移远通信物联网设备管理平台设备接入应用指导

（Python 版本）

移远物联网平台

版本：2.9.0

日期：2021-11-26

状态：临时文件

上海移远通信技术股份有限公司始终以为客户提供最及时、最全面的服务为宗旨。如需任何帮助，请随时联系我司上海总部，联系方式如下：

上海移远通信技术股份有限公司
上海市闵行区田林路 1016 号科技绿洲 3 期（B 区）5 号楼　　邮编：200233
电话：+86 21 51086236　　邮箱：info@quectel.com

或联系我司当地办事处，详情请登录：http://www.quectel.com/cn/support/sales.htm。

如需技术支持或反馈我司技术文档中的问题，可随时登陆如下网址：
http://www.quectel.com/cn/support/technical.htm 或发送邮件至：support@quectel.com。

前言

上海移远通信技术股份有限公司提供该文档内容用以支持其客户的产品设计。客户须按照文档中提供的规范、参数来设计其产品。因未能遵守有关操作或设计规范而造成的损害，上海移远通信技术股份有限公司不承担任何责任。在未声明前，上海移远通信技术股份有限公司有权对该文档进行更新。

免责声明

上海移远通信技术股份有限公司尽力确保开发中功能的完整性、准确性、及时性或效用，但不排除上述功能错误或遗漏的可能。除非其他有效协议另有规定，否则上海移远通信技术股份有限公司对开发中功能的使用不做任何暗示或明示的保证。在适用法律允许的最大范围内，上海移远通信技术股份有限公司不对任何因使用开发中功能而遭受的损失或损害承担责任，无论此类损失或损害是否可以预见。

保密义务

除非上海移远通信技术股份有限公司特别授权，否则我司所提供文档和信息的接收方须对接收的文档和信息保密，不得将其用于除本项目的实施与开展以外的任何其他目的。未经上海移远通信技术股份有限公司书面同意，不得获取、使用或向第三方泄露我司所提供的文档和信息。对于任何违反保密义务、未经授权使用或以其他非法形式恶意使用所述文档和信息的违法侵权行为，上海移远通信技术股份有限公司有权追究法律责任。

版权申明

本文档版权属于上海移远通信技术股份有限公司，任何人未经我司允许而复制转载该文档将承担法律责任。

文档历史

修订记录

版本	日期	作者	变更表述
-	2021-06-28	Evan LI/jc.wu	文档创建
1.0.0	2021-07-12	Evan LI/ Jc.wu	临时版本
1.0.1	2021-07-13	Jc.wu	1、增加定位功能 API
2.7.2	2021-07-05	Jc.wu	增加定位功能
2.8.3	2021-11-01	Jc.wu	增加 HTTP OTA 服务
2.9.0	2021-11-26	laili.jia	增加子设备服务

目录

表格索引

图片索引

1 引言

本文档主要介绍如何通过Python连接应用移远通信无线模块的设备（以下简称“设备”）至移远通信物联网设备管理平台（以下简称“设备管理平台”）并实现数据交互。

备注

此文档适用于移远通信物联网设备管理平台 V2.3.0 及之后版本。

2 平台接入流程及数据交互

2.1. 平台接入流程

用户登录设备管理平台 https://iot-cloud.quectel.com/后，连接设备至设备管理平台的具体操作步骤如下：

1. 设备上电后，首先声明 **quecIot** 模块，并执行初始化接口。

```
import quecIot
""" 初始化 qucsdk """
quecIot.init()
```

2. 若未配置信息，执行如下接口设置产品信息和服务器信息（若信息已配置，此步骤可忽略）。

```
""" 配置服务器信息，可选参数，默认连接 MQTT 生产环境服务器 """
quecIot.setServer(1,"http://iot-south.quectel.com:2883")
""" 配置产品信息 """
quecIot.setProductinfo("p1116a","UHg1d********VG")
""" 配置 Lifetime，可选参数，MQTT 默认为 120 """
quecIot.setLifetime(120)
```

3. 使用如下接口配置事件回调接口并接入设备管理平台。

```
""" 注册事件回调函数 """
quecIot.setEventCB(self.eventCB)
""" 启动云平台连接 """
quecIot.setConnmode(1)
```

备注

事件回调接口用于获取设备连接状态、数据收发状态、OTA 等。具体可参考***第 44.1 章***。

2.2. 透传数据

2.2.1. 发送透传数据至设备管理平台

设备成功接入移远通信物联网设备管理平台后，用户可通过如下接口发送数据到平台。任何类型的数据均可采用透传模式。

```
""" 发送透传数据 """
quecIot.passTransSend(1,"pass data")
```

模块成功发送数据后将返回 **True** 并产生如下事件：
若 *mode* 为 1 或以上，模块将调用事件回调函数，并传入参数 **4,10200**；
若 *mode* 为 0，模块不产生事件。

