**TCP透传-快速接入文档**

**修订记录**

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **修订版本** | **修改描述** | **作者** |
| 2017-02-09 | V1.0 | 添加 | XiaoBo |
| 2017-02-14 | V1.1 | 添加部分Lua函数说明 | XiaoBo |
| 2017-03-29 | V1.2 | 支持同一连接传输多个设备的数据支持API下发命令 | XiaoBo |
| 2017-04-01 | V1.3 | Lua支持获取时间戳、年月日、时分秒 | XiaoBo |
|  |  |  |  |

**缩略语清单**

|  |  |  |
| --- | --- | --- |
| **缩略语** | **英文全名** | **中文解释** |
|  |  |  |
|  |  |  |

# 说明

便于用户能够快速地接入设备到ONENET平台，故编写此说明文档。

# 名词解析

PID：产品ID，创建产品时OneNET生成的产品唯一性数字标识

AuthCode: 设备鉴权码，在设备注册时，用户创建设备时指定的唯一字符串标识

ParserName: 用户自定义解析脚本的名称，用户上传脚本时指定的唯一字符串标识

# 接入流程

**第1步 创建产品**

登录ONENET平台进入开发者中心，选择公共协议产品，根据实际情况创建自己的产品（此文档以dtu-test为例），如下图所示。





**第2步 创建设备**

 进入dtu-test产品的“设备管理”，进行添加设备，（此文档以 “test”举例说明）,如下图所示:









 同时，也可以进入dtu-test产品的“在线调试》API调试工具”， 参照文档中心下的“应用开发创建设备”说明，创建设备，（此文档以test\_ds举例说明）

 请求内容如下：

{

 "title" : "test",

 "auth\_info" : "test"

}

 返回结果：

{

 "errno": 0,

 "data": {

 "device\_id": "4658765"

 },

 "error": "succ"

}



**第3步 上传解析脚本**

 进入dtu-test产品的“脚本管理”页面，上传LUA解析脚本文件，此文档以“modbus.lua“举例说明，如下图所示：







**第4步**

设备上电，发送登录报文到OneNET接入服务器（见文档中心FAQ）进行鉴权。

登录报文格式如下：

\*PID#AuthCode#ParserName\*

**第5步 查看设备数据点**

设备登录成功后，OneNET接入服务会加载登录报文中指定的脚本，然后根据脚本内的定时下发数据设置，定时下发数据给设备。

进入dtu-test产品下的“设备管理》设备数据”，查看数据点情况，如下图所示：



其中，数据流名称是在解析脚本中指定的。

# LUA脚本编写说明

* 1. **设置定时下发设备的数据**

用户需实现Lua函数device\_timer\_init(dev)，以完成数据定时下发设备的设置，device\_timer\_init()无返回值。

dev为一个user\_data类型的值，提供了以下几个函数：

1. dev:add(interval,name,data)

添加定时下发数据。

 @param interval number 数据下发的时间间隔（秒）

 name string 名称（须保证唯一性）

 data string 数据（二进制数据），使用lua转义字符串

 @return 成功返回true，否则返回false

 @notice 定时数据下发的平均频率不超过1，及1/interval\_1+...+1/interval\_n<=1

 @example dev:add(10,”test”,”\1\1\0\150\0\37\253\29”)

1. dev:timeout(sec)

设置下发数据的设备响应超时时间（秒）。

 @param sec number 响应超时时间（秒）

 如果值为0，表示不检测设备响应超时

 @return 无

 @example dev:timeout(3)

1. dev:response()

设备响应成功。

 @param 无

 @return 无

 @example dev:response()

1. dev:send(data)

下发数据到设备。

 @param data string 数据（二进制数据），使用lua转义字符串

 @return 无

 @example dev:send("\2\2\0\150\0\37\206\89")

* 1. **解析设备上传数据**

用户需实现Lua函数device\_data\_analyze(dev)，以完成对设备上传数据的解析，device\_data\_analyze()有2个返回值size,json。

其中，size表示已解析设备上传数据的字节数，json表示解析后的数据点集合，格式如下：

[

 {

 “i” : ”dsname1”, // 数据流或数据流模板名称1

 “a” : 1234567890, // 毫秒级时间戳，距离（00:00:00 UTC, January 1, 1970）的毫秒

 // 如果值为0，表示使用当前时间

 “v” : 123 | ”123” | {…} // 布尔值、数值、字符串、json

 “b” : “0A0B0C0D…” // 二进制数据（16进制字符串），与v互斥，不同时存在

 “d” : xxx | “xxx” | {…} // 用于描述b（可选）；布尔值、数值、字符串、json

 “c” : “authcode1” // 用于标识数据点归属(设备AuthCode)

 // 如果值为“”或不存在，表示数据点归属建立TCP连接的设备

 }

 …

 {

 “i” : ”dsnamen”, // 数据流或数据流模板名称n

 “a” : 1234567890, // 毫秒级时间戳，距离（00:00:00 UTC, January 1, 1970）的毫秒

 // 如果值为0，表示使用当前时间

 “v” : 123 | ”123” | {…} // 布尔值、数值、字符串、json

 “b” : “0A0B0C0D…” // 二进制数据（16进制字符串），与v互斥，不同时存在

 “d” : xxx | “xxx” | {…} // 用于描述b（可选）；布尔值、数值、字符串、json

 “c” : “authcode1” // 用于标识数据点归属(设备AuthCode，可选)

