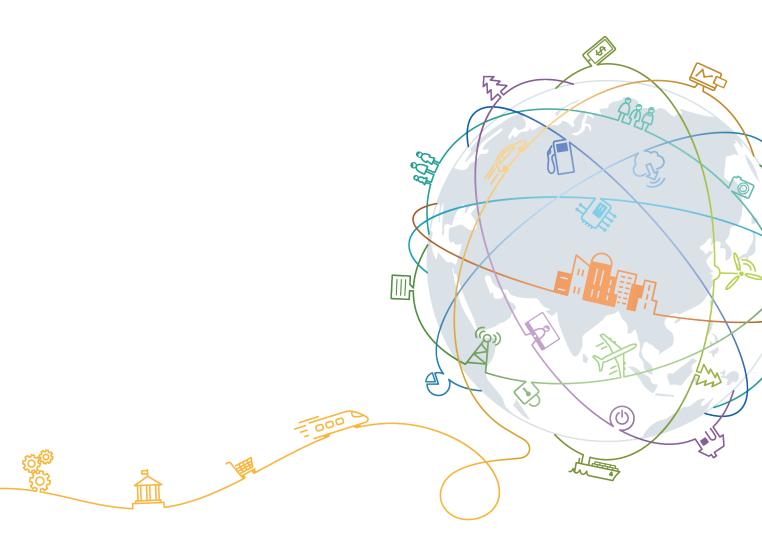
## 02 设备接入 IoT Device SDK API 参考(C)-正式发布

## 02 设备接入 loT Device SDK API 参考 (C)-正式发布

文档版本 01

发布日期 2021-02-26





#### 版权所有 © 华为技术有限公司 2021。 保留一切权利。

非经本公司书面许可,任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部,并不得以任何形式传播。

#### 商标声明



HUAWE和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标,由各自的所有人拥有。

#### 注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束,本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定,华为公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因,本文档内容会不定期进行更新。除非另有约定,本文档仅作为使用指导,本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

### 华为技术有限公司

地址: 深圳市龙岗区坂田华为总部办公楼 邮编: 518129

网址: <a href="https://www.huawei.com">https://www.huawei.com</a>

客户服务邮箱: support@huawei.com

客户服务电话: 4008302118

## 目录

1 使用前必读	
2 开发说明	4
2.1 数据类型说明	4
3 直连设备接入	
3.1 初始化 SDK 资源	
3.2 释放 SDK 资源	
3.3 绑定配置	
3.4 设置日志打印回调函数	
3.5 设置回调函数	
3.5.1 设置协议层回调函数	
3.5.2 设置业务层回调函数	
3.6 设备连接	
3.7 设备断开链路	
4 直连设备上报数据	
4.1 直连设备上报自定义 topic 消息	
4.1 直连设备上报百定义 topic	
4.3 直连设备上报商住	
4.4 直连设备上报设备属性设置结果	
4.5 直连设备上报设备属性查询结果	
+.5	
4.7 直连设备上报新增子设备请求	
4.8 直连设备上报删除子设备请求	
5 <b>直连设备命令接收</b>	
5.1 接收消息下发	
5.2 接收自定义 topic 消息	
5.2.1 订阅自定义 topic	
5.2.2 接收自定义 topic 消息	
5.3 接收命令下发	
5.4 接收平台设置设备属性5.5 接收平台查询设备属性	
5.6 接收平台新增子设备通知	
5.0 接收平台新增于设备通知	
3.7 按拟十口删除了以留遇叫	40

日求
43
45
48
48
48
51
51
53
54
56
56
59
59
60
62
62
63
65
65
67
67
69

# **1** 使用前必读

#### 概述

物联网平台提供了IoT Device SDK(以下简称SDK),帮助设备快速连接到物联网平台。支持TCP/IP协议栈的设备在集成IoT Device SDK后,可以直接与物联网平台通信。不支持TCP/IP协议栈的设备例如蓝牙设备、ZigBee设备等,可以通过集成了IoT Device SDK的网关将设备数据转发给物联网平台,与平台进行通信。

#### 接口列表

SDK提供的接口功能如下所示。

• 直连设备:通过设备鉴权直接接入物联网平台的设备。

• 非直连设备:通过网关设备接入物联网平台的设备。

功能	接口	说明
直连设备接入	IOTA_Init 3.1 初始化SDK资源	
	IOTA_Destroy	3.2 释放SDK资源
	IOTA_ConfigSetXXX /	3.3 绑定配置
	IOTA_SetPrintLogCallback	3.4 设置日志打印回调函数
	IOTA_SetCallbackXXX	3.5 设置回调函数
	IOTA_Auth	3.6 设备连接
	IOTA_DisAuth	3.7 设备断开链路
直连设备上报数据	IOTA_MessageReport  4.1 直连设备上报自定义 topic消息	
	IOTA_PropertiesReport	4.2 直连设备上报属性
	IOTA_CommandResponse	4.3 直连设备上报命令执行结果
	IOTA_PropertiesSetResponse	4.4 直连设备上报设备属性 设置结果

功能	接口	说明
	IOTA_PropertiesGetResponse	4.5 直连设备上报设备属性 查询结果
	IOTA_UpdateSubDeviceStatus	4.6 直连设备上报子设备状态
直连设备命令接收 (相关回调接口)	HW_VOID (*PFN_MESSAGE_CALLBACK_HA NDLER)(EN_IOTA_MESSAGE *message)	5.1 接收消息下发
	HW_VOID (*PFN_CMD_CALLBACK_HANDL ER)(EN_IOTA_COMMAND *message)	5.3 接收命令下发
	HW_VOID  (*PFN_PROP_SET_CALLBACK_H ANDLER)  (EN_IOTA_PROPERTY_SET *message)	5.4 接收平台设置设备属性
	HW_VOID (*PFN_PROP_GET_CALLBACK_H ANDLER) (EN_IOTA_PROPERTY_GET *message)	5.5 接收平台查询设备属性
	HW_VOID (*PFN_EVENT_CALLBACK_HAND LER)(EN_IOTA_EVENT *message)	5.6 接收平台新增子设备通知
	HW_VOID (*PFN_EVENT_CALLBACK_HAND LER)(EN_IOTA_EVENT *message)	5.7 接收平台删除子设备通知
子设备上报数据	IOTA_MessageReport	6.1 子设备上报消息
	IOTA_BatchPropertiesReport	6.2 批量上报设备属性
OTA升级	HW_VOID (*PFN_CALLBACK_HANDLER)	7.1 平台下发获取版本信息 通知
	IOTA_OTAVersionReport	7.2 设备上报软固件版本
	HW_VOID (*PFN_CALLBACK_HANDLER)	7.3 平台下发升级通知
	IOTA_GetOTAPackages	7.4 设备请求下载包
	IOTA_OTAStatusReport	7.5 设备上报升级状态

功能	接口	说明
设备影子	IOTA_GetDeviceShadow	8.1 设备请求获取平台的设 备影子数据
	HW_VOID (*PFN_CALLBACK_HANDLER_WI TH_TOPIC)	8.2 设备接收平台返回的设 备影子数据

## **2** 开发说明

#### 2.1 数据类型说明

## 2.1 数据类型说明

#### 常用数据类型

类型名称	类型原型
HW_INT	int
HW_UINT	unsigned int
HW_CHAR	char
HW_UCHAR	unsigned char
HW_BOOL	int
HW_ULONG	unsigned long
HW_USHORT	unsigned short
HW_MSG	void*
HW_VOID	void
HW_NULL	0

#### 函数标准返回值

返回值名称	值	描述
IOTA_SUCCESS	0	执行成功。
IOTA_FAILURE	-1	执行错误。

返回值名称	值	描述
IOTA_PARAMETER_EMPT Y	-101	参数为空。
IOTA_RESOURCE_NOT_A VAILABLE	-102	资源不被允许。
IOTA_INITIALIZATION_RE PEATED	-103	重复初始化。
IOTA_LIBRARY_LOAD_FAI LED	-104	加载库文件失败。
IOTA_SECRET_ENCRYPT_ FAILED	-105	加密密钥失败。
IOTA_MQTT_CONNECT_F AILED	-106	连接失败。
IOTA_MQTT_CONNECT_E XISTED	-107	连接已存在。
IOTA_CERTIFICATE_NOT_ FOUND	-108	找不到证书。
IOTA_MQTT_DISCONNEC T_FAILED	-109	断链失败。
IOTA_PARSE_JSON_FAILE D	-110	解析字符串失败。
IOTA_PARAMETER_ERRO R	-111	参数错误。
IOTA_NUMBER_EXCEEDS	-112	数字超过最大值。

# **3** <sub>直连设备接入</sub>

- 3.1 初始化SDK资源
- 3.2 释放SDK资源
- 3.3 绑定配置
- 3.4 设置日志打印回调函数
- 3.5 设置回调函数
- 3.6 设备连接
- 3.7 设备断开链路

## 3.1 初始化 SDK 资源

#### 接口功能

初始化SDK资源。

#### 接口描述

HW\_INT IOTA\_Init(HW\_CHAR \*pcWorkPath)

#### 参数说明

字段	必选/可选	类型	描述
pcWorkPat h	必选	String	SDK工作路径,用于存放SDK的配置 文件,工作路径必须有效,该参数必 须带结束符'\0'。建议设置为"."(当 前路径),默认的证书文件在当前路径 的conf文件夹下。

#### 接口返回值

参见函数标准返回值。

#### 示例

// 开发者调用该接口初始化资源 IOTA\_Init(".");//当前目录

## 3.2 释放 SDK 资源

#### 接口功能

调用此函数,SDK会释放申请的所有动态资源(内存、线程等等)。

#### 接口描述

HW\_INT IOTA\_Destroy()

#### 接口返回值

参见函数标准返回值

#### 示例

// 开发者调用该接口销毁资源 IOTA\_Destroy();

## 3.3 绑定配置

#### 接口功能

配置SDK相关参数。

#### 接口描述

HW\_INT IOTA\_ConfigSetStr(HW\_INT iItem, HW\_CHAR \*pValue) HW\_INT IOTA\_ConfigSetUint(HW\_INT iItem, HW\_UINT uiValue)

#### 参数说明

字段	必选/可选	类型	描述
字段 iltem (key)	<b>必选/可选</b> 必选	### HW_UINT	<ul> <li>協・設・ では、</li></ul>

字段	必选/可选	类型	描述
pValue/ uiValue (value)	必选	HW_CHAR */ HW_UINT	设置的值。  设备ID:设备注册时返回  设备密钥:设备注册时返回  平台IP:SDK对接平台地址  平台端口:8883  日志的facility类型:记录日志的来源。可以选择LOG_LOCAL0~LOG_LOCAL7中的任意一个  日志的显示级别:可以按需选择LOG_ERR、LOG_WARNING、LOG_INFO、LOG_DEBUG中的一个  MQTT链接保活时间:单位s  MQTT连接超时时间:单位s  MQTT连接超时时间:单位s  MQTT尝试重连时间:单位s  MQTT尝试重连时间:单位s  MQTT尝试重连时间:单位s  海急发布Qos设置:0:最多一次;1:至少一次;2是只发一次。不设置默认设置为1。  设备接入模式:EN_IOTA_CFG_AUTH_MODE_SECRET是密码接入模式,EN_IOTA_CFG_AUTH_MODE_CERT是证书接入模式