2.2.2. 设备管理平台下发送透传数据至设备

1. 设备接入管理平台后，在设备管理平台“**设备管理**”页面，点击待接收数据的设备对应操作栏中的“**查看**”按钮，进入设备详情页面。

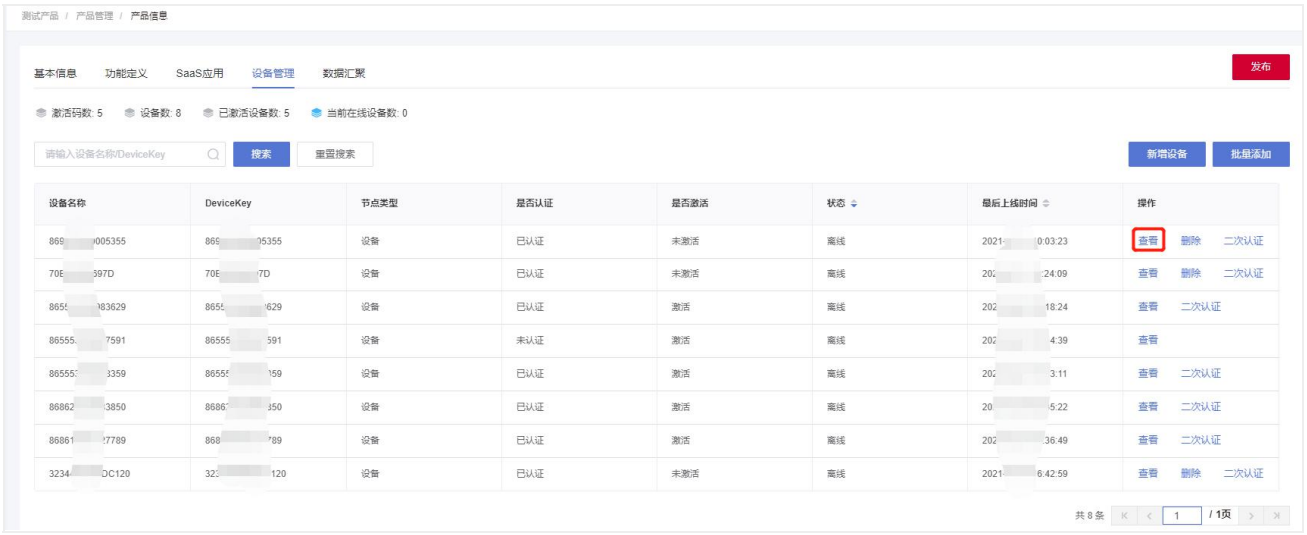

图 1：查看设备详情

2. 在“设备详情”页面，点击“**设备调试**”页签，打开数据调试发送窗口。

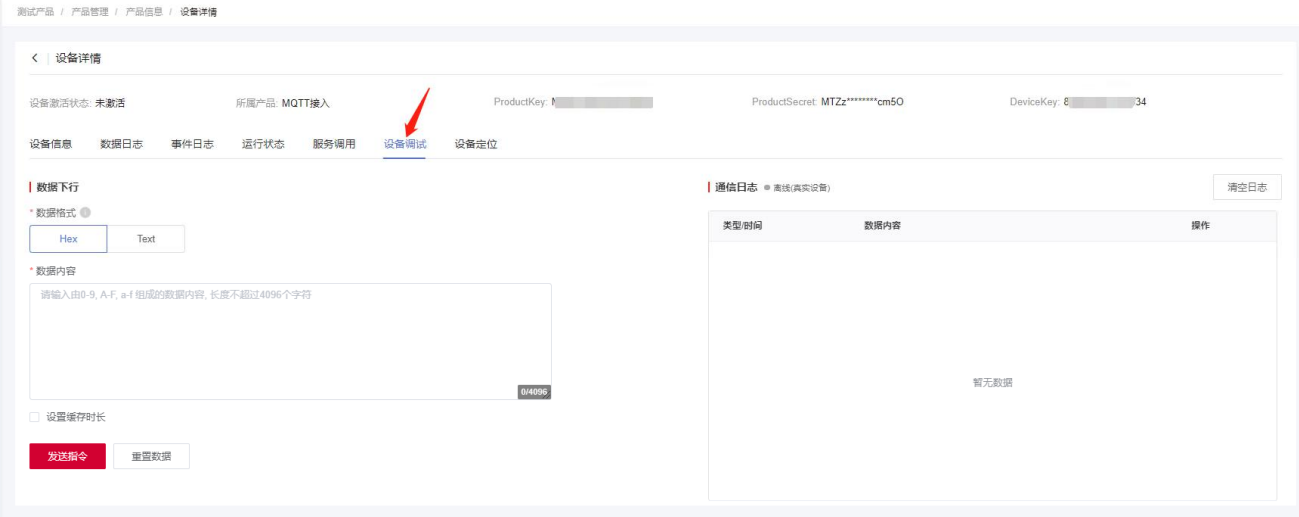

图 2：设备调试发送窗口

3. 选择数据格式并输入待发送的数据，如需，也可同时设置缓存的时长（数据如果下发失败，可尝试多次下发。超过设置的缓存时长，平台将不会下发此数据）；然后点击“**发送指令**”按钮完成数据发送。

4. 模块成功接收数据后，会通过注册事件回调函数通知应用层，用户只需要对事件类型进行判断并提取数据即可。详情可参考***第 44.1 章***。

2.3. 物模型数据

说明：物模型概述请参考：https://iot-cloud.quectel.com/help?id=3280

2.3.1. 发送物模型数据至设备管理平台

设备成功接入设备管理平台后，用户可通过 *quecIot.phymodelReport()*发送物模型数据到设备管理平台。物模型数据为 dict 类型数据，Key 是在平台创建物模型数据时生成的对应功能点 ID，value 是这个功能点 ID 对应的数据。如下图所示：

功能ID Key	功能类型	功能名称	标识符	数据类型	数据定义 value 取值范围	功能描述	操作
24	属性	bool	bool	BOOL	布尔值： true - 开 false - 关	-	查看
25	属性	整数	int	INT	取值范围：-100000 ~ 100000	-	查看
26	属性	浮点数	float	FLOAT	取值范围：-1111.1 ~ 1111.1	-	查看
27	属性	文本	TEXT	TEXT	数据长度：120	-	查看
28	属性	日期	date	DATE	-	-	查看
29	属性	结构体	struct	STRUCT	-	-	查看
33	属性	数组	array	ARRAY	-	-	查看

图 3：物模型数据信息定义

```
""" 发送 bool 型数据"""
quecIot.phymodelReport(1, {24: True})
""" 发送数值 """
""" 整数 """
quecIot.phymodelReport(1, {25: 123})
""" 浮点数 """
quecIot.phymodelReport(1, {26: 123.123})
""" 发送 array """
quecIot.phymodelReport(1, {33: [1, 2, 3]})
""" 发送结构体 """
quecIot.phymodelReport(1, {29: {30: False, 31: 10,32: "string"}})
```

成功发送数据后将返回 **True** 并产生如下事件：
若 *mode* 为 1 或以上，模块将调用事件回调函数，并传入参数 **4,10210**；
若 *mode* 为 0，模块不产生事件。

2.3.2. 设备管理平台下发物模型数据至设备

1. 在“设备管理”页面，点击需要接收数据的设备对应的操作栏中的“**查看**”按钮进入“设备详情”页面。

2. 在“设备详情”页面，点击“**设备调试**”页签，打开数据调试发送窗口。

3. 当接收到云平台下发的物模型数据时，会通过事件回调函数通知应用层，用户只需要对事件类型进行判断并提取数据即可。详情可参考***第 44.1 章***。

4. 注册回调函数后，通过 Secure-CRT 工具在 REPL 模式下会收到数据，该事件回调函数返回的 dict 类型数据，Key 是在平台创建物模型数据时生成的对应功能点 ID，value 是这个功能点 ID 对应的数据。

```
5,10210,{24, Ture}
5,10210,{25, -100000}
```

2.3.3. 设备管理平台读取物模型数据

当接收到云平台请求物模型数据时，会通过事件回调函数进行通知，用户只需要对事件类型进行判断并提取数据即可。处理完成后发送物模型应答数据至设备管理平台。

注册回调函数后，通过 Secure-CRT 工具在 REPL 模式下会收到数据，该事件回调函数返回的是 list 类型数据，第一个元素是 pkgid（即请求包 ID），第二个元素是 id list（即需要请求读取的物模型功能点 ID 列表）。

```
5,10211,[56180, [2, 3]]
```

设备收到云平台请求命令时，可通过 *quecIot.phymodelAck()*应答云平台请求功能点 ID 的值。

```
""" 应答数据"""
quecIot.phymodelAck(1, 56180, {2: True,3: 123})
```

2.4. OTA 升级

2.4.1 MQTTOTA

2.4.1.1 FOTA 升级交互流程

1. 使用 *quecIot.devInfoReport()*上报模块信息。

2. 如果平台有升级计划，则会下发升级任务。此时将会执行回调。参数如下：
7,10700，"<componentNo>","<sourceVersion>","<targetVersion>",<batteryLimit>,<minSignalIntensity>,<useSpace>

3. 收到升级任务通知后，可通过 *quecIot.otaAction(1)*确认升级。

4. 开始下载固件，会触发事件回调函数。参数如下：
7,10701
7,10703
7,10704

5. 固件下载成功后，模组会自动重启更新。

6. 更新完成后模组自动启动，然后连接平台。此时会触发事件回调函数。
若升级成功，参数为：**7,10705**
若升级失败，参数为：**7,10706**

备注

有关事件回调函数的参数信息，可参考***第 44.1 章***。

2.4.2 HTTPOTA

2.4.2.1 FOTA 升级交互流程

1. 使用 *quecIot.*setHttpOtaUp()查询平台是否有升级计划
2. 如果平台有升级计划，则会下发计划。此时将会执行回调。

```
void (*eventCB)(1, 0,value,valLen);
```