 // 如果值为“”或不存在，表示数据点归属建立TCP连接的设备

 }

]

dev为一个user\_data类型的值，提供了以下几个函数：

1. dev:add(interval,name,data)

添加定时下发数据。

 @param interval number 数据下发的时间间隔（秒）

 name string 名称（须保证唯一性）

 data string 数据（二进制数据），使用lua转义字符串

 @return 成功返回true，否则返回false

 @notice 定时数据下发的平均频率不超过1，及1/interval\_1+...+1/interval\_n<=1

 @example local ok = dev:add(10,”test”,”\1\1\0\150\0\37\253\29”)

1. dev:timeout(sec)

设置下发数据的设备响应超时时间（秒）。

 @param sec number 响应超时时间（秒）

 如果值为0，表示不检测设备响应超时

 @return 无

 @example dev:timeout(3)

1. dev:response()

设备响应成功。

 @param 无

 @return 无

 @example dev:response()

1. dev:send(data)

下发数据到设备。

 @param data string 数据（二进制数据），使用lua转义字符串

 @return 无

 @example dev:send("\2\2\0\150\0\37\206\89")

1. dev:size()

获取设备数据大小（字节数）。

 @param 无

 @return 返回设备数据大小（字节数）

 @example local sz = dev:size()

1. dev:byte(pos)

获取pos对应位置的设备数据（字节）。

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

 @return 成功返回设备数据（int），否则返回nil

 @example local data = dev:byte(1)

1. dev:bytes(pos,count)

获取从pos开始，count个设备数据。

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

 count number 指定的获取数据总数，取值范围[0,dev:size()+1-pos]

 @return 成功返回设备数据（string），否则返回nil

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

* 1. **Lua工具函数**
1. u2f(u)

将32为整数内存转换为浮点数内存，并返回浮点数值（不同于值转换）；其类似于C/C++的强制内存转换，例如：\*(float\*)(&u)。

@param u number

@return 成功返回浮点数值，否则返回nil

@example local f = utf(u)

1. time()

获取时间戳，距离（00:00:00 UTC, January 1, 1970）的毫秒数

@return 返回时间戳

@example local t = time()

1. year(t)

获取年，距离1900的年数

@param t number 时间戳，距离（00:00:00 UTC, January 1, 1970）的秒数

@return 返回年

@example local y = year(t)+1900

1. month(t)

获取月（0-11）

@param t number 时间戳，距离（00:00:00 UTC, January 1, 1970）的秒数

@return 返回月

@example local m = month(t)+1

1. day(t)

获取日（1-31）

@param t number 时间戳，距离（00:00:00 UTC, January 1, 1970）的秒数

@return 返回日

@example local d = day(t)

1. hour(t)

获取时（0-23）

@param t number 时间戳，距离（00:00:00 UTC, January 1, 1970）的秒数

@return 返回时

@example local h = hour(t)

1. minute(t)

获取分（0-59）

@param t number 时间戳，距离（00:00:00 UTC, January 1, 1970）的秒数

@return 返回分

@example local m = minute(t)

1. second(t)

获取秒（0-59）

@param t number 时间戳，距离（00:00:00 UTC, January 1, 1970）的秒数

@return 返回秒

@example local s = second(t)

1. to\_hex(s)

将bytes string转换为hex string。

@param s string bytes string

@return 返回hex string，类似”0A0B0C0D…”

@example local hex = to\_hex(s)

1. to\_str(o)

将lua对象序列化成字符串。

@param o boolean|number|string|table

@return 返回序列化string

@example local str = to\_str(o)

1. add\_val(t,i,a,v)

添加值数据点到table中。

@param t table

 i string 数据流或数据流模板名称

 a number   毫秒级时间戳，距离（00:00:00 UTC, January 1, 1970）的毫秒；

 如果值为0，表示使用当前时间

 v boolean|number|string|table 布尔值、数值、字符串、json

 c string 用于标识数据点归属(设备AuthCode,可选)

 如果值为“”或nil，表示数据点归属建立TCP连接的设备

@return 成功返回true，否则返回false

@example local ok = add\_val(t,”dsname”,0,100,”dev”)

1. add\_bin(t,i,a,b,d)

添加二进制数据点到table中。

@param t table

 i string 数据流或数据流模板名称

 a number 毫秒级时间戳，距离（00:00:00 UTC, January 1, 1970）的毫秒；

 如果值为0，表示使用当前时间

 b string 二进制数据（hex string），类似”0A0B0C0D…”

 d boolean|number|string|table 用于描述b（可选），布尔值、数值、字符串、json

 c string 用于标识数据点归属(设备AuthCode,可选)

 如果值为“”或nil，表示数据点归属建立TCP连接的设备

@return 成功返回true，否则返回false

@example local ok = add\_bin(t,”dsname”,0,”0A0B0C0D...”,{...},”dev”)

1. to\_json(t)

将table序列化成json字符串。

@param t table 通过add\_val、add\_bin构建起来的table

@return 返回序列化json字符串

@example local json = to\_json(t)