#### 接口返回值

#### 参见函数标准返回值

#### 示例

```
// 开发者调用该接口设置参数
IOTA_ConfigSetStr(EN_IOTA_CFG_MQTT_ADDR, serverlp_);
IOTA_ConfigSetUint(EN_IOTA_CFG_MQTT_PORT, port_);
IOTA_ConfigSetStr(EN_IOTA_CFG_DEVICEID, username_);
IOTA_ConfigSetStr(EN_IOTA_CFG_DEVICESECRET, password_);
IOTA_ConfigSetUint(EN_IOTA_CFG_AUTH_MODE, EN_IOTA_CFG_AUTH_MODE_SECRET);
#ifdef_SYS_LOG
IOTA_ConfigSetUint(EN_IOTA_CFG_LOG_LOCAL_NUMBER, LOG_LOCAL7);
IOTA_ConfigSetUint(EN_IOTA_CFG_LOG_LEVEL, LOG_INFO);
#endif
```

## 3.4 设置日志打印回调函数

### 接口功能

SDK的日志呈现方式可以自定义回调函数来实现。

#### 接口描述

void IOTA\_SetPrintLogCallback(PFN\_LOG\_CALLBACK\_HANDLER pfnLogCallbackHandler);

#### 参数说明

字段	必选/可选	类型	描述
pfnLogCallba ckHandler	必选	PFN_LOG_C ALLBACK_H ANDLER	入参为自定义函数名称。

#### PFN\_LOG\_CALLBACK\_HANDLER 类型:

HW\_VOID (\*PFN\_LOG\_CALLBACK\_HANDLER)(int level, char\* format, va\_list args);

字段	必选/可选	类型	描述
level	必选	int	日志打印等级,可以填写如下四种:
			EN_LOG_LEVEL_DEBUG
			EN_LOG_LEVEL_INFO
			EN_LOG_LEVEL_WARNING
			EN_LOG_LEVEL_ERROR
format	必选	String	打印的日志内容 ,里面可以包含要输出的变量。
args	可选	String/int	要打印的变量值。

#### 接口返回值

#### 参见函数标准返回值

```
//开发者自定义日志打印函数
void myPrintLog(int level, char* format, va_list args)
{
    vprintf(format, args); //日志打印在控制台
// vsyslog(level, format, args); //日志打印在系统文件中
}
//开发者设置自定义日志打印函数
IOTA_SetPrintLogCallback(myPrintLog);
```

#### □ 说明

- 用vprintf函数打印的日志,显示在控制台中。
- 用vsyslog函数打印的日志,显示在系统日志文件中。一般会在"/var/log/messages"文件里 (可以考虑按需分包)。建议自行实现日志打印函数。

由于linux下的DEBUG日志需要在调试的时候查看,如果想让DEBUG日志打印在控制台上,可以调低DEBUG日志级别再打印。

#### 例如:

## 3.5 设置回调函数

设备收到下行数据时,开发者可以通过提前设置回调函数,来对下行数据进行处理,下行数据主要包括协议层和业务层,SDK已实现自动订阅业务层相关的TOPIC ,业务层下行数据可参考接口参数说明。

### 3.5.1 设置协议层回调函数

#### 接口描述

IOTA SetProtocolCallback函数主要用于设置协议层回调函数。

HW\_VOID IOTA\_SetProtocolCallback(HW\_INT iItem, PFN\_PROTOCOL\_CALLBACK\_HANDLER pfnCallbackHandler)

#### 参数说明

字段	必选/可选	类型	描述
iltem	必选	HW_INT	回调函数对应的通知: 1. 鉴权成功的通知: EN_IOTA_CALLBACK_CONNECT_S UCCESS 2. 鉴权失败的通知: EN_IOTA_CALLBACK_CONNECT_F AILURE 3. 链接断开的通知: EN_IOTA_CALLBACK_CONNECTIO N_LOST 4. 设备主动断链成功的通知: EN_IOTA_CALLBACK_DISCONNEC T_SUCCESS 5. 设备主动断链失败的通知: EN_IOTA_CALLBACK_DISCONNEC T_FAILURE 6. 设备订阅成功的通知: EN_IOTA_CALLBACK_SUBSCRIBE_ SUCCESS 7. 设备订阅失败的通知: EN_IOTA_CALLBACK_SUBSCRIBE_ FAILURE 8. 设备发布数据成功的通知: EN_IOTA_CALLBACK_PUBLISH_SU CCESS 9. 设备发布数据失败的通知: EN_IOTA_CALLBACK_PUBLISH_SU CCESS 9. 设备发布数据失败的通知: EN_IOTA_CALLBACK_PUBLISH_FAI LURE
pfnCallbackH andler	必选	PFN_PROTO COL_CALLBA CK_HANDLE R	自定义函数名称

#### PFN\_CALLBACK\_HANDLER类型:

HW\_VOID (\*PFN\_PROTOCOL\_CALLBACK\_HANDLER) (EN\_IOTA\_MQTT\_PROTOCOL\_RSP \*message)

#### EN\_IOTA\_MQTT\_PROTOCOL\_RSP类型:

字段	必选/可选	类型	描述
mqtt_msg_i nfo	必选	EN_IOTA_M QTT_MSG_I NFO	mqtt协议层返回信息

字段	必选/可选	类型	描述
message	必选	HW_CHAR*	消息体

#### EN\_IOTA\_MQTT\_MSG\_INFO类型:

字段	必选/可选	类型	描述
context	必选	HW_VOID	指向任何应用程序特定上下文的指针。 上下文指针被传递给每个回调函数,以 提供对回调中的上下文信息的访问。当 前预留,默认为NULL。
messageId	必选	HW_INT	消息ID
code	必选	HW_INT	消息码(业务层通知该值默认为0)

#### 示例

```
// 开发者设置回调函数
//HandleConnectSuccess、HandleConnectFailure等函数可参考SDK中src/device_demo/device_demo.c中的实现
void SetMyCallbacks() {
    IOTA_SetProtocolCallback(EN_IOTA_CALLBACK_CONNECT_SUCCESS, HandleConnectSuccess);
    IOTA_SetProtocolCallback(EN_IOTA_CALLBACK_CONNECT_FAILURE, HandleConnectFailure);

    IOTA_SetProtocolCallback(EN_IOTA_CALLBACK_DISCONNECT_SUCCESS, HandleDisConnectSuccess);
    IOTA_SetProtocolCallback(EN_IOTA_CALLBACK_DISCONNECT_FAILURE, HandleDisConnectFailure);
    IOTA_SetProtocolCallback(EN_IOTA_CALLBACK_DISCONNECTION_LOST, HandleConnectionLost);

    IOTA_SetProtocolCallback(EN_IOTA_CALLBACK_SUBSCRIBE_SUCCESS, HandleSubscribesuccess);
    IOTA_SetProtocolCallback(EN_IOTA_CALLBACK_SUBSCRIBE_FAILURE, HandleSubscribeFailure);

    IOTA_SetProtocolCallback(EN_IOTA_CALLBACK_PUBLISH_SUCCESS, HandlePublishSuccess);
    IOTA_SetProtocolCallback(EN_IOTA_CALLBACK_PUBLISH_FAILURE, HandlePublishFailure);
```

### 3.5.2 设置业务层回调函数

#### 接口描述

以下函数均为设置业务层回调函数。

函数	功能	入参描述
HW_VOID IOTA_SetMessageCal lback(PFN_MESSAG E_CALLBACK_HAND LER pfnCallbackHandler)	设置 消息回调	入参为自定义回调函数指针,请参考 <b>5.1</b>

函数	功能	入参描述
HW_VOID IOTA_SetUserTopicM sgCallback(PFN_USE R_TOPIC_MSG_CALL BACK_HANDLER pfnCallbackHandler)	设置 自定义TOPIC消息 回调	入参为自定义回调函数指针,请 参考 <b>5.2</b>
HW_VOID IOTA_SetCmdCallbac k(PFN_CMD_CALLB ACK_HANDLER pfnCallbackHandler)	设置命令回调	入参为自定义回调函数指针,请 参考5.3
HW_VOID IOTA_SetPropSetCall back(PFN_PROP_SET _CALLBACK_HANDL ER pfnCallbackHandler)	设置平台属性设置回调	入参为自定义回调函数指针,请参考 <b>5.4</b>
HW_VOID IOTA_SetPropGetCall back(PFN_PROP_GE T_CALLBACK_HAND LER pfnCallbackHandler)	设置平台属性查询回调	入参为自定义回调函数指针,请 参考 <b>5.5</b>
HW_VOID IOTA_SetEventCallba ck(PFN_EVENT_CAL LBACK_HANDLER pfnCallbackHandler)	设置 事件回调( <b>子设备</b> 新增/删除、OTA升级命 令均属于事件)	入参为自定义回调函数指针,子设备相关处理可参考5.6、5.7,OTA相关可参考7.1、7.3。
HW_VOID IOTA_SetShadowGet Callback(PFN_SHAD OW_GET_CALLBACK _HANDLER pfnCallbackHandler)	设置 设备影子回调	入参为自定义回调函数指针,请参考 <b>8.2</b>

```
IOTA_SetProtocolCallback(EN_IOTA_CALLBACK_PUBLISH_FAILURE, HandlePublishFailure);

//业务层回调函数
IOTA_SetMessageCallback(HandleMessageDown);
IOTA_SetUserTopicMsgCallback(HandleUserTopicMessageDown);
IOTA_SetCmdCallback(HandleCommandRequest);
IOTA_SetPropSetCallback(HandlePropertiesSet);
IOTA_SetPropGetCallback(HandlePropertiesGet);
IOTA_SetPropGetCallback(HandlePropertiesGet);
IOTA_SetEventCallback(HandleEventsDown);
IOTA_SetShadowGetCallback(HandleDeviceShadowRsp);
```

## 3.6 设备连接

#### 接口功能

设备可以调用鉴权接口来连接loT平台。

#### 接口描述

HW\_INT IOTA\_Connect()

#### 接口返回值

参见函数标准返回值

#### 示例

```
//开发者调用鉴权接口
int ret = IOTA_Connect();
if (ret != 0)
{
    printfLog(EN_LOG_LEVEL_ERROR, "AgentLiteDemo: IOTA_Connect() error, Auth failed, result %d\n", ret);
}
```

## 3.7 设备断开链路

#### 接口功能

设备可以调用断开链路接口主动与IoT平台断开。

#### 接口描述

HW\_INT IOTA\_DisConnect()