3. 下载中

```
void (*eventCB)(3, 0,NULL,0);
```

4. 下载成功

```
void (*eventCB)(4, 0,NULL,0);
```

5. 下载失败

```
void (*eventCB)(5, 0,NULL,0);
```

6. 开始更新

```
void (*eventCB)(6, 0,NULL,0);
```

7. 更新成功

```
void (*eventCB)(7, 0,NULL,0);
```

8. 更新失败

```
void (*eventCB)(8, 0,NULL,0);
```

3 数据结构及 API 介绍

3.1. 模块

使用此功能需要加载 **quecIot** 模块。

3.2. 函数概览

表 1：函数概览

函数	说明
quecIot.init()	初始化 QuecThing 功能
quecIot.setConnmode()	配置云平台连接模式
quecIot.getConnmode()	获取 SDK 当前配置的云平台连接模式
quecIot.setPdpContextId()	配置 SDK 连接云平台使用的 PDP 场景 ID
quecIot.getPdpContextId()	获取 SDK 连接云平台使用的 PDP 场景 ID
quecIot.setServer()	配置通信协议和引导/认证服务器
quecIot.getServer()	获取通信协议和引导/认证服务器
quecIot.setProductinfo()	配置云平台产品信息
quecIot.getProductinfo()	获取云平台产品信息
quecIot.setLifetime()	配置模块 Lifetime
quecIot.getLifetime()	获取云平台 Lifetime
quecIot.setMcuVersion()	配置 MCU 编号和其对应的版本号
quecIot.getMcuVersion()	获取 MCU 编号和其对应的版本号

quecIot.getWorkState()	获取云平台连接状态
quecIot.setEventCB()	设置云平台事件回调处理函数
quecIot.passTransSend()	向云平台发送透传数据
quecIot.phymodelReport()	向云平台发送物模型数据
quecIot.phymodelAck()	向云平台应答物模型数据
quecIot.otaAction()	接收到云平台推送升级任务时，进行 OTA 确认
quecIot.mcuFWDataRead()	读取保存在模块的固件数据
quecIot.statusReport()	上报指定内容状态
quecIot.devInfoReport()	上报指定内容信息
quecIot.locSupList()	该函数用于获取模组当前的内置定位支持类型。
quecIot.locDataGet()	该函数用于获取模组当前的内置定位数据。
quecIot.locReportInside()	该函数用于上报模组当前的内置定位数据。
quecIot.locReportOutside()	该函数用于上报模组外置定位数据。
quecIot.setHttpOtaEventCb()	设置 HTTP OTA 平台事件回调处理函数
quecIot.setHttpOtaProductInfo()	配置产品信息
quecIot.getHttpOtaProductInfo()	获取产品信息
quecIot.setHttpOtaTls()	配置通信加密方式
quecIot.getHttpOtaTls()	获取通信加密方式
quecIot.setHttpOtaServer()	配置服务器地址
quecIot.getHttpOtaServer()	获取服务器地址
quecIot.setHttpOtaUp()	配置参数以及查询平台是否存在升级计划
quecIot.getHttpOtaUp()	查询配置的参数
quecIot.subDevSetEventCB()	设置子设备云平台事件回调处理函数
quecIot.subDevConn()	发起子设备连接平台操作
quecIot.subDevDisconn()	发起子设备断开平台操作

quecIot.subDevDeauth()	发起子设备从平台注销操作
quecIot.subDevPassTransSend()	子设备发送透传数据
quecIot.subDevPhymodelReport()	子设备发送物模型数据
quecIot.subDevPhymodelAck()	子设备回复物模型查询
quecIot.subDevHTB()	子设备刷新心跳

3.3. 相关 API

3.3.1. queclot.init

该函数用于初始化配置，启动 QuecThing 服务。只能在使用 QuecThing 之前初始化一次。

- **函数原型**

```
quecIot.init()
```

- **参数**

无

- **返回值**

True 函数执行成功
False 函数执行失败

3.3.1. queclot.setConnmode

该函数用于配置连接云平台的模式。可在 QuecThing 启动前和运行中进行配置。

- **函数原型**

```
quecIot.setConnmode(mode)
```

- **参数**

mode：
[In] 整型。连接模式
0 退出或不连接云平台
1 发起连接云平台

- **返回值**

True 函数执行成功
False 函数执行失败

3.3.2. queclot.getConnmode

该函数用于获取 SDK 当前配置的云平台连接模式。

- **函数原型**

```
quecIot.getConnmode()
```

- **参数**

无

- **返回值**

整型。连接模式
0 未发起连接
1 发起连接状态

3.3.3. queclot.getWorkState

该函数用于获取云平台当前连接状态。

- **函数原型**

```
quecIot.getWorkState()
```

- **参数**

无。

- **返回值**

整型。状态编号。
0 未初始化 .
1 已初始化
2 正在认证
3 认证成功
4 认证失败
5 正在注册

6 注册成功，等待订阅
7 注册失败
8 已订阅，数据可发送
9 订阅失败
10 正在注销
11 注销成功
12 注销失败

3.3.4. queclot.setEventCB

该函数用于设置云平台事件回调处理函数。

- **函数原型**

```
quecIot.setEventCB(eventCb)
```

- **参数**

eventCb：
[In] 回调函数。**quecIot** 模块事件回调函数。

- **返回值**

True 函数执行成功
False 函数执行失败

3.3.4.1. eventCb

queclot 模块事件回调函数。

- **函数原型**

```
eventCb(data_list)
```

- **参数**

data_list：
[In] 形式参数，序列类型。

- **返回值**

data_list[0] 整型。Event 事件标识符。详情请参考*第 44.1 章*。
data_list[1] 整型。Errcode，错误码。详情请参考*第 44.1 章*。

data_list[2]　Value，附带数据。详情请参考*第 44.1 章*。

3.4. 产品配置相关 API

3.4.1. queclot.setProductinfo

该函数用于配置云平台产品信息，从云平台创建产品时获取。有关产品信息的获取可参考*文档 [1]*。

- **函数原型**

```
quecIot.setProductinfo(product_key,product_secret)
```

- **参数**

product_key：
[In] 字符串类型。创建产品生成的 ProductKey。
product_secret：
[In] 字符串类型。创建产品生成的 ProductSecret。

- **返回值**

True 函数执行成功
False　函数执行失败

3.4.2. queclot.getProductinfo

该函数用于获取云平台产品信息。

- **函数原型**

```
quecIot.getProductinfo()
```

- **参数**

无。

- **返回值**

index 0　字符串类型。创建产品生成的 ProductKey。
index 1　字符串类型。创建产品生成的 ProductSecret。
index 2　字符串类型。平台协议版本号。

3.4.3. queclot.setServer

该函数用于配置云平台通信协议和引导/认证服务器；

- **函数原型**

```
quecIot.setServer(type,server)
```

- **参数**

type：
[In] 整型。协议类型。可选参数，不配置时使用默认值 1。
0 LwM2M（暂不支持）
1 MQTT

server：
[In] 字符串类型。认证服务器地址。默认连接 MQTT 生产环境服务器：iot-south.quectel.com:2883。
可选参数，不配置时使用默认值。

