格式如下：

{

"t": 3,

"datatype":1~3,

"datas": { 见下表说明 } 或 [ 见下表说明 ],

"msgid": 123

}

|  |  |  |  |
| --- | --- | --- | --- |
| **JSON键** | **JSON值** | **说明** | **报文示例** |
| t | 3 | 固体数字3，代表数据上报 | 3 |
| datatype | 数据上报格式类型 | 具体为datas属性内的 传感数据格式类型，如下  = 1：JSON格式1字符串；  = 2：JSON格式2字符串；  = 3：JSON格式3字符串； | 1 |
| datas | 要上报的传感数据数组 | 该属性根据datatype类型的不同，可以上报多个传感器数据，也可以上报同一传感器的多条数据，其中**apitag1**为传感的标识名，**value**为传感值，可以是数字、浮点、字符串、二进制（最大48字节大小）  *数据类型为1（JSON格式1字符串）：*  **"**datas**"**:  {  **"**apitag1**"**: **"**value1**"**,  **"**apitag2**"**: value2,  …  } | 示例：  **"**datas**"**:  {  **"**temperature**"**: 23.5,  **"**rgb-r**"**: **"**#999**"**,   …  } |
| *数据类型为2（JSON格式2字符串）：*  apitag1与value数据格式同上，datetime1须是yyyy-mm-dd hh:mm:ss格式  **"**datas**"**:  {  **"**apitag1**"**:{**"**datetime1**"**:**"**value1**"**},  **"**apitag2**"**: {**"**datetime2":"value2"},  …  } | 示例：  "datas":  {  "temperature": {"2015-03-22 22:31:12”:22.5},  …  } |
| *数据类型为3（JSON格式3字符串）示例：*  value数据格式同上，  dt须是yyyy-mm-dd hh:mm:ss格式  "datas":  [  {  "apitag":"temperature",  "datapoints":  [  {  "dt":"2018-01-22 22:22:22", //可选  "value": 36.5 //数字浮点字符串  }  ]  },  {  "apitag": "location",  "datapoints":  […]  },  { … }  ] |  |
| msgid | 消息编号 | 由客户端生成的一个用于表示该条报文的编号，用于服务器下发“上报响应”时原样带回 | 123 |

若设备能够获取并处理通过平台下发的命令编号，即可在解析函数device\_data\_analyze(data)中根据自定义的判断实现数据上报和命令响应的双重解析。

命令响应需解析的格式如下

{

"t": 6,

"cmdid": 123

"status":0,

"data": { 见下表说明 }

}

|  |  |  |  |
| --- | --- | --- | --- |
| **JSON键** | **JSON值** | **说明** | **消息示例** |
| t | 6 | 固体数字6 | 6 |
| cmdid | 命令编号 | 命令下发时的编号，原样返回 | 123 |
| status | 状态结果 | 一个字节表示  0：处理成功；  1：处理失败；  其它：保留值； | 0 |
| data | 命令响应值 | 响应值，可以是数字、浮点、字符串、JSON | 同上 |

* 1. **平台命令下发解析**

用户需实现Lua函数cmd\_send\_analyze(cmd)，以完成平台向设备下发命令数据的解析，

其中：

cmd为一个类型对象，它包含以下几个属性：

cmdid ：命令编号

@example local cmdID= cmd.cmdid

apitag：执行器标识

@example local apiTag = cmd.apitag

data：命令内容

@example local data = tostring(cmd.data)

返回值：设备所能处理的自定义命令。