#### 接口返回值

参见函数标准返回值

```
// 开发者调用断开链路接口:
int ret = IOTA_DisConnect();
if (ret != 0)
{
    printfLog(EN_LOG_LEVEL_ERROR, "AgentLiteDemo: IOTA_DisConnect() error, DisAuth failed, result %d\n",
```

ret); }

## 4 直连设备上报数据

- 4.1 直连设备上报自定义topic消息
- 4.2 直连设备上报属性
- 4.3 直连设备上报命令执行结果
- 4.4 直连设备上报设备属性设置结果
- 4.5 直连设备上报设备属性查询结果
- 4.6 直连设备上报子设备状态
- 4.7 直连设备上报新增子设备请求
- 4.8 直连设备上报删除子设备请求

## 4.1 直连设备上报自定义 topic 消息

#### 接口功能

可以通过该接口上报平台不解析的透传消息。

#### 接口描述

HW\_INT IOTA\_MessageReport(HW\_CHAR \*object\_device\_id, HW\_CHAR \*name, HW\_CHAR \*id, HW\_CHAR \*content, HW\_CHAR \*topicParas, HW\_INT compressFlag, void \*context)

#### 参数说明

字段名	必选/可选	类型	参数描述
object_device _id	可选	HW_CHAR*	消息对应的最终目标设备,传NULL则 表示目标设备即网关设备。
name	可选	HW_CHAR*	消息名称
id	可选	HW_CHAR*	消息的唯一标识
content	必选	HW_CHAR*	消息内容。

字段名	必选/可选	类型	参数描述
topicParas	可选	HW_CHAR*	自定义topic参数,例如"devMsg" (前面不要加'/'或者特殊字符),如果 设置为NULL,则用平台默认topic上 报。
compressFlag	必选	HW_INT	是否压缩上报,0代表非压缩上报,1代 表压缩上报。压缩上报会消耗一定的内 存,如果没有特定流量要求的话,建议 使用非压缩上报。
context	可选	void *	指向任何数据体的上下文指针,该指针 会在成功或者失败的回调函数中返回。 不使用该指针可以传递NULL。

#### 接口返回值

#### 参见函数标准返回值

#### 示例

## 4.2 直连设备上报属性

#### 接口功能

网关可以该接口上报平台解析的设备属性值,设备属性需与profile中设置的一致。

#### 接口描述

 $HW_INT\ IOTA\_PropertiesReport(ST_IOTA\_SERVICE\_DATA\_INFO\ pServiceData[],\ HW_INT\ serviceNum,\ HW_INT\ compressFlag,\ void\ *context)$ 

#### 参数说明

字段	必选/可选	类型	描述
pServiceDat a[]	必选	ST_IOTA_SER VICE_DATA_I NFO	上报的属性数据结构体

字段	必选/可选	类型	描述
serviceNum	必选	HW_INT	上报的服务个数
compressFla g	必选	HW_INT	是否压缩上报,0代表非压缩上报,1代表压缩上报。压缩上报会消耗一定的内存,如果没有特定流量要求的话,建议使用非压缩上报。
context	可选	void *	指向任何数据体的上下文指针,该 指针会在成功或者失败的回调函数 中返回。不使用该指针可以传递 NULL。

#### ST\_IOTA\_SERVICE\_DATA\_INFO 类型:

字段	必选/可选	类型	描述
service_id	必选	HW_CHAR*	设备服务的ID,可从profile中获取
event_time	可选	HW_CHAR*	设备采集数据UTC时间(格式: yyyyMMddTHHmmssZ),如: 20161219T114920Z。 设备上报数据将参数设置为NULL时, 则数据上报时间以平台时间为准。
properties	必选	HW_CHAR*	一个服务的数据,具体字段在profile里 定义。注意:格式需能解析成JSON, 可参考下方的样例。

#### 接口返回值

#### 参见函数标准返回值

```
services[0].event_time = NULL;
services[1].service_id = "CPU";
services[1].properties = service2;

int messageId = IOTA_PropertiesReport(services, serviceNum);
if(messageId != 0)
{
    printfLog(EN_LOG_LEVEL_ERROR, "AgentLiteDemo: Test_propertiesReport() failed, messageId %d\n", messageId);
}
```

## 4.3 直连设备上报命令执行结果

#### 接口功能

当网关收到平台下发的命令(5.3)后,可以调用该接口上报命令执行结果。

#### 接口描述

HW\_INT IOTA\_CommandResponse(HW\_CHAR \*requestId, HW\_INT result\_code, HW\_CHAR \*response\_name, HW\_CHAR \*pcCommandResponse, void \*context)

#### 参数说明

字段	必选/可选	类型	描述
requestId	必选	HW_CHAR*	请求的唯一标识,设备收到的消息带该参数时,响应消息需要将 该参数值返回给平台。
result_code	必选	HW_INT	标识命令的执行结果,0表示成 功,其他表示失败。不带默认认 为成功。
response_na me	可选	HW_CHAR*	命令的响应名称,在设备关联的 产品模型中定义。
pcCommand Response	必选	HW_CHAR*	命令响应结果,需跟profile中定 义的commandResponse保持一 致。注意:格式需能解析成 JSON,可参考下方的样例。
context	可选	void *	指向任何数据体的上下文指针, 该指针会在成功或者失败的回调 函数中返回。不使用该指针可以 传递NULL。

#### 接口返回值

#### 参见**函数标准返回值**

```
// 开发者调用该接口上报命令执行结果
char *pcCommandResponse = "{\"SupWh\": \"aaa\"}"; // in service accumulator
```

```
int result_code = 0;
char *response_name = "cmdResponses";
    char *requestId = "1005";

int messageId = IOTA_CommandResponse(requestId, result_code, response_name, pcCommandResponse);
if(messageId != 0)
{
    printfLog(EN_LOG_LEVEL_ERROR, "AgentLiteDemo: Test_commandResponse() failed, messageId %d
\n", messageId);
}
```

## 4.4 直连设备上报设备属性设置结果

#### 接口功能

当网关收到平台下发的设置设备属性命令(5.4)后,可以调用该接口上报结果。

#### 接口描述

HW\_INT IOTA\_PropertiesSetResponse(HW\_CHAR \*requestId, HW\_INT result\_code, HW\_CHAR \*result\_desc, void \*context)

#### 参数说明

字段	必选/可选	类型	描述
requestId	必选	HW_CHAR*	请求的唯一标识,设备收到的消息带该参数时,响应消息需要将 该参数值返回给平台。
result_code	必选	Integer	命令的执行结果,0表示成功,其 他表示失败。不带默认认为成 功。
result_desc	可选	String	属性设置的响应描述,可以设置 为NULL。
context	可选	void *	指向任何数据体的上下文指针, 该指针会在成功或者失败的回调 函数中返回。不使用该指针可以 传递NULL。

#### 接口返回值

#### 参见函数标准返回值

```
// 开发者调用该接口上报属性设置结果
// int messageId = IOTA_PropertiesSetResponse(requestId, 0, "success");
    int messageId = IOTA_PropertiesSetResponse(requestId, 0, NULL);
    if(messageId != 0)
    {
        printfLog(EN_LOG_LEVEL_ERROR, "AgentLiteDemo: Test_propSetResponse() failed, messageId %d\n", messageId);
    }
```

## 4.5 直连设备上报设备属性查询结果

#### 接口功能

当网关收到平台下发的查询设备属性命令(5.5)后,可以调用该接口上报结果。

#### 接口描述

HW\_INT IOTA\_PropertiesGetResponse(HW\_CHAR \*requestId, ST\_IOTA\_SERVICE\_DATA\_INFO serviceProp[], HW\_INT serviceNum, void \*context);

#### 参数说明

字段	必选/可选	类型	描述
requestId	必选	HW_CHAR *	请求的唯一标识,设备收到的消息带该参数时,响应消息需要将 该参数值返回给平台。
serviceProp	可选	ST_IOTA_SER VICE_DATA_I NFO	设备属性结构体。 注意: 如果不携带该参数,需将 serviceNum设置为0
serviceNum	必选	HW_INT	要上报的服务个数
context	可选	void *	指向任何数据体的上下文指针, 该指针会在成功或者失败的回调 函数中返回。不使用该指针可以 传递NULL。

#### ST\_IOTA\_SERVICE\_DATA\_INFO 类型:

字段	必选/可选	类型	描述
service_id	必选	HW_CHAR *	设备服务的ID,可从profile中获取
event_time	可选	HW_CHAR *	设备采集数据UTC时间(格式: yyyyMMddTHHmmssZ),如: 20161219T114920Z。
			设备上报数据将参数设置为NULL时, 则数据上报时间以平台时间为准。
properties	必选	HW_CHAR *	一个服务的数据,具体字段在profile里 定义。注意:格式需能解析成JSON, 可参考下方的样例。

#### 接口返回值

#### 参见函数标准返回值

#### 示例

```
// 开发者调用该接口上报属性查询结果
int serviceNum = 1;
ST_IOTA_SERVICE_DATA_INFO serviceProp[serviceNum];
char *property = "{\"mno\":\"5\",\"imsi\":\"6\"}";
    //serviceProp[0].event_time = GetEventTimesStamp();
    serviceProp[0].event_time = NULL;
    serviceProp[0].service_id = "LTE";
    serviceProp[0].properties = property;

int messageId = IOTA_PropertiesGetResponse(requestId, serviceProp, serviceNum);
    if(messageId != 0)
    {
        printfLog(EN_LOG_LEVEL_ERROR, "AgentLiteDemo: Test_propGetResponse() failed, messageId %d\n",
        messageId);
    }
```

## 4.6 直连设备上报子设备状态

#### 接口功能

网关更新子设备状态。

#### 接口描述

HW\_INT IOTA\_UpdateSubDeviceStatus(ST\_IOTA\_DEVICE\_STATUSES \*device\_statuses, HW\_INT deviceNum, void \*context)

#### 参数说明

字段	必选/可选	类型	描述
device_statu ses	必选	ST_IOTA_DEVICE_ STATUSES*	上报的子设备状态信息列表
deviceNum	必选	HW_INT	上报的子设备个数
context	可选	void *	指向任何数据体的上下文指针,该指针会在成功或者失败的回调函数中返回。不使用该指针可以传递NULL。

#### ST\_IOTA\_SERVICE\_DATA\_INFO 类型:

字段	必选/可选	类型	描述
event_time	可选	String	事件时间

字段	必选/可选	类型	描述
device_stat uses[]	必选	ST_IOTA_DEVICE _STATUS	子设备状态的列表

#### ST\_IOTA\_DEVICE\_STATUS类型:

字段	必选/可选	类型	描述
device_id	必选	String	子设备ID
status	必选	String	子设备状态: OFFLINE:设备离线 ONLINE:设备上线;