- **返回值**

True 函数执行成功
False 函数执行失败

3.4.4. queclot.getServer

该函数用于获取云平台通信协议和引导/认证服务器。

- **函数原型**

```
quecIot.getServer()
```

- **参数**

无。

- **返回值**

index 0 整型。协议类型。
index 1 字符串类型。MQTT 认证或 LwM2M 引导服务器地址。

3.4.5. queclot.setLifetime

该函数用于配置模块 Lifetime。

- **函数原型**

```
quecIot.setLifetime(life_time)
```

- **参数**

life_time：
[In] 整型。心跳周期，单位：秒。可选参数，不配置时使用默认值 120。

- **返回值**

True 函数执行成功
False 函数执行失败

备注

使用 MQTT 协议时，范围为 1~65535。使用 LwM2M 协议时，范围为 1~0xFFFFFFFF。接口不会检查输入值的合法性。

3.4.6. queclot.getLifetime

该函数用于获取云平台 Lifetime。

- **函数原型**

```
quecIot.getLifetime()
```

- **参数**

无。

- **返回值**

整型。心跳周期，单位：秒。

3.4.7. queclot.setPdpContextId

该函数用于配置 SDK 连接云平台使用的 PDP 场景 ID。

- **函数原型**

```
quecIot.setPdpContextId(contextID)
```

- **参数**

contextID：
[In] 整型。场景 ID。可选参数，不配置时使用默认值 1。

- **返回值**

True 函数执行成功
False 函数执行失败

备注

当前场景 ID 仅能配置 1。

3.4.8. queclot.getPdpContextId

该函数用于获取 SDK 连接云平台使用的 PDP 场景 ID。

- **函数原型**

```
quecIot.getPdpContextId()
```

- **参数**

无。

- **返回值**

整型。场景 ID。当查询失败时，返回 None。

3.4.9. queclot.setSessionFlag

该函数用于配置与云平台通信的数据是否采用 session 加密。

- **函数原型**

```
quecIot.setSessionFlag(sessionFlag)
```

- **参数**

sessionFlag：
[In] session 启用标识。不配置时使用默认为 FALSE，不启用。

- **返回值**

True　　函数执行成功
False　　函数执行失败

3.4.10. queclot.getSessionFlag

该函数用于获取与云平台通信的数据是否采用 session 加密。

- **函数原型**

```
quecIot.getSessionFlag()
```

- **参数**

无。

- **返回值**

True　　启用 session 机密
False　　关闭 session 机密

3.4.11. queclot.getSoftVersion

该函数用于获取软件版本标识。

- **函数原型**

```
quecIot.getSoftVersion()
```

- **参数**

无。

- **返回值**

字符串，当前固件版本号。

3.4.12. queclot.setMcuVersion

该函数用于配置云平台 MCU 编号和其对应的版本号，需要外挂 MCU 时配置。

- **函数原型**

```
quecIot.setMcuVersion(compno,version)
```

- **参数**

compno：
[In] 字符串类型。MCU 标识，最大长度 32 字节。

version：
[In] 字符串类型。MCU 编号对应的版本，最大长度 64 字节。

- **返回值**

True 函数执行成功
False 函数执行失败

3.4.13. queclot.getMcuVersion

该函数用于获取云平台 MCU 编号和其对应的版本号。

- **函数原型**

```
quecIot.getMcuVersion()
```

- **参数**

无。

- **返回值**

序列。元素是元组。
compno 字符串类型。MCU 组件标识。
version 字符串类型。MCU 版本号。

3.4.14. queclot.setDkDs

该函数用于修改连接云平台的默认 DK 和 DS。

- **函数原型**

```
quecIot.setDkDs(dk,ds)
```

- **参数**

dk：
[In] device key。字符串类型，最大限制字节为 16 字节，为空且当前非默认 DK 时将清空配置的 DK/DS，恢复使用默认 DK，并需要重新认证。

ds：
[In] device secret。字符串类型，最大限制字节为 32 字节。若设置 dk 非空且非默认值时，将允许设置 DS。

- **返回值**

True　　函数执行成功
False　　函数执行失败

3.4.15. queclot.getDkDs

该函数用于获取连接云平台的 DK 和 DS。

- **函数原型**

```
quecIot.getDkDs()
```

- **参数**

无。

- **返回值**

序列。元素是元组。
dk　　字符串类型。设置的 device key。
ds　　字符串类型。设备密钥。

备注

仅在 setDkDs 设置了非模组默认的 DK 后才允许查询。

3.5. 发送数据相关 API

3.5.1. queclot.passTransSend

该函数用于向云平台发送透传数据。

- **函数原型**

```
quecIot.passTransSend(mode,data)
```

- **参数**

mode：
[In] 整型。发送模式。
若协议为 MQTT：
0 QoS=0，仅发送一次。
1 QoS=1，最少发送一次。
2 QoS=2，最多发送一次。

若协议为 LwM2M：
0 发送 NON 消息。
1 发送 NON 消息并携带 RELEASE 标记。
2 发送 CON 消息。
3 发送 CON 消息并携带 RELEASE_AFTER_REPLY 标记。

data：
[In] bytes 类型。待发送数据。

- **返回值**

True 函数执行成功
False 函数执行失败

3.5.2. queclot.phymodelReport

该函数用于向云平台发送物模型数据。

- **函数原型**

```
quecIot.phymodelReport(mode,data)
```

- **参数**

mode：
[In] 整型。发送模式。
若协议为 MQTT：
0 QoS=0，仅发送一次。
1 QoS=1，最少发送一次。
2 QoS=2，最多发送一次。

若协议为 LwM2M：
0 发送 NON 消息。

1　发送 NON 消息并携带 RELEASE 标记。
2　发送 CON 消息。
3　发送 CON 消息并携带 RELEASE_AFTER_REPLY 标记。

data：
[In] dict 类型。待发送物模型数据。使用方法详见***第 4.2.3 章***。

- **返回值**

True 函数执行成功
False　函数执行失败

3.5.3. queclot.phymodelAck

该函数用于向云平台应答物模型数据，在接收到云平台读取物模型数据时进行应答。

- **函数原型**

```
quecIot.phymodelAck(mode,pkgid,data)
```

- **参数**

mode：
[In] 整型。发送模式。
若协议为 MQTT：
0　QoS=0，仅发送一次。
1　QoS=1，最少发送一次。
2　QoS=2，最多发送一次。

若协议为 LwM2M：
0　发送 NON 消息。
1　发送 NON 消息并携带 RELEASE 标记。
2　发送 CON 消息。
3　发送 CON 消息并携带 RELEASE_AFTER_REPLY 标记。

pkgid：
[In] 整型。请求包 ID。

data：
[In] dict 类型。待发送物模型数据。使用方法详见***第 4.2.3 章***。

- **返回值**

True　函数执行成功

False　　函数执行失败

3.5.4. queclot.statusReport

该函数用于上报指定内容状态。

- **函数原型**

```
quecIot.statusReport(data)
```

- **参数**

data：
[In] 序列，元素是整型。上报内容列表：
1　电量百分比
2　电压（单位伏特）
3　信号强度（RSSI）
4　剩余空间（单位字节）
5　参考信号接收功率（RSRP）
6　LTE 参考信号接收质量（RSRQ）
7　信号与干扰噪声比（SNR）

- **返回值**

True　　函数执行成功
False　　函数执行失败

3.5.5. queclot.getDevStatus

该函数用于获取设备状态。

- **函数原型**

```
quecIot.getDevStatus(id_list)
```

- **参数**

id_list：
[In] 序列，元素是整型。上报内容列表：
1　电量百分比
2　电压（单位伏特）
3　信号强度（RSSI）
4　剩余空间（单位字节）
5　参考信号接收功率（RSRP）

6 LTE 参考信号接收质量（RSRQ）
7 信号与干扰噪声比（SNR）

- **返回值**

字典：
Key：查询的内容 ID
*Value:*查询的返回值

3.5.6. queclot.devInfoReport

该函数用于上报指定内容信息。

- **函数原型**

```
quecIot.devInfoReport(data)
```

- **参数**

data：
[In] 序列，元素是整型。上报内容列表：
1 模块型号
2 模块版本
3 MCU 版本
4 小区 ID（Cell ID）
5 集成电路卡识别码（ICCID）
6 移动国家代码（MCC）
7 移动网络代码（MNC）
8 位置区代码（LAC）
9 电话号码
10 国际移动用户识别码（IMSI）
11 IoT SDK 版本号
12 定位模式支持返回值

- **返回值**

True 函数执行成功
False 函数执行失败

3.5.7. queclot.getDevInfo

该函数用于获取模块信息。

- **函数原型**

```
quecIot.getDevInfo(id_list)
```

- **参数**

id_list：
[In] 序列，元素是整型。上报内容列表：
1 模块型号
2 模块版本
3 MCU 版本
4 小区 ID（Cell ID）
5 集成电路卡识别码（ICCID）
6 移动国家代码（MCC）
7 移动网络代码（MNC）
8 位置区代码（LAC）
9 电话号码
10 国际移动用户识别码（IMSI）
11 IoT SDK 版本号
12 定位模式支持返回值