#### 接口返回值

#### 参见函数标准返回值

#### 示例

```
// 开发者上报子设备状态
void Test_UpdateSubDeviceStatus(char *deviceId) {
    int deviceNum = 1;
    ST_IOTA_DEVICE_STATUSES device_statuses;
    device_statuses.event_time = NULL;
    device_statuses.device_statuses[0].device_id = deviceId;
    device_statuses.device_statuses[0].status = ONLINE;
    int messageId = IOTA_UpdateSubDeviceStatus(&device_statuses, deviceNum);
    if (messageId != 0) {
        PrintfLog(EN_LOG_LEVEL_ERROR, "device_demo: Test_UpdateSubDeviceStatus() failed, messageId %d
        \n", messageId);
    }
}
```

## 4.7 直连设备上报新增子设备请求

#### 接口功能

网关新增子设备。

#### 接口描述

HW\_INT IOTA\_AddSubDevice(ST\_IOTA\_SUB\_DEVICE\_INFO \*subDevicesInfo, HW\_INT deviceNum, void \*context)

#### 参数说明

字段	必选/可选	类型	描述
subDevicesIn fo	必选	ST_IOTA_SUB_DE VICE_INFO *	子设备信息结构体
deviceNum	必选	HW_INT	新增的子设备个数,单次最大 为50。有多个子设备时,建 议分多次添加。
context	可选	void *	指向任何数据体的上下文指针,该指针会在成功或者失败的回调函数中返回。不使用该指针可以传递NULL。

#### ST\_IOTA\_SUB\_DEVICE\_INFO 类型:

字段	必选/可选	类型	描述
event_time	可选	String	事件时间。不携带则平台采用平台 的UTC时间。默认传NULL。
event_id	可选	String	事件ID,用来标识该请求。不携带 着平台默认生成。默认传NULL
deviceInfo	必选	ST_IOTA_DEVICE _INFO	添加的子设备信息

#### ST\_IOTA\_DEVICE\_INFO类型:

字段名	必选/可选	类型	参数描述
parent_devic e_id	可选	String	父节点设备ID
node_id	必选	String	设备标识。
device_id	可选	String	设备ID
name	可选	String	设备名称
description	可选	String	设备描述
product_id	必选	String	产品ID
extension_in fo	可选	String	设备扩展信息。用户可以自定义任 何想要的扩展信息。字段值大小上 限为1K。

#### 接口返回值

#### 参见函数标准返回值

平台响应参数请参考5.8 接收网关新增子设备响应

#### 示例

```
// 开发者调用直连设备新增子设备接口
void Test_GtwAddSubDevice() {
  ST_IOTA_SUB_DEVICE_INFO subDeviceInfos;
  int deviceNum = 2;
     subDeviceInfos.deviceInfo[0].description = "description";
  subDeviceInfos.deviceInfo[0].device_id = "device_id123";
  subDeviceInfos.deviceInfo[0].extension_info = NULL;
  subDeviceInfos.deviceInfo[0].name = "sub_device111";
  subDeviceInfos.deviceInfo[0].node_id = "node_id123";
  subDeviceInfos.deviceInfo[0].parent_device_id = NULL;
  subDeviceInfos.deviceInfo[0].product_id = "your_product_id";
     subDeviceInfos.deviceInfo[1].description = "description";
  subDeviceInfos.deviceInfo[1].device_id = "device_id1234";
  subDeviceInfos.deviceInfo[1].extension_info = NULL;
  subDeviceInfos.deviceInfo[1].name = "sub_device222";
  subDeviceInfos.deviceInfo[1].node_id = "node_id123";
  subDeviceInfos.deviceInfo[1].parent_device_id = NULL;
  subDeviceInfos.deviceInfo[0].product_id = "your_product_id";
  subDeviceInfos.deviceInfo[1].product_id = "5f58768785edc002bc69cbf2";
  subDeviceInfos.event id = "123123";
  subDeviceInfos.event_time = NULL;
  int messageId = IOTA_AddSubDevice(&subDeviceInfos, deviceNum, NULL);
  if (messageId != 0) {
     PrintfLog(EN_LOG_LEVEL_ERROR, "device_demo: Test_GtwAddSubDevice() failed, messageId %d\n",
messageId);
  }
```

## 4.8 直连设备上报删除子设备请求

#### 接口功能

网关删除子设备。

#### 接口描述

HW\_INT IOTA\_DelSubDevice(ST\_IOTA\_DEL\_SUB\_DEVICE \*delSubDevices, HW\_INT deviceNum, void \*context)

#### 参数说明

字段	必选/可选	类型	描述
delSubDevic es	必选	ST_IOTA_DEL_SUB _DEVICE*	子设备信息结构体

字段	必选/可选	类型	描述
deviceNum	必选	HW_INT	删除的子设备个数,单次最大 为50。有多个子设备时,建 议分多次删除。
context	可选	void *	指向任何数据体的上下文指针,该指针会在成功或者失败的回调函数中返回。不使用该指针可以传递NULL。

#### ST\_IOTA\_DEL\_SUB\_DEVICE类型:

字段	必选/可选	类型	描述
event_time	可选	String	事件时间。不携带则平台采用平台 的UTC时间。默认传NULL。
event_id	可选	String	事件ID,用来标识该请求。不携带 着平台默认生成。默认传NULL
delSubDevi ce	必选	HW_CHAR **	删除的子设备ID列表

#### 接口返回值

#### 参见函数标准返回值

平台响应参数请参考5.9 接收网关删除子设备响应

```
// 开发者调用直连设备新增子设备接口
void Test_GtwDelSubDevice () {
    ST_IOTA_DEL_SUB_DEVICE delSubDevices;
    int deviceNum = 3;

    delSubDevices.event_id = NULL;
    delSubDevices.event_time = NULL;
    delSubDevices.delSubDevice[0] = "device_id123";
    delSubDevices.delSubDevice[1] = "device_id1234";
    delSubDevices.delSubDevice[2] = "device_id12345";

    int messageId = IOTA_DelSubDevice(&delSubDevices, deviceNum, NULL);
    if (messageId != 0) {
        PrintfLog(EN_LOG_LEVEL_ERROR, "device_demo: Test_GtwDelSubDevice() failed, messageId %d\n",
    messageId);
    }
}
```

## 5 直连设备命令接收

设备可以接受平台命令(SDK已自动实现相关TOPIC的订阅)。主要有如下命令:设备 消息下发、平台命令下发、平台设置设备属性、平台查询设备属性、平台通知网关新 增子设备、平台通知网关删除子设备等。

开发者可以通过提前设置回调函数,来对命令进行处理,回调函数的设置方法请参考 3.5.2。本章节主要讲命令**消息体**(回调函数的入参),及回调函数的实现。

#### □ 说明

- 下行函数的结构体由SDK自行释放,如需保存结构体中的内容,需在业务处理时进行拷贝。
- 消息体中的可选参数均由第三方决定是否下发。
- 5.1 接收消息下发
- 5.2 接收自定义topic消息
- 5.3 接收命令下发
- 5.4 接收平台设置设备属性
- 5.5 接收平台查询设备属性
- 5.6 接收平台新增子设备通知
- 5.7 接收平台删除子设备通知
- 5.8 接收网关新增子设备响应
- 5.9 接收网关删除子设备响应

## 5.1 接收消息下发

#### 命令描述

设备收到应用下发的透传消息,该消息平台不解析。

#### 消息体

EN\_IOTA\_MESSAGE结构体:

字段名	必选/可选	类型	参数描述
mqtt_msg_inf o	必选	EN_IOTA_MQTT_ MSG_INFO*	mqtt协议层信息
object_device _id	可选	String	消息对应的最终目标设备,没有 携带则表示目标设备即网关设 备。
name	可选	String	消息名称
id	可选	String	消息的唯一标识
content	必选	String	消息内容。

#### EN\_IOTA\_MQTT\_MSG\_INFO类型:

字段	必选/可选	类型	描述
context	必选	HW_VOID	指向任何应用程序特定上下文的指针。 上下文指针被传递给每个回调函数,以 提供对回调中的上下文信息的访问。当 前预留,默认为NULL。
messageId	必选	HW_INT	消息ID
code	必选	HW_INT	消息码(业务层通知该值默认为0)

#### 示例

```
//开发者实现消息下发处理
void HandleMessageDown (EN_IOTA_MESSAGE *rsp) {
    if (rsp == NULL) {
        return;
    }
    PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleMessageDown(), content %s\n", rsp->content);
    PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleMessageDown(), id %s\n", rsp->id);
    PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleMessageDown(), name %s\n", rsp->name);
    PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleMessageDown(), object_device_id %s\n", rsp->object_device_id);
}
//设置回调函数
IOTA_SetMessageCallback(HandleMessageDown);
```

## 5.2 接收自定义 topic 消息

设备侧需先订阅自定义topic,才能接收到自定义topic消息。

## 5.2.1 订阅自定义 topic

#### 接口功能

可以通过该接口自定义topic。

#### 接口描述

HW\_INT IOTA\_SubscribeUserTopic(HW\_CHAR \*topicParas)

#### 参数说明

字段名	必选/可选	类型	参数描述
topicParas	必选	HW_CHAR*	自定义topic参数,例如"devMsg" (前面不要加'/'或者特殊字符)

#### 接口返回值

参见**函数标准返回值** 

#### 示例

// 开发者调用该接口进行自定义topic订阅 IOTA\_SubscribeUserTopic("devMsg");

## 5.2.2 接收自定义 topic 消息

#### 命令描述

设备收到应用下发的自定义topic消息,该消息平台不解析。

#### 消息体

EN\_IOTA\_USER\_TOPIC\_MESSAGE结构体:

字段名	必选/可选	类型	参数描述
mqtt_msg_inf o	必选	EN_IOTA_MQTT_ MSG_INFO*	mqtt协议层信息
object_device _id	可选	String	消息对应的最终目标设备,没有 携带则表示目标设备即网关设 备。
name	可选	String	消息名称
id	可选	String	消息的唯一标识
content	必选	String	消息内容。
topic_para	必选	String	自定义topic参数,例如 "devMsg"

EN\_IOTA\_MQTT\_MSG\_INFO类型:

字段	必选/可选	类型	描述
context	必选	HW_VOID	指向任何应用程序特定上下文的指针。 上下文指针被传递给每个回调函数,以 提供对回调中的上下文信息的访问。当 前预留,默认为NULL。
messageld	必选	HW_INT	消息ID
code	必选	HW_INT	消息码(业务层通知该值默认为0)

#### 示例

```
//开发者实现自定义topic消息下发处理
void HandleUserTopicMessageDown(EN_IOTA_USER_TOPIC_MESSAGE *rsp) {
    if (rsp == NULL) {
        return;
    }
    PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleMessageDown(), topic_para %s\n", rsp->topic_para);
    PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleMessageDown(), content %s\n", rsp->content);
    PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleMessageDown(), id %s\n", rsp->id);
    PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleMessageDown(), name %s\n", rsp->name);
    PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleMessageDown(), object_device_id %s\n", rsp->object_device_id);
}
//设置回调函数
IOTA_SetUserTopicMsgCallback(HandleUserTopicMessageDown);
```

## 5.3 接收命令下发

#### 命令描述

用于平台向设备下发设备控制命令。

#### 消息体

#### EN\_IOTA\_COMMAND结构体:

字段名	必选/可 选	类型	参数描述
mqtt_msg_ info	必选	EN_IOTA_MQTT_M SG_INFO*	mqtt协议层信息
object_devi ce_id	可选	String	命令对应的目标设备ID
service_id	可选	String	设备的服务ID。
command_ name	可选	String	设备命令名称,在设备关联的产品 模型中定义。

字段名	必选/可 选	类型	参数描述
paras	必选	String	设备命令的执行参数,具体字段在 设备关联的产品模型中定义。
request_id	必选	String	请求的唯一标识,设备收到的消息 带该参数时,响应消息需要将该参 数值返回给平台。

#### EN\_IOTA\_MQTT\_MSG\_INFO类型:

字段	必选/可选	类型	描述
context	必选	HW_VOID	指向任何应用程序特定上下文的指针。 上下文指针被传递给每个回调函数,以 提供对回调中的上下文信息的访问。当 前预留,默认为NULL。
messageId	必选	HW_INT	消息ID
code	必选	HW_INT	消息码(业务层通知该值默认为0)

#### 示例

```
//开发者实现命令下发处理
void HandleCommandRequest(EN_IOTA_COMMAND *command) {
  if (command == NULL) {
    return;
  PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleCommandRequest(), messageId %d\n", command-
>mqtt_msg_info->messageId);
  PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleCommandRequest(), object_device_id %s\n",
command->object_device_id);
  PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleCommandRequest(), service_id %s\n", command-
>service_id);
  PrintfLog(EN LOG LEVEL INFO, "device demo: HandleCommandRequest(), command name %s\n",
command_scommand_name);
  PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleCommandRequest(), paras %s\n", command-
>paras);
  PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleCommandRequest(), request_id %s\n", command-
>request_id);
  Test_CommandResponse(command->request_id); //response command
//设置回调函数
IOTA_SetCmdCallback(HandleCommandRequest);
```

#### □ 说明

Test\_commandResponse(requestId)函数里实现了命令响应逻辑,详细请参考**4.3 直连设备上报** 命令执行结果。

## 5.4 接收平台设置设备属性

#### 命令描述

用于平台设置设备属性。

#### 消息体

#### EN\_IOTA\_PROPERTY\_SET结构体:

字段名	必选/ 可选	类型	参数描述
mqtt_msg _info	必选	EN_IOTA_MQTT_ MSG_INFO*	mqtt协议层信息
object_dev ice_id	可选	String	属性设置对应的目标设备ID
request_id	必选	String	请求的唯一标识,设备收到的消息带该 参数时,响应消息需要将该参数值返回 给平台。
services	必选	EN_IOTA_SERVICE _PROPERTY*	设备服务数据列表。
services_c ount	必选	int	设备服务数据列表的个数

#### EN\_IOTA\_SERVICE\_PROPERTY类型:

字段名	必选/可 选	类型	参数描述
service_id	必选	String	设备的服务ID。
propertie s	必选	String	设备服务的属性列表,具体字段在产品模型 里定义。

#### EN\_IOTA\_MQTT\_MSG\_INFO类型:

字	段	必选/可选	类型	描述
со	ntext	必选	HW_VOID	指向任何应用程序特定上下文的指针。 上下文指针被传递给每个回调函数,以 提供对回调中的上下文信息的访问。当 前预留,默认为NULL。

字段	必选/可选	类型	描述
messageId	必选	HW_INT	消息ID
code	必选	HW_INT	消息码(业务层通知该值默认为0)

```
//开发者实现设置设备属性命令处理
void HandlePropertiesSet (EN_IOTA_PROPERTY_SET *rsp) {
  if (rsp == NULL) {
     return:
  PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandlePropertiesSet(), messageId %d \n", rsp-
>mqtt_msg_info->messageId);
  PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandlePropertiesSet(), request_id %s \n", rsp-
>request_id);
  PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandlePropertiesSet(), object_device_id %s \n", rsp-
>object_device_id);
  int i = 0:
  while (rsp->services_count > 0) {
     PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandlePropertiesSet(), service_id %s \n", rsp-
>services[i].service_id);
     PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandlePropertiesSet(), properties %s \n", rsp-
>services[i].properties);
     rsp->services_count--;
     j++;
  Test_PropSetResponse(rsp->request_id); //response
//设置回调函数
IOTA_SetPropSetCallback(HandlePropertiesSet);
```

#### □□ 说明

Test\_propSetResponse(requestId)函数里实现了命令响应逻辑,详细请参考**4.4 直连设备上报设备属性设置结果**。

## 5.5 接收平台查询设备属性

### 命令描述

用于平台向设备查询属性信息。

#### 消息体

EN\_IOTA\_PROPERTY\_GET结构体:

字段名	必选/可 选	类型	参数描述
mqtt_ms g_info	必选	EN_IOTA_MQTT _MSG_INFO*	mqtt协议层信息

字段名	必选/可 选	类型	参数描述
object_de vice_id	可选	String	属性查询对应的目标设备ID,不设置默 认是查询网关设备。
service_id	可选	String	设备的服务ID,不设置默认查询全部 service。
request_i d	必选	String	请求的唯一标识,设备收到的消息带该 参数时,响应消息需要将该参数值返回 给平台。

#### EN\_IOTA\_MQTT\_MSG\_INFO类型:

字段	必选/可选	类型	描述
context	必选	HW_VOID	指向任何应用程序特定上下文的指针。 上下文指针被传递给每个回调函数,以 提供对回调中的上下文信息的访问。当 前预留,默认为NULL。
messageId	必选	HW_INT	消息ID
code	必选	HW_INT	消息码(业务层通知该值默认为0)

#### 示例

```
//开发者实现查询设备属性命令处理
void HandlePropertiesGet (EN_IOTA_PROPERTY_GET *rsp) {
    if (rsp == NULL) {
        return;
    }
    PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandlePropertiesSet(), messageId %d \n", rsp-
>mqtt_msg_info->messageId);
    PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandlePropertiesSet(), request_id %s \n", rsp-
>request_id);
    PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandlePropertiesSet(), object_device_id %s \n", rsp-
>object_device_id);

PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandlePropertiesSet(), service_id %s \n", rsp->service_id);

Test_PropGetResponse(rsp->request_id); //response
}

// 设置回调函数
IOTA_SetPropGetCallback(HandlePropertiesGet);
```

#### □ 说明

Test\_propGetResponse(requestId)函数里实现了命令响应逻辑,详细请参考**4.5 直连设备上报设备属性查询结果**。

## 5.6 接收平台新增子设备通知

#### 命令描述

平台将该新增的子设备列表信息通知给设备。

#### 消息体

#### EN\_IOTA\_EVENT结构体:

字段名	必选/可选	类型	参数描述
mqtt_msg_in fo	必选	EN_IOTA_MQTT_ MSG_INFO*	mqtt协议层信息
object_devic e_id	可选	String	事件对应的最终目标设备,没有 携带则表示目标设备即网关设备
services	可选	EN_IOTA_SERVICE _EVENT*	事件服务列表
services_cou nt	可选	int	事件服务列表的个数

#### EN\_IOTA\_MQTT\_MSG\_INFO类型:

字段	必选/可选	类型	描述
context	必选	HW_VOID	指向任何应用程序特定上下文的指针。 上下文指针被传递给每个回调函数,以 提供对回调中的上下文信息的访问。当 前预留,默认为NULL。
messageId	必选	HW_INT	消息ID
code	必选	HW_INT	消息码(业务层通知该值默认为0)

#### EN\_IOTA\_SERVICE\_EVENT类型:

字段名	必选/可选	类型	参数描述
service_id	必选	int	EN_IOTA_EVENT_SUB_DEVICE_MA NAGER(枚举值为0)
event_type	必选	int	EN_IOTA_EVENT_ADD_SUB_DEVIC E_NOTIFY(枚举值为0)
event_time	可选	String	事件时间

字段名	必选/可选	类型	参数描述
paras	必选	EN_IOTA_DEVI CE_PARAS*	子设备事件参数
ota_paras	可选	EN_IOTA_OTA _PARAS*	ota事件参数,此时为NULL
ntp_paras	可选	EN_IOTA_NTP _PARAS*	NTP事件参数,此时为NULL
gtw_add_de vice_paras	可选	EN_IOTA_GTW _ADD_DEVICE _PARAS	网关新增子设备事件,此时为NULL
gtw_del_devi ce_paras	可选	EN_IOTA_GTW _DEL_DEVICE_ PARAS	网关删除子设备事件,此时为NULL

#### EN\_IOTA\_DEVICE\_PARAS结构体:

字段名	必选/可选	类型	参数描述
devices	必选	EN_IOTA_DEVI CE_INFO*	设备列表
devices_coun t	必选	int	设备列表个数
version	必选	long long	子设备信息版本

#### EN\_IOTA\_DEVICE\_INFO定义表

字段名	必选/可选	类型	参数描述
parent_devic e_id	必选	String	父节点设备ID
node_id	必选	String	设备标识。
device_id	必选	String	设备ID
name	可选	String	设备名称
description	可选	String	设备描述
manufacture r_id	可选	String	厂商ID
model	可选	String	设备型号
product_id	可选	String	产品ID