- **返回值**

字典：
Key：查询的内容 ID
*Value:*查询的返回值

3.6. OTA 相关 API

3.6.1. queclot.otaRequest

该函数用于接收到云平台推送升级任务时，需要进行 OTA 确认。

- **函数原型**

```
quecIot.otaRequest(mp_mode)
```

- **参数**

mp_mode：
[In] 额外信息。

- **返回值**

True 函数执行成功
False 函数执行失败

额外信息定义如下：

- **参数**

参数	描述
0	无额外信息请求
1	额外请求 SHA256 信息

3.6.2. queclot.otaAction

该函数用于接收到云平台推送升级任务时，进行 OTA 确认。

- **函数原型**

```
quecIot.otaAction(action)
```

- **参数**

action：
[In] 整型。OTA 确认升级行为。
0 拒绝升级
1 确认升级
2 MCU 通知下载下一块数据
3 MCU 上报更新状态

- **返回值**

True 函数执行成功
False 函数执行失败

3.6.3. queclot.mcuFWDataRead

该函数用于读取保存在模块的固件数据。

- **函数原型**

```
quecIot.mcuFWDataRead(addr,len)
```

- **参数**

addr：
[In] 整型。读取数据开始位置。

len：

[In] 整型。读取长度。

- **返回值**

bytes 类型。读取的固件数据。当读取失败时，返回 None。

3.7. GNSS&LBS 定位相关 API

3.7.1. queclot.getLocSupList

该函数用于获取模组当前的内置定位支持类型。

- **函数原型**

```
quecIot.getLocSupList()
```

- **参数**

无

- **返回值**

list 类型。元素是字符串类型。模组内置支持的定位类型。详情参考**表 2。**

表 2：定位类型

定位类型	说明
NONE	关闭所有类型
AUTO	更新定位信号自动选择上报类型
LBS	基站定位
GGA	任意卫星系统的 GGA
RMC	任意卫星系统的 RMC
GSV	任意卫星系统的 GSV
GSA	任意卫星系统的 GSA
VTG	任意卫星系统的 VTG
GPGGA	GPS 卫星系统的 GGA

GPRMC	GPS 卫星系统的 RMC
GPGSV	GPS 卫星系统的 GSV
GPGSA	GPS 卫星系统的 GSA
GPVTG	GPS 卫星系统的 VTG
BDGGA	北斗卫星系统的 GGA
BDRMC	北斗卫星系统的 RMC
BDGSV	北斗卫星系统的 GSV
BDGSA	北斗卫星系统的 GSA
BDVTG	北斗卫星系统的 VTG
GLGGA	GLONASS 卫星系统的 GGA
GLRMC	GLONASS 卫星系统的 RMC
GLGSV	GLONASS 卫星系统的 GSV
GLGSA	GLONASS 卫星系统的 GSA
GLVTG	GLONASS 卫星系统的 VTG
GAGGA	Galileo 卫星系统的 GGA
GARMC	Galileo 卫星系统的 RMC
GAGSV	Galileo 卫星系统的 GSV
GAGSA	Galileo 卫星系统的 GSA
GAVTG	Galileo 卫星系统的 VTG
GNGGA	全球卫星导航系统的 GGA
GNRMC	全球卫星导航系统的 RMC
GNGSV	全球卫星导航系统的 GSV
GNGSA	全球卫星导航系统的 GSA
GNVTG	全球卫星导航系统的 VTG

3.7.2. queclot.getLocData

该函数用于获取模组当前的内置定位数据。

- **函数原型**

```
quecIot.getLocData(types)
```

- **参数**

types：
[In] list 类型。待获取定位类型列表。详情参考**表 2**。

- **返回值**

list 类型。返回获取到的指定定位类型 NMEA 数据。

3.7.3. queclot.locReportInside

该函数用于上报模组当前的内置定位数据。

- **函数原型**

```
quecIot.locReportInside(types)
```

- **参数**

types：
[In] list 类型。待上报的内置定位类型列表。详情参考**表 2**。

- **返回值**

True　　函数执行成功
False　　函数执行失败

3.7.4. queclot.locReportOutside

该函数用于上报模组外置定位数据。

- **函数原型**

```
quecIot.locReportOutside(nmea_data)
```

- **参数**

nmea_data：
[In] list 类型。待上报的外置定位 NMEA 数据列表。

- **返回值**

True　　函数执行成功
False　　函数执行失败

3.8. HTTP OTA 相关 API

3.8.1. queclot.setHttpOtaEventCb

该函数用于注册事件回调函数，后续 HTTP OTA 运行过程中产生的事件均通过该回调函数进行通知。

- **函数原型**

```
quecIot.setHttpOtaEventCb(eventCb)
```

- **参数**

eventCb：
[In] HTTP OTA 事件回调函数。详情请参考***第 4.2 章***。

- **返回值**

无。

3.8.1.1. eventCb

HTTP OTA 服务事件回调函数。

- **函数原型**

```
eventCb(data_list)
```

- **参数**

data_list：
[In] 形式参数，序列类型。

- **返回值**

data_list[0] 整型。Event 事件标识符。详情请参考*第 4.2 章*。
data_list[1] 整型。Errcode，错误码。详情请参考*第 4.2 章*。
data_list[2] Value，附带数据。详情请参考*第 4.2 章*。

3.8.2. queclot.setHttpOtaProductInfo

该函数用于配置产品信息。

- **函数原型**

```
quecIot.setHttpOtaProductInfo(product_key, product_secret)
```

- **参数**

*product*_key：
[In] 创建产品生成的 ProductKey。

*product*_secret：
[In] 创建产品生成的 ProductSecret。

- **返回值**

True 函数执行成功
False 函数执行失败

3.8.3. queclot.getHttpOtaProductInfo

该函数用于获取产品信息。

- **函数原型**

```
quecIot.getHttpOtaProductInfo()
```

- **参数**

无。

- **返回值**

index 0 字符串类型。创建产品生成的 ProductKey。
index 1 字符串类型。创建产品生成的 ProductSecret。

3.8.4. queclot.setHttpOtaTls

该函数用于配置通信加密方式。

- **函数原型**

```
quecIot.setHttpOtaTls(mp_tls)
```

- **参数**

mp_tls：
[In] 通信加密方式。

- **返回值**

True 函数执行成功
False 函数执行失败

通信加密方式定义如下：

- **参数**

参数	描述
0	不加密
1	加密

3.8.5. queclot.getHttpOtaTls

该函数用于获取通信加密方式。

- **函数原型**

```
quecIot.getHttpOtaTls()
```

- **参数**

无

- **返回值**

True 使用加密方式
False 使用非加密方式

3.8.6. queclot.setHttpOtaServer

该函数用于配置服务器地址。

- **函数原型**

```
quecIot.setHttpOtaServer(mp_server)
```

- **参数**

mp_server：
[In] 服务器地址：ip:port。

- **返回值**

True 函数执行成功
False 函数执行失败

3.8.7. queclot.getHttpOtaServer

该函数用于获取服务器地址。

- **函数原型**

```
quecIot.getHttpOtaServer()
```

- **参数**

无。

- **返回值**

字符串类型。服务器地址：ip:port。

3.8.8. queclot.setHttpOtaUp

该函数用于配置参数以及查询平台是否存在升级计划。

- **函数原型**

```
quecIot.setHttpOtaUp(mp_battery,mp_upmode,mp_url)
```

- **参数**

mp_battery：

[In] 电量。

mp_upmode：
[In] 升级模式。

mp_url：
[In] 服务器信息：ip:port。

- **返回值**

True 函数执行成功
False 函数执行失败

3.8.9. queclot.getHttpOtaUp

该函数用于查询配置的参数。

- **函数原型**

```
quecIot.getHttpOtaUp()
```

- **参数**

无

- **返回值**

index 0 字符串类型。电量。
index 1 字符串类型。升级模式。
index 2 字符串类型。服务器信息：ip:port。

3.9. 子设备相关 API

3.9.1. queclot.subDevSetEventCB

该函数用于设置子设备事件回调处理函数。

- **函数原型**

```
quecIot.subDevSetEventCB(event_sub_dev_urc_cb)
```

- **参数**

event_sub_dev_urc_cb:
[In] 回调函数。**quecIot** 模块事件回调函数。

- **返回值**

True 函数执行成功
False 函数执行失败

3.9.1.1. event_sub_dev_urc_cb

queclot 模块事件回调函数。

- **函数原型**

```
event_sub_dev_urc_cb(data_list)
```

- **参数**

data_list:
[In] 形式参数，序列类型。

- **返回值**

data_list[0] 字符串类型。创建子设备产品生成的 ProductKey。
data_list[1] 字符串类型。子设备唯一标识符。
data_list[2] 整型。Event 事件标识符。详情请参考***第 44.1 章***。
data_list[3] 整型。Errcode，错误码。详情请参考***第 44.1 章***。
data_list[4] Value，附带数据。详情请参考***第 44.1 章***。