字段名	必选/可选	类型	参数描述
fw_version	可选	String	固件版本
sw_version	可选	String	软件版本
status	可选	String	设备在线状态 ONLINE:设备在线 OFFLINE:设备离线
extension_in fo	可选	Object	设备扩展信息。用户可以自定义任 何想要的扩展信息

```
//设置回调函数
IOTA_SetCallback(EN_IOTA_CALLBACK_EVENT_DOWN, HandleEventsDown);
//开发者实现事件相关命令处理(接收子设备新增/删除、OTA升级通知)
void HandleEventsDown(EN_IOTA_EVENT *message){
    if (message == NULL) {
         return;
     PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleEventsDown(), messageId %d\n", message-
>mgtt msg info->messageId);
     PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleEventsDown(), services_count %d\n", message-
>services_count);
    PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleEventsDown(), object_device_id %s\n", message-
>object_device_id);
    int i = 0:
     while (message->services_count > 0) {
         printf("servie_id: %d \n", message->services[i].servie_id);
         printf("event_time: %s \n", message->services[i].event_time);
         printf("event_type: %d \n", message->services[i].event_type);
         //sub device manager
         if (message->services[i].servie_id == EN_IOTA_EVENT_SUB_DEVICE_MANAGER) {
              //if it is the platform inform the gateway to add or delete the sub device
              if(message -> services[i].event\_type == EN\_IOTA\_EVENT\_ADD\_SUB\_DEVICE\_NOTIFY \parallel message-properties and the properties of the properties of
>services[i].event_type == EN_IOTA_EVENT_DELETE_SUB_DEVICE_NOTIFY) {
                   printf("version: %lld \n", message->services[i].paras->version);
                   int i = 0:
                   while(message->services[i].paras->devices_count > 0) {
                        PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleEventsDown(), parent_device_id: %s
\n", message->services[i].paras->devices[j].parent_device_id);
                        PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleEventsDown(), device_id: %s \n",
message->services[i].paras->devices[j].device_id);
                        PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleEventsDown(), node_id: %s \n",
message->services[i].paras->devices[j].node_id);
                        //add a sub device
                        if (message->services[i].event_type == EN_IOTA_EVENT_ADD_SUB_DEVICE_NOTIFY) {
                             PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleEventsDown(), name: %s \n",
message->services[i].paras->devices[j].name);
                             PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleEventsDown(), manufacturer_id: %s
\n", message->services[i].paras->devices[j].manufacturer_id);
                             PrintfLog(EN LOG LEVEL INFO, "device demo: HandleEventsDown(), product id: %s \n",
message->services[i].paras->devices[j].product_id);
```

```
Test_UpdateSubDeviceStatus(message->services[i].paras->devices[j].device_id); //report
status of the sub device
                         Test_BatchPropertiesReport(message->services[i].paras->devices[j].device_id); //report data
of the sub device
                     } else if (message->services[i].event_type == EN_IOTA_EVENT_DELETE_SUB_DEVICE_NOTIFY)
{ //delete a sub device
                         PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleEventsDown(), the sub device is
deleted: %s\n", message->services[i].paras->devices[j].device_id);
                    j++;
                    message->services[i].paras->devices_count--;
            } else if (message->services[i].event_type == EN_IOTA_EVENT_ADD_SUB_DEVICE_RESPONSE) {
                int i = 0;
                while (message->services[i].gtw\_add\_device\_paras->successful\_devices\_count>0) \ \{ constant = 0 \ (a.c., b.c., b.
                     PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleEventsDown(), device_id: %s \n",
message->services[i].gtw_add_device_paras->successful_devices[j].device_id);
                     PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleEventsDown(), name: %s \n", message-
>services[i].gtw_add_device_paras->successful_devices[j].name);
                     message->services[i].gtw_add_device_paras->successful_devices_count--;
                j = 0;
                while(message->services[i].gtw_add_device_paras->failed_devices_count > 0) {
                     PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleEventsDown(), error_code: %s \n",
message->services[i].gtw_add_device_paras->failed_devices[j].error_code);
                    PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleEventsDown(), error_msg: %s \n",
message->services[i].gtw_add_device_paras->failed_devices[j].error_msg);
                     PrintfLog(EN LOG LEVEL INFO, "device demo: HandleEventsDown(), node id: %s \n",
message->services[i].gtw_add_device_paras->failed_devices[j].node_id);
                     PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleEventsDown(), product_id: %s \n",
message->services[i].gtw_add_device_paras->failed_devices[j].product_id);
                    j++;
                     message->services[i].gtw_add_device_paras->failed_devices_count--;
            } else if (message->services[i].event_type == EN_IOTA_EVENT_DEL_SUB_DEVICE_RESPONSE) {
                int j = 0;
                while(message->services[i].gtw_del_device_paras->successful_devices_count > 0) {
                    PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleEventsDown(), device_id: %s \n",
message->services[i].gtw_del_device_paras->successful_devices[j]);
                    j++;
                    message->services[i].gtw_del_device_paras->successful_devices_count--;
                i = 0;
                while(message->services[i].gtw_del_device_paras->failed_devices_count > 0) {
                     PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleEventsDown(), error_code: %s \n",
message->services[i].gtw_del_device_paras->failed_devices[j].error_code);
                     PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleEventsDown(), error_msg: %s \n",
message->services[i].gtw_del_device_paras->failed_devices[j].error_msg);
                     PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleEventsDown(), device_id: %s \n",
message->services[i].gtw_del_device_paras->failed_devices[j].device_id);
                     message->services[i].gtw_del_device_paras->failed_devices_count--;
            }
        } else if (message->services[i].servie_id == EN_IOTA_EVENT_OTA) {
            if (message->services[i].event_type == EN_IOTA_EVENT_VERSION_QUERY) {
                 //report OTA version
                 Test_ReportOTAVersion();
            }
            if (message->services[i].event_type == EN_IOTA_EVENT_FIRMWARE_UPGRADE || message-
>services[i].event type == EN_IOTA_EVENT_SOFTWARE_UPGRADE) {
```

```
//check md5
          char pkg_md5 = "yourMd5"; //the md5 value of your ota package
          if (strcmp(pkg_md5, message->services[i].ota_paras->sign)) {
             //report failed status
             Test_ReportUpgradeStatus(-1, message->services[i].ota_paras->version);
          //start to receive packages and firmware_upgrade or software_upgrade
          if (IOTA_GetOTAPackages(message->services[i].ota_paras->url, message->services[i].ota_paras-
>access_token, 1000) == 0) {
             usleep(3000 * 1000);
             //report successful upgrade status
             Test_ReportUpgradeStatus(0, message->services[i].ota_paras->version);
          } else {
             //report failed status
             Test_ReportUpgradeStatus(-1, message->services[i].ota_paras->version);
       }
     } else if (message->services[i].servie_id == EN_IOTA_EVENT_TIME_SYNC) {
       if(message->services[i].event_type == EN_IOTA_EVENT_GET_TIME_SYNC_RESPONSE) {
          PrintfLog(EN LOG LEVEL INFO, "device demo: HandleEventsDown(), device real time: %lld \n",
message->services[i].ntp_paras->device_real_time);
     } else if (message->services[i].servie_id == EN_IOTA_EVENT_DEVICE_LOG) {
       if(message->services[i].event_type == EN_IOTA_EVENT_LOG_CONFIG) {
          PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleEventsDown(), log_switch: %s \n",
message->services[i].device_log_paras->log_switch);
          PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleEventsDown(), end_time: %s \n", message-
>services[i].device_log_paras->end_time);
     i++:
     message->services_count--;
  }
```

#### □说明

- if (message->services[i].event\_type == EN\_IOTA\_EVENT\_ADD\_SUB\_DEVICE\_NOTIFY)判断是新增子设备事件通知,于是给子设备上报一条数据。
- if (message->services[i].event\_type == EN\_IOTA\_EVENT\_DELETE\_SUB\_DEVICE\_NOTIFY)
   判断是删除子设备事件通知。
- if (message->services[i].event\_type == EN\_IOTA\_EVENT\_ADD\_SUB\_DEVICE\_RESPONSE)判断是网关主动添加子设备响应
- if (message->services[i].event\_type == EN\_IOTA\_EVENT\_DEL\_SUB\_DEVICE\_RESPONSE)判断是网关主动删除子设备响应
- Test\_UpdateSubDeviceStatus函数实现了子设备状态上报功能,详细请参考4.6 直连设备上报子设备状态。
- Test\_BatchPropertiesReport函数里实现了子设备数据上报功能,详细请参考6.2 批量上报设备属性。

## 5.7 接收平台删除子设备通知

#### 命令描述

平台将该删除的子设备信息通知给设备。

#### 消息体

EN\_IOTA\_EVENT结构体:

字段名	必选/可选	类型	参数描述
mqtt_msg_in fo	必选	EN_IOTA_MQTT_ MSG_INFO*	mqtt协议层信息
object_devic e_id	可选	String	事件对应的最终目标设备,没有 携带则表示目标设备即网关设备
services	可选	EN_IOTA_SERVICE _EVENT*	事件服务列表
services_cou nt	可选	int	事件服务列表的个数

#### EN\_IOTA\_MQTT\_MSG\_INFO类型:

字段	必选/可选	类型	描述
context	必选	HW_VOID	指向任何应用程序特定上下文的指针。 上下文指针被传递给每个回调函数,以 提供对回调中的上下文信息的访问。当 前预留,默认为NULL。
messageld	必选	HW_INT	消息ID
code	必选	HW_INT	消息码(业务层通知该值默认为0)

#### EN\_IOTA\_SERVICE\_EVENT类型:

字段名	必选/可选	类型	参数描述
service_id	必选	int	EN_IOTA_EVENT_SUB_DEVICE_MA NAGER(枚举值为0)
event_type	必选	int	EN_IOTA_EVENT_DELETE_SUB_DE VICE_NOTIFY(枚举值为1)
event_time	可选	String	事件时间
paras	必选	EN_IOTA_DEVI CE_PARAS*	子设备事件参数
ota_paras	可选	EN_IOTA_OTA _PARAS*	ota事件参数,此时为NULL
ntp_paras	可选	EN_IOTA_NTP _PARAS*	NTP事件参数,此时为NULL
gtw_add_de vice_paras	可选	EN_IOTA_GTW _ADD_DEVICE _PARAS	网关新增子设备事件,此时为NULL

字段名	必选/可选	类型	参数描述
gtw_del_devi ce_paras	可选	EN_IOTA_GTW _DEL_DEVICE_ PARAS	网关删除子设备事件,此时为NULL

#### EN\_IOTA\_DEVICE\_PARAS结构体:

字段名	必选/可选	类型	参数描述
devices	必选	EN_IOTA_DEVI CE_INFO*	设备列表
devices_coun t	必选	int	设备列表个数
version	必选	long long	子设备信息版本

#### EN\_IOTA\_DEVICE\_PARAS结构体:

字段名	必选/可选	类型	参数描述
devices	必选	EN_IOTA_DEVI CE_INFO*	设备列表
devices_coun t	必选	int	设备列表个数
version	必选	long long	子设备信息版本

#### EN\_IOTA\_DEVICE\_INFO定义表

字段名	必选/可选	类型	参数描述
parent_devic e_id	必选	String	父节点设备ID
node_id	可选	String	设备标识。
device_id	必选	String	设备ID

#### 示例

请参考5.6 接收平台新增子设备通知示例。

#### 山 说明

新增子设备和删除子设备通知Topic一致,可通过消息体中的event\_type字段来进行判断。

## 5.8 接收网关新增子设备响应

#### 命令描述

平台下发网关主动添加的子设备响应。

### 消息体

#### EN\_IOTA\_EVENT结构体:

字段名	必选/可选	类型	参数描述
mqtt_msg_in fo	必选	EN_IOTA_MQTT_ MSG_INFO*	mqtt协议层信息
object_devic e_id	可选	String	事件对应的最终目标设备,没有 携带则表示目标设备即网关设备
services	可选	EN_IOTA_SERVICE _EVENT*	事件服务列表
services_cou nt	可选	int	事件服务列表的个数

#### EN\_IOTA\_MQTT\_MSG\_INFO类型:

字段	必选/可选	类型	描述
context	必选	HW_VOID	指向任何应用程序特定上下文的指针。 上下文指针被传递给每个回调函数,以 提供对回调中的上下文信息的访问。当 前预留,默认为NULL。
messageld	必选	HW_INT	消息ID
code	必选	HW_INT	消息码(业务层通知该值默认为0)

#### EN\_IOTA\_SERVICE\_EVENT类型:

字段名	必选/可选	类型	参数描述
service_id	必选	int	EN_IOTA_EVENT_SUB_DEVICE_MA NAGER(枚举值为0)
event_type	必选	int	EN_IOTA_EVENT_ADD_SUB_DEVIC E_RESPONSE(枚举值为6)
event_time	可选	String	事件时间

字段名	必选/可选	类型	参数描述
paras	必选	EN_IOTA_DEVI CE_PARAS*	子设备事件参数,此时为NULL
ota_paras	可选	EN_IOTA_OTA _PARAS*	ota事件参数,此时为NULL
ntp_paras	可选	EN_IOTA_NTP _PARAS*	NTP事件参数,此时为NULL
gtw_add_de vice_paras	必选	EN_IOTA_GTW _ADD_DEVICE _PARAS	网关新增子设备事件
gtw_del_devi ce_paras	可选	EN_IOTA_GTW _DEL_DEVICE_ PARAS	网关删除子设备事件,此时为NULL
device_log_p aras	可选	EN_IOTA_DEVI CE_LOG_PARA S*	设备日志事件参数,此时为NULL

#### EN\_IOTA\_GTW\_ADD\_DEVICE\_PARAS结构体:

字段名	必选/可 选	类型	参数描述
successful_devi ces	必选	EN_IOTA_DEVI CE_INFO*	成功新增的子设备列表详情
successful_devi ces_count	必选	int	成功添加的子设备个数
failed_devices	必选	EN_IOTA_ADD _DEVICE_FAIL ED_REASON*	新增子设备失败的原因
failed_devices_ count	必选	int	添加失败的子设备个数

#### EN\_IOTA\_DEVICE\_INFO 定义表

字段名	必选/可选	类型	参数描述
parent_devic e_id	必选	String	父节点设备ID
node_id	必选	String	设备标识。
device_id	必选	String	设备ID

字段名	必选/可选	类型	参数描述
name	可选	String	设备名称
description	可选	String	设备描述
manufacture r_id	可选	String	厂商ID
model	可选	String	设备型号
product_id	可选	String	产品ID
fw_version	可选	String	固件版本
sw_version	可选	String	软件版本
status	可选	String	设备在线状态 ONLINE:设备在线 OFFLINE:设备离线
extension_in fo	可选	Object	设备扩展信息。用户可以自定义任 何想要的扩展信息

#### EN\_IOTA\_ADD\_DEVICE\_FAILED\_REASON定义表

字段名	必选/可选	类型	参数描述
node_id	必选	String	对应请求中指定的设备的node_id
product_id	必选	String	对应请求中指定的设备的product_id
error_code	必选	String	新增失败错误原因码
error_msg	必选	String	新增失败原因描述

#### 示例

请参考5.6 接收平台新增子设备通知示例。

#### □ 说明

平台新增/删除子设备子设备通知Topic与网关新增/删除子设备响应的TOPIC一致,可通过消息体中的event\_type字段来进行判断。

## 5.9 接收网关删除子设备响应

#### 命令描述

平台下发网关主动删除的子设备响应。

#### 消息体

#### EN\_IOTA\_EVENT结构体:

字段名	必选/可选	类型	参数描述
mqtt_msg_in fo	必选	EN_IOTA_MQTT_ MSG_INFO*	mqtt协议层信息
object_devic e_id	可选	String	事件对应的最终目标设备,没有 携带则表示目标设备即网关设备
services	可选	EN_IOTA_SERVICE _EVENT*	事件服务列表
services_cou nt	可选	int	事件服务列表的个数

#### EN\_IOTA\_MQTT\_MSG\_INFO类型:

字段	必选/可选	类型	描述
context	必选	HW_VOID	指向任何应用程序特定上下文的指针。 上下文指针被传递给每个回调函数,以 提供对回调中的上下文信息的访问。当 前预留,默认为NULL。
messageId	必选	HW_INT	消息ID
code	必选	HW_INT	消息码(业务层通知该值默认为0)

#### EN\_IOTA\_SERVICE\_EVENT类型:

字段名	必选/可选	类型	参数描述
service_id	必选	int	EN_IOTA_EVENT_SUB_DEVICE_MA NAGER(枚举值为0)
event_type	必选	int	EN_IOTA_EVENT_DEL_SUB_DEVICE _RESPONSE(枚举值为7)
event_time	可选	String	事件时间
paras	必选	EN_IOTA_DEVI CE_PARAS*	子设备事件参数,此时为NULL
ota_paras	可选	EN_IOTA_OTA _PARAS*	ota事件参数,此时为NULL
ntp_paras	可选	EN_IOTA_NTP _PARAS*	NTP事件参数,此时为NULL

字段名	必选/可选	类型	参数描述
gtw_add_de vice_paras	必选	EN_IOTA_GTW _ADD_DEVICE _PARAS	网关新增子设备事件,此时为NULL
gtw_del_devi ce_paras	可选	EN_IOTA_GTW _DEL_DEVICE_ PARAS	网关删除子设备事件
device_log_p aras	可选	EN_IOTA_DEVI CE_LOG_PARA S*	设备日志事件参数,此时为NULL

#### EN\_IOTA\_GTW\_ADD\_DEVICE\_PARAS结构体:

字段名	必选/可 选	类型	参数描述
successful_devi ces	必选	HW_CHAR **	成功删除的子设备列表详情
successful_devi ces_count	必选	int	成功删除的子设备个数
failed_devices	必选	EN_IOTA_DEL_ DEVICE_FAILE D_REASON*	删除子设备失败的原因
failed_devices_ count	必选	int	删除失败的子设备个数

#### EN\_IOTA\_DEL\_DEVICE\_FAILED\_REASON定义表

字段名	必选/可选	类型	参数描述
device_id	必选	String	对应请求中指定的设备的device_id
error_code	必选	String	删除失败错误原因码
error_msg	必选	String	删除失败原因描述

#### 示例

#### 请参考5.6 接收平台新增子设备通知示例。

#### 山 说明

平台新增/删除子设备子设备通知Topic与网关新增/删除子设备响应的TOPIC一致,可通过消息体中的event\_type字段来进行判断。

# 6 子设备上报数据

- 6.1 子设备上报消息
- 6.2 批量上报设备属性

### 6.1 子设备上报消息

请参考**4.1 直连设备上报自定义topic消息**,将object\_device\_id设置为子设备Id,即可通过网关设备上报子设备消息。

## 6.2 批量上报设备属性

#### 接口功能

用于批量设备上报数据给平台。网关设备可以用此接口同时上报多个子设备的数据。

#### 接口描述

HW\_INT IOTA\_BatchPropertiesReport(ST\_IOTA\_DEVICE\_DATA\_INFO pDeviceData[], HW\_INT deviceNum, HW\_INT serviceLenList[], HW\_INT compressFlag, void \*context)

#### 参数说明

字段	必选/可选	类型	描述
pServiceDat a[]	必选	ST_IOTA_DEV ICE_DATA_IN FO	上报的设备数据结构体数组
deviceNum	必选	HW_INT	上报的子设备个数
serviceLenLis t[]	必选	HW_INT	子设备上报的服务个数数组

字段	必选/可选	类型	描述
compressFla g	必选	HW_INT	是否压缩上报,0代表非压缩上报,1代表压缩上报。压缩上报会消耗一定的内存,如果没有特定流量要求的话,建议使用非压缩上报。
context	可选	void *	指向任何数据体的上下文指针,该 指针会在成功或者失败的回调函数 中返回。不使用该指针可以传递 NULL。

#### ST\_IOTA\_SERVICE\_DATA\_INFO 类型:

字段	必选/可选	类型	描述
device_id	必选	HW_CHAR *	设备ID
services[Ma xServiceRep ortNum]	必选	ST_IOTA_SE RVICE_DATA _INFO	一个服务的数据,具体字段在profile里 定义。注意:格式需能解析成JSON, 可参考下方的示例。 MaxServiceReportNum表示一个设备 一次数据包含的最大服务个数,默认为 10,开发者可以自定义。

#### ST\_IOTA\_SERVICE\_DATA\_INFO 类型:

字段	必选/可选	类型	描述
service_id	必选	HW_CHAR *	设备服务的ID,可从profile中获取
event_time	可选	HW_CHAR *	设备采集数据UTC时间(格式: yyyyMMddTHHmmssZ),如: 20161219T114920Z。 设备上报数据将参数设置为NULL时, 则数据上报时间以平台时间为准。
properties	必选	HW_CHAR *	一个服务的数据,具体字段在profile里 定义。注意:格式需能解析成JSON, 可参考下方的样例。

#### 示例

// 开发者通过该接口批量上报设备属性

int deviceNum = 1; //要上报的子设备的个数 ST\_IOTA\_DEVICE\_DATA\_INFO devices[deviceNum]; //子设备要上报的结构体数组

int serviceList[deviceNum]; //对应存储每个子设备要上报的服务个数

```
char *device1_service1 = "{\"mno\":\"1\",\"imsi\":\"3\"}"; //service1要上报的属性数据,必须是json格式
  char *device1_service2 = "{\"hostCpuUsage\":\"2\",\"containerCpuUsage\":\"4\"}";//service2要上报的属性
数据,必须是json格式
  devices[0].device_id = subDeviceId;
  devices[0].services[0].event_time = GetEventTimesStamp();
  devices[0].service_id = "LTE";
  devices[0].services[0].properties = device1_service1;
  devices[0].services[1].event_time = GetEventTimesStamp();
  devices[0].services[1].service_id = "CPU";
  devices[0].services[1].properties = device1_service2;
// char *device2_service1 = "{\"AA\":\"2\",\"BB\":\"4\"}";
    devices[1].device_id = "subDevices22222";
devices[1].services[0].event_time = "d2s1";
// devices[1].services[0].service_id = "device2_service11111111";
    devices[1].services[0].properties = device2_service1;
  int messageId = IOTA_BatchPropertiesReport(devices, deviceNum, serviceList);
  if(messageId != 0)
     printfLog(EN_LOG_LEVEL_ERROR, "AgentLiteDemo: Test_batchPropertiesReport() failed, messageId %d
\n", messageld);
```