3.9.2. queclot.subDevConn

该函数用于子设备连接网络；认证到平台时不需要传输 device_descret。登陆平台时需要传输 device_descret。

- **函数原型**

```
quecIot.subDevConn(product_key, product_descret, device_key,device_descret, sessopm_type,keyalive)
```

- **参数**

product_key:
[In] 字符串类型；创建子设备产品生成的 ProductKey。

product_descret:
[In] 字符串类型；创建子设备产品生成的 ProductDecret。

device_key：
[In] 字符串类型；子设备唯一标识符。

device_descret：
[In] 字符串类型；子设备认证成功后返回的设备秘钥。

session_type：
[In] 字符串类型；子设备加密方式。

keeyalive：
[In] 字符串类型；子设备与网关间心跳间隔

- **返回值**

True 函数执行成功
False 函数执行失败

3.9.3. queclot.subDevDisconn

该函数用于子设备登出平台。

- **函数原型**

```
quecIot.subDevDisconn(product_key, device_key)
```

- **参数**

product_key：
[In] 字符串类型。创建子设备产品生成的 ProductKey。

device_key：
[In] 字符串类型。子设备唯一标识符。

- **返回值**

True 函数执行成功
False 函数执行失败

3.9.4. queclot.subDevDeauth

该函数用于子设备从平台注销。

- **函数原型**

```
quecIot.subDevDeauth(product_key, product_descret, device_key,device_descret)
```

- **参数**

product_key:
[In] 字符串类型；创建子设备产品生成的 ProductKey。

product_descret:
[In] 字符串类型；创建子设备产品生成的 ProductDecret。

device_key:
[In] 字符串类型；子设备唯一标识符。

device_descret:
[In] 字符串类型；子设备认证成功后返回的设备秘钥。

- **返回值**

True 函数执行成功
False 函数执行失败

3.9.5. queclot.subDevPassTransSend

该函数用于子设备向平台发送透传数据。

- **函数原型**

```
quecIot.subDevPassTransSend(product_key, device_key, mp_data)
```

- **参数**

product_key:
[In] 字符串类型。创建子设备产品生成的 ProductKey。

device_key:
[In] 字符串类型。子设备唯一标识符。

mp_data:
[In] bytes 类型。待发送数据。

- **返回值**

True 函数执行成功
False 函数执行失败

3.9.6. queclot.subDevTslReport

该函数用于子设备向平台发送物模型数据。

- **函数原型**

```
quecIot.subDevTslReport(product_key, device_key, mp_data)
```

- **参数**

product_key:
[In] 字符串类型。创建子设备产品生成的 ProductKey。

device_key:
[In] 字符串类型。子设备唯一标识符。

mp_data:
[In] dict 类型。待发送物模型数据。使用方法详见***第 4.2.3 章***。

- **返回值**

True 函数执行成功
False 函数执行失败

3.9.7. queclot.subDevTslAck

该函数用于子设备回复平台查询物模型数据。

- **函数原型**

```
quecIot.subDevTslAck(product_key, device_key, mp_pkgid, mp_data)
```

- **参数**

product_key:
[In] 字符串类型。创建子设备产品生成的 ProductKey。

device_key:
[In] 字符串类型。子设备唯一标识符。

mp_pkgid:
[In] 整型。请求包 ID。

mp_data:

[In] dict 类型。待发送物模型数据。使用方法详见*第 4.2.3 章*。

- **返回值**

True 函数执行成功
False 函数执行失败

3.9.8. queclot.subDevHTB

该函数用于子设备刷新心跳时间。

- **函数原型**

```
quecIot.subDevHTB(product_key, device_key)
```

- **参数**

product_key:
[In] 字符串类型。创建子设备产品生成的 ProductKey。

device_key:
[In] 字符串类型。子设备唯一标识符。

- **返回值**

True 函数执行成功
False 函数执行失败

3.10. Modbus 相关 API

3.10.1. queclot.MBInit

该函数用于初始化 modbus 组件，需要注册可用于烧写配置文件的串口列表，以及注册串口发送函数和初始化回调函数。

- **函数原型**

```
quecIot.MBInit(uartList, sendFunc, initCb);
```

- **参数**

uartList:
[In] 可用于串口烧写配置文件的串口列表，需要自行打开对应串口。

sendFunc：
[In] 串口发送函数。详情请参考***第 3.10.1.1.章***。

initCb：
[In] 初始化回调函数。详情请参考***第 3.10.1.2.章***。

- **返回值**

TRUE　modbus 组件初始化失败
FALSE　modbus 组件初始化成功

3.10.1.1. sendFunc

modbus 组件需要往串口发送数据时将调用该函数，用户需要在该函数中实现串口发送能力。

- **函数原型**

```
sendFunc(data_list)
```

- **参数**

data_list：
[In] 形式参数，序列类型。

- **返回值**

data_list[0]　整型。端口号。
data_list[1]　字节数组。需要发送的数据内容。

3.10.1.2. initCb

modbus 组件初始化完成后，将配置文件信息回调到该函数中，需要在该函数中实现串口和设备连接 DMP 平台等操作。

- **函数原型**

```
initCb(data_dict)
```

- **参数**

data_dict：
[In] 形式参数，字典类型。

- **返回值**

data_dict["product"]	字典类型。
product_dict["pk"]	字符串类型。创建产品生成的 ProductKey。
product_dict["ps"]	字符串类型。创建产品生成的 ProductSecret。
data_dict["uart"]	序列类型。
uart_list[0]	整形。端口号。
uart_list[1]	整形。波特率。
uart_list[2]	整形。数据位。详情请参考***第 3.10.1.2.1. 章***。
uart_list[3]	整形。校验位。详情请参考***第 3.10.1.2.2. 章***。
uart_list[4]	整形。停止位。详情请参考***第 3.10.1.2.3. 章***。

3.10.1.2.1. 数据位

数据位枚举定义如下：

- **参数**

参数	描述
0	数据位为 5
1	数据位为 6
2	数据位为 7
3	数据位为 8

3.10.1.2.2. 校验位

校验位枚举定义如下：

- **参数**

参数	描述
0	无校验
1	偶校验
2	奇校验
3	始终为 1
4	始终为 0

3.10.1.2.3. 停止位

停止位枚举定义如下：

- **参数**

参数	描述
0	停止位为 1
1	停止位为 1.5
2	停止位为 2

3.10.2. queclot.MBDataSend

该函数用于接收到 DMP 下发的物模型数据时，把数据转为 modbus 格式后发送到 modbus 从机设备。

- **函数原型**

```
quecIot.MBDataSend(data);
```

- **参数**

data：
[In] 字典类型。需要发送的数据内容。。

- **返回值**

True 函数执行成功
False 函数执行失败

3.10.3. queclot.MBDataRecv

该函数用于串口接收到 modbus 数据后，把数据转发到 modbus 组件中进行处理。

- **函数原型**

```
quecIot.MBDataRecv(port, data);
```

- **参数**

port：
[In] 整形。串口端口号。

data：
[In] 字节数组，串口接收到的 modbus 数据。

- **返回值**

True 函数执行成功
False 函数执行失败

3.10.4. queclot.MBDeinit

用于删除已经初始化的 modbus 组件。

- **函数原型**

```
quecIot.MBDeinit();
```

- **参数**

无

- **返回值**

True　　函数执行成功
False　　函数执行失败

4 相关事件描述

4.1. 设备接入设备管理平台相关事件

表 3：事件回调函数相关事件

Event	Errcode	Value	valLen	描述
1	10200	NULL	0	设备认证成功
	10420	NULL	0	请求数据错误（连接失败）
	10422	NULL	0	设备已认证（连接失败）
	10423	NULL	0	没有找到产品信息（连接失败）
	10424	NULL	0	PAYLOAD 解析失败（连接失败）
	10425	NULL	0	签名验证未通过（连接失败）
	10426	NULL	0	认证版本错误（连接失败）
	10427	NULL	0	散列信息不合法（连接失败）
	10430	NULL	0	PK 发生改变（连接失败）
	10431	NULL	0	DK 不合法（连接失败）
	10432	NULL	0	PK 与认证版本不匹配（连接失败）
	10450	NULL	0	设备内部错误（连接失败）
	10466	NULL	0	引导服务器地址未找到(连接失败）
	10500	NULL	0	设备认证失败(系统发生未知异常）
	10300	NULL	0	其他错误
2	10200	NULL	0	接入成功
	10430	NULL	0	设备密钥不正确（连接失败）

	10431	NULL	0	设备被禁用（连接失败）
	10450	NULL	0	设备内部错误（连接失败）
	10471	NULL	0	实现方案版本不支持(连接失败)
	10473	NULL	0	接入心跳异常（连接超时）
	10474	NULL	0	网络异常（连接超时）
	10475	NULL	0	服务器发生改变
	10476	NULL	0	连接 AP 异常
	10500	NULL	0	接入失败（系统发生未知异常）
3	10200	NULL	0	订阅成功
	10300	NULL	0	订阅失败
4	10200	NULL	0	透传数据发送成功
	10210	NULL	0	物模型数据发送成功
	10220	NULL	0	定位数据发送成功
	10300	NULL	0	透传数据发送失败
	10310	NULL	0	物模型数据发送失败
	10320	NULL	0	定位数据发送失败
5	10200	uint8_t*类型	字节长度	收到透传数据。下行数据为非缓存模式时，收到数据将直接下发
	10210	ttlv 指针类型	0	收到物模型下发数据。下行数据为非缓存模式时，收到数据将直接下发
	10211	uint16_t 数组类型，第一个值为 pkgId，剩下的是物模型 ID	物模型 ID 的数量	收到物模型查询命令，非缓存数据，收到直接下发
	10473	NULL	0	收到数据但长度超过模块 buffer 限制，接收失败
	10428	NULL	0	设备接收缓存过多导致限流
6	10200	NULL	0	注销成功（断开连接成功）
7	10700	字符串类型，格式： **"<componentNo>","<sourceVersion>","<targetVersion>",<batteryLimit>,<minSignalIntensity>,<useSpace>**	字符串长度	有升级任务,为配置信息 componentNo：组件标识 sourceVersion：源版本 targetVersion：目标版本 batteryLimit：OTA 升级最小电量

				minSignalIntensity：OTA 升级最小信号强度 useSpace：OTA 升级需要磁盘空间
	10701	字符串类型，格式：**"<componentNo>",length,"<MD5>"**	字符串长度	模块开始下载 componentNo：组件标识 Length：OTA 升级资源包大小 MD5：OTA 升级资源包 md5 值
	10702	NULL	0	包下载中
	10703	字符串类型，格式：**"<componentNo>","<length>","<startaddr>","<piece_length>"**	字符串长度	包下载完成 componentNo：组件标识 Length：OTA 升级资源包大小 startaddr：OTA 升级资源包 md5 值 piece_length：当前文件块大小
	10704	NULL	0	包更新中
	10705	NULL	0	更新固件完成
	10706	NULL	0	更新固件失败
8	10428	NULL	0	设备高频消息导致限流
	10429	NULL	0	超过单设备激活数量或者每日请求数导致限流

4.2. HTTP OTA 下载服务平台相关事件

表 4：HTTP OTA 事件回调函数相关事件

Event	Errcode	Value	valLen	描述
1	0	字符串类型，JSON 格式：OTA 升级事件信息	字符串长度	有升级任务
2	10700	NULL	0	设备无升级计划
	10701	NULL	0	系统忙
	10703	NULL	0	设备失败次数达到设置的升级重试次数
	10704	NULL	0	模组的信号强度小于升级策略中设置的信号强度
	10705	NULL	0	模组的电量小于升级策略中设置的电量
	10706	NULL	0	access_token 非法

	10707	NULL	0	access_token 过期
	10708	NULL	0	参数校验失败
	10709	NULL	0	设备认证失败
	10710	NULL	0	密码错误
	10713	NULL	0	系统未知错误
	10714	NULL	0	获取升级包 url 失败
	10715	NULL	0	重试时间间隔小于升级策略中设置的时间间隔
	10716	NULL	0	模组 imei 号处于多个正在升级的计划中
	10600	NULL	0	模组电量不满足升级要求
	10601	NULL	0	模组信号强度不满足升级要求
	10602	NULL	0	请求升级包超时
	10603	NULL	0	模组的剩余空间不满足升级要求
	10604	NULL	0	请求升级包失败
3	0	NULL	0	模组开始下载
4	0	NULL	0	下载完成
5	10605	NULL	0	升级包下载过程发生未知错误
6	0	NULL	0	下载失败
7	0	NULL	0	开始升级
8	10606	NULL	0	升级包 MD5 值校验失败
	10607	NULL	0	升级包文件格式校验失败
	10608	NULL	0	升级包的真实版本和设置的目标版本不一致
	10609	NULL	0	模组文件系统没有足够的升级空间
	10610	NULL	0	补丁与模块中的源文件不匹配
	10611	NULL	0	升级进程因不可预测错误退出，系统将会重启，升级重试中
	10612	NULL	0	升级进程因不可预测错误退出，而且已达到重试次数，升级失败

4.3. 子设备连接网关相关事件

表 5：子设备事件回调函数相关事件

Event	Errcode	Value	valLen	描述
1	10200	uint8_t*类型	字节长度	子设备设备认证成功
	10420	NULL	0	请求数据错误（连接失败）
	10422	NULL	0	设备已认证（连接失败）
	10423	NULL	0	没有找到产品信息（连接失败）
	10424	NULL	0	PAYLOAD 解析失败（连接失败）
	10425	NULL	0	签名验证未通过（连接失败）
	10426	NULL	0	认证版本错误（连接失败）
	10427	NULL	0	散列信息不合法（连接失败）
	10430	NULL	0	PK 发生改变（连接失败）
	10431	NULL	0	DK 不合法（连接失败）
	10432	NULL	0	PK 与认证版本不匹配（连接失败）
	10450	NULL	0	设备内部错误（连接失败）
	10466	NULL	0	引导服务器地址未找到(连接失败）
	10440	NULL	0	网关与子设备没有关联关系
	10500	NULL	0	设备认证失败(系统发生未知异常）
	10300	NULL	0	其他错误
2	10200	NULL	0	接入成功
	10430	NULL	0	设备密钥不正确（连接失败）
	10431	NULL	0	设备被禁用（连接失败）
	10440	NULL	0	网关与子设备没有关联关系
	10441	NULL	0	子设备已连接（连接失败）