## **7** OTA 升级

- 7.1 平台下发获取版本信息通知
- 7.2 设备上报软固件版本
- 7.3 平台下发升级通知
- 7.4 设备请求下载包
- 7.5 设备上报升级状态

## 7.1 平台下发获取版本信息通知

#### 命令描述

平台下发获取版本信息通知。

#### 消息体

#### EN IOTA EVENT结构体:

字段名	必选/可选	类型	参数描述
mqtt_msg_in fo	必选	EN_IOTA_MQTT_ MSG_INFO*	mqtt协议层信息
object_devic e_id	可选	String	事件对应的最终目标设备,没有 携带则表示目标设备即网关设备
services	可选	EN_IOTA_SERVICE _EVENT*	事件服务列表
services_cou nt	可选	int	事件服务列表的个数

EN\_IOTA\_MQTT\_MSG\_INFO类型:

字段	必选/可选	类型	描述
context	必选	HW_VOID	指向任何应用程序特定上下文的指针。 上下文指针被传递给每个回调函数,以 提供对回调中的上下文信息的访问。当 前预留,默认为NULL。
messageld	必选	HW_INT	消息ID
code	必选	HW_INT	消息码(业务层通知该值默认为0)

#### EN\_IOTA\_SERVICE\_EVENT类型:

字段名	必选/可选	类型	参数描述
service_id	必选	int	EN_IOTA_EVENT_OTA(枚举值为 1)
event_type	必选	int	EN_IOTA_EVENT_VERSION_QUERY (枚举值为2)
event_time	可选	String	事件时间
paras	必选	EN_IOTA_DEVI CE_PARAS*	子设备事件参数,此时为NULL
ota_paras	可选	EN_IOTA_OTA _PARAS*	ota事件参数
ntp_paras	可选	EN_IOTA_NTP _PARAS*	NTP事件参数,此时为NULL
gtw_add_de vice_paras	可选	EN_IOTA_GTW _ADD_DEVICE _PARAS	网关新增子设备事件,此时为NULL
gtw_del_devi ce_paras	可选	EN_IOTA_GTW _DEL_DEVICE_ PARAS	网关删除子设备事件,此时为NULL
device_log_p aras	可选	EN_IOTA_DEVI CE_LOG_PARA S*	设备日志事件参数,此时为NULL

#### ota\_paras参数列表

字段名	必选/可选	类型	参数描述
-	-	-	-

请参考5.6 接收平台新增子设备通知示例。

## 7.2 设备上报软固件版本

#### 接口功能

可以通过该接口上报当前软固件版本号。

#### 接口描述

HW\_INT IOTA\_OTAVersionReport(ST\_IOTA\_OTA\_VERSION\_INFO otaVersionInfo, void \*context)

#### 参数说明

字段	必选/可选	类型	描述
otaVersionI nfo	必选	ST_IOTA_OTA_V ERSION_INFO	软固件版本数据结构体
context	可选	void *	指向任何数据体的上下文指针,该 指针会在成功或者失败的回调函数 中返回。不使用该指针可以传递 NULL。

#### ST\_IOTA\_OTA\_VERSION\_INFO 类型:

字段	必选/可选	类型	描述
sw_version	必选	HW_CHAR*	软件版本(和fw_version必须保证有一 个不为NULL)
fw_version	必选	HW_CHAR*	固件版本(和sw_version必须保证有一 个不为NULL)
event_time	可选	HW_CHAR*	设备采集数据UTC时间(格式: yyyyMMddTHHmmssZ),如: 20161219T114920Z。
			设备上报数据将参数设置为NULL时, 则数据上报时间以平台时间为准。
object_devi ce_id	可选	HW_CHAR*	事件对应的最终目标设备,没有携带则 表示目标设备即topic中指定的设备

#### 接口返回值

参见函数标准返回值

## 7.3 平台下发升级通知

#### 命令描述

平台下发升级通知。

#### 消息体

#### EN\_IOTA\_EVENT结构体:

字段名	必选/可选	类型	参数描述
mqtt_msg_in fo	必选	EN_IOTA_MQTT_ MSG_INFO*	mqtt协议层信息
object_devic e_id	可选	String	事件对应的最终目标设备,没有 携带则表示目标设备即网关设备
services	可选	EN_IOTA_SERVICE _EVENT*	事件服务列表
services_cou nt	可选	int	事件服务列表的个数

#### EN\_IOTA\_MQTT\_MSG\_INFO类型:

字段	必选/可选	类型	描述
context	必选	HW_VOID	指向任何应用程序特定上下文的指针。 上下文指针被传递给每个回调函数,以 提供对回调中的上下文信息的访问。当 前预留,默认为NULL。
messageld	必选	HW_INT	消息ID
code	必选	HW_INT	消息码(业务层通知该值默认为0)

#### EN\_IOTA\_SERVICE\_EVENT类型:

字段名	必选/可选	类型	参数描述
service_id	必选	int	EN_IOTA_EVENT_OTA(枚举值为 1)
event_type	必选	int	<ul> <li>EN_IOTA_EVENT_FIRMWARE_U PGRADE(枚举值为3,固件升级)</li> <li>EN_IOTA_EVENT_SOFTWARE_U PGRADE(枚举值为4,软件升级)</li> </ul>
event_time	可选	String	事件时间
paras	必选	EN_IOTA_DEVI CE_PARAS*	子设备事件参数,此时为NULL
ota_paras	可选	EN_IOTA_OTA _PARAS*	ota事件参数
ntp_paras	可选	EN_IOTA_NTP _PARAS*	NTP事件参数,此时为NULL
gtw_add_de vice_paras	可选	EN_IOTA_GTW _ADD_DEVICE _PARAS	网关新增子设备事件,此时为NULL
gtw_del_devi ce_paras	可选	EN_IOTA_GTW _DEL_DEVICE_ PARAS	网关删除子设备事件,此时为NULL
device_log_p aras	可选	EN_IOTA_DEVI CE_LOG_PARA S*	设备日志事件参数,此时为NULL

#### ota\_paras参数列表

字段名	必选/可选	类型	参数描述
version	必选	String	软固件包版本号
url	必选	String	软固件包下载地址
file_size	必选	Integer	软固件包文件大小
access_token	可选	String	软固件包url下载地址的临时token
expires	可选	Integer	access_token的超期时间
sign	必选	String	软固件包MD5值

请参考5.6 接收平台新增子设备通知示例。

### 7.4 设备请求下载包

#### 接口功能

网关可以该接口请求下载软/固件包,下载完毕后会在当前目录下生成一个软/固件包。

#### 接口描述

HW\_INT IOTA\_GetOTAPackages(HW\_CHAR \*url, HW\_CHAR \*token, HW\_INT timeout)

#### 参数说明

字段	必选/可选	类型	描述
url	必选	HW_CHAR*	软固件包下载地址
token	必选	HW_CHAR*	软固件包url下载地址的临时token
serviceNum	必选	timeout	请求下载包超时时间,值需要大于 300秒。建议小于24h。

#### 接口返回值

参见函数标准返回值

#### 示例

请参考5.6 接收平台新增子设备通知示例。

## 7.5 设备上报升级状态

#### 接口功能

当设备升级完成后,可以通过该接口上报升级状态。

#### 接口描述

HW\_INT IOTA\_OTAStatusReport(ST\_IOTA\_UPGRADE\_STATUS\_INFO otaStatusInfo, void \*context)

#### 参数说明

字段	必选/可选	类型	描述
otaStatusIn fo	必选	ST_IOTA_UPGRAD E_STATUS_INFO	升级状态数据结构体

字段	必选/可选	类型	描述
context	可选	void *	指向任何数据体的上下文指针,该 指针会在成功或者失败的回调函数 中返回。不使用该指针可以传递 NULL。

#### ST\_IOTA\_UPGRADE\_STATUS\_INFO 类型:

字段	必选/可选	类型	描述
result_code	必选	HW_INT	设备的升级状态,结果码定义如下:
			0 处理成功
			1 设备使用中
			2 信号质量差
			3 已经是最新版本
			4 电量不足
			5 剩余空间不足
			6 下载超时
			7 升级包校验失败
			8 升级包类型不支持
			9 内存不足
			10 安装升级包失败
			255 内部异常
progress	可选	HW_INT	设备的升级进度,范围: 0到100
event_time	可选	HW_CHAR*	设备采集数据UTC时间(格式: yyyyMMddTHHmmssZ),如: 20161219T114920Z。
			设备上报数据将参数设置为NULL时, 则数据上报时间以平台时间为准。
object_devi ce_id	可选	HW_CHAR*	事件对应的最终目标设备,没有携带则 表示目标设备即topic中指定的设备
description	可选	HW_CHAR*	升级状态描述信息,可以返回具体升级 失败原因。
version	必选	HW_CHAR*	设备当前版本号

#### 接口返回值

参见函数标准返回值

```
// 开发者调用该接口进行升级状态上报
void Test_ReportUpgradeStatus(int i, char *version) {
  ST_IOTA_UPGRADE_STATUS_INFO statusInfo;
  if (i == 0) {
     statusInfo.description = "success";
     statusInfo.progress = 100;
     statusInfo.result_code = 0;
     statusInfo.version = version;
  } else {
     statusInfo.description = "failed";
     statusInfo.result_code = 1;
     statusInfo.progress = 0;
     statusInfo.version = version;
  }
  statusInfo.event_time = NULL;
  statusInfo.object_device_id = NULL;
  int messageId = IOTA_OTAStatusReport(statusInfo);
  if (messageId != 0) {
    PrintfLog(EN_LOG_LEVEL_ERROR, "AgentLiteDemo: Test_ReportUpgradeStatus() failed, messageId %d
\n", messageld);
  }
```

## 8 设备影子

- 8.1 设备请求获取平台的设备影子数据
- 8.2 设备接收平台返回的设备影子数据

## 8.1 设备请求获取平台的设备影子数据

#### 接口功能

用于设备向平台获取设备影子数据。

#### 接口描述

HW\_INT IOTA\_GetDeviceShadow(HW\_CHAR \*requestId, HW\_CHAR \*object\_device\_id, HW\_CHAR \*service\_id, void \*context)

#### 参数说明

字段	必选/可选	类型	描述
requestId	必选	HW_CHAR*	请求的唯一标识,设备收到的消息带该 参数时,响应消息需要将该参数值返回 给平台。
service_id	可选	HW_CHAR*	需要获取设备影子的设备服务ID,不带的 话查询所有服务ID的设备影子数据。
object_devi ce_id	可选	HW_CHAR*	事件对应的最终目标设备,没有携带则 表示目标设备即topic中指定的设备
context	可选	void *	指向任何数据体的上下文指针,该指针 会在成功或者失败的回调函数中返回。 不使用该指针可以传递NULL。

#### 接口返回值

参见函数标准返回值

// 开发者调用该接口进行请求影子数据 //get device shadow IOTA\_GetDeviceShadow("1232", NULL, NULL);

## 8.2 设备接收平台返回的设备影子数据

#### 命令描述

用于接收平台返回的设备影子数据。

#### 消息体

#### EN\_IOTA\_EVENT结构体:

字段名	必选/可选	类型	参数描述
mqtt_msg_in fo	必选	EN_IOTA_MQTT_ MSG_INFO*	