	10450	NULL	0	设备内部错误（连接失败）
	10471	NULL	0	实现方案版本不支持(连接失败)
	10473	NULL	0	接入心跳异常（连接超时）
	10474	NULL	0	网络异常（连接超时）
	10475	NULL	0	服务器发生改变
	10476	NULL	0	连接 AP 异常
	10500	NULL	0	接入失败（系统发生未知异常）
3	10200	NULL	0	订阅成功
	10300	NULL	0	订阅失败
4	10200	NULL	0	透传数据发送成功
	10210	NULL	0	物模型数据发送成功
	10300	NULL	0	透传数据发送失败
	10310	NULL	0	物模型数据发送失败
5	10200	uint8_t*类型	字节长度	收到透传数据。下行数据为非缓存模式时，收到数据将直接下发
	10210	ttlv 指针类型	0	收到物模型下发数据。下行数据为非缓存模式时，收到数据将直接下发
	10211	uint16_t 数组类型，第一个值为 pkgId，剩下的是物模型 ID	物模型 ID 的数量	收到物模型查询命令，非缓存数据，收到直接下发
	10473	NULL	0	收到数据但长度超过模块 buffer 限制，接收失败
	10428	NULL	0	设备接收缓存过多导致限流
6	10200	NULL	0	子设备登出成功（断开连接成功）
8	10428	NULL	0	设备高频消息导致限流
	10429	NULL	0	超过单设备激活数量或者每日请求数导致限流
	10442	NULL	0	子设备没有登录
9	10200	NULL	0	子设备注销成功

5 示例

```
import quecIot
import misc
import utime as time
import pm
import osTimer
import uos
from machine import Pin

DEMO_VERSION = "21060101"

data_trans_ready = False
gpio_toggle_flag = False

def get_battery_vol():
    """
    获取电池电压，单位 mV
    """
    return misc.Power.getVbatt()

"""
    模组 OTA 升级和 python 脚本升级
"""
def ota_event(event):
    if 7 == event[0] and 10700 == event[1]:
        """ 检查组件标志，组件标志在平台创建 """
        if "SCRIPT" in event[2]:
            quecIot.otaAction(1)
        elif "IMEI" in event[2]:
            quecIot.otaAction(1)
    elif 7 == event[0] and 10701 == event[1]:
        pass
    elif 7 == event[0] and 10703 == event[1]:
        if "SCRIPT" in event[2]:

            uos.rename("usr/demo_quecthing.py","usr/demo_quecthing.bk")
            uos.rename("usr/qiot_ota.bin","usr/demo_quecthing.py")
```

```
            """ 设置版本，上报升级成功 """
            quecIot.setMcuVersion("SCRIPT","v2")
        elif "IMEI" in event[2]:
            pass

class QuecThing:
    def __init__(self):
        """ 初始化 qucsdk """
        quecIot.init()
        """ 注册事件回调函数 """
        quecIot.setEventCB(self.eventCB)
        """ 配置产品信息"""
        quecIot.setProductinfo("p1116a","UHg1dTRBRVh3MkVG")
        """ 配置服务器信息，可选，默认连接 MQTT 生产环境服务器 """
        quecIot.setServer(1,"http://iot-south.quectel.com:2883")
        """ 配置 Lifetime，可选，MQTT 默认为 120 """
        quecIot.setLifetime(120)
        """ 配置外部 MCU 标识号和版本号，可选，如没有外部 MCU 则不需要配置 """
        quecIot.setMcuVersion("MCU1","1_0_0")

        """ 创建定时器任务 """
        ostimer = osTimer()
        """ 每一百秒调用一次回调函数 """
        ostimer.start(100000,1,self.mainTask)

        """" 创建 GPIO 对象 """
        self.gpio1 = Pin(Pin.GPIO1, Pin.OUT, Pin.PULL_DISABLE, 0)

        """ 设置自动休眠模式 """
        pm.autosleep(1)

        """ 启动云平台连接 """
        quecIot.setConnmode(1)

    @staticmethod
    def eventCB(data_list):
        print("\r\n{},{}\r\n".format(str(data_list[0]), str(data_list[1])))
        if len(data_list) == 3:
            print(data_list[2])

        ota_event(data_list)
        if 1 == data[0] and 10422 == data_list[1]:
            quecIot.setConnmode(0)
            exit(0)
```

```
        elif 3 == data_list[0] and 10200 == data_list[1]:
            """
            测试发送物模型数据
            向平台发送数据需要在返回 3, 10200 之后进行
            """
            global data_trans_ready
            data_trans_ready = True


            """ 发送 bool 型数据"""
            quecIot.phymodelReport(1, {2: True})
            """ 发送数值 """
            # 整数
            quecIot.phymodelReport(1, {3: 123})
            # 浮点数
            quecIot.phymodelReport(1, {9: 123.123})
            """ 发送 array """
            quecIot.phymodelReport(1, {4: [1, 2, 3]})
            """ 发送结构体 """
            quecIot.phymodelReport(1, {6: {8: 1.0, 7: 1.0}})

            """ 发送中文，透传数据和物模型数据无法同时在平台调试 """

            """
                bytes_temp = bytes("中文".encode("utf-8"))
                quecIot.passTransSend(1,bytes_temp)
            """


    def mainTask(self, argv):
        global data_trans_ready
        global gpio_toggle_flag
        """
        主任务，上报电池电压
        """
        if data_trans_ready:
            print("main task")
            """ 获取电压 """
            battery_vol = get_battery_vol()

            """ 发送物模型数据 """
            quecIot.phymodelReport(0, {110 : battery_vol})

            """ 翻转引脚电平 """
            # gpio_toggle_flag = not gpio_toggle_flag
```

```
                    # self.gpio1.write(int(gpio_toggle_flag))



if __name__ == "__main__":
    print("\r\n**********\r\n")
    print(DEMO_VERSION)
    quecthing = QuecThing()
```

6 附录

表 6：参考文档

文档名称
[1] Quectel_BMS_云平台使用手册

表 7：术语缩写

缩写	英文全称	中文全称
CRC	Cyclic Redundancy Check	循环冗余校验
DMP	Device Management Platform	设备管理平台
DK	Device Key	设备标识
FOTA	Firmware Over-The-Air	固件空中升级
GGA	Global Positioning System Fix Data	全球定位系统定位数据
GSA	GPS DOP and Active Satellites	GNSS 精度因子（DOP）与有效卫星
GSV	GNSS Satellites in View	可视的 GNSS 卫星
ICCID	Integrated Circuit Card Identifier	集成电路卡识别码
ID	Mostly Refers to Identifier in Terms of Software	软件中多数指“标识符”
IMEI	International Mobile Equipment Identity	国际移动设备识别码
IMSI	Internation Mobile Subscriber Identity	国际移动用户识别码
IoT	Internet of Things	物联网
JSON	JavaScript Object Notation	JavaScript 对象表示法
LAC	Location Area Code	位置区码

LBS	Location-based Service	基于位置的服务
LwM2M	Lightweight M2M	轻量化 M2M（协议）
MCC	Mobile Country Code	移动设备国家代码
MCU	Microprogrammed Control Unit	微程序控制器
MD5	message-digest algorithm 5	信息-摘要算法
MNC	Mobile Network Code	移动设备网络代码
MQTT	Message Queuing Telemetry Transport	消息队列遥测传输
OTA	Over-the-air programming	空中编程
PDP	Packet Data Protocol	分组数据协议
PK	Product Key	产品标识
QoS	Quality of Service	服务质量
REPL	Read-Eval-Print-Loop	交互时编程环境
RMC	Recommended Minimum Specific GNSS Data	推荐的最少专用 GNSS 数据
RSRP	Reference Signal Received Power	参考信号接收功率
RSRQ	Reference Signal Received Quality	参考信号接收质量
RSSI	Received Signal Strength Indicator	接收信号强度指示
SDK	Software Development Kit	软件开发工具包
SNR	Signal-to-Noise Ratio	信噪比
SOTA	Software Over the Air	空中软件
TA	Terminal Adapter	终端适配器
URL	Uniform Resource Locator	统一资源定位符
VTG	Course Over Ground & Ground Speed	对地航向和对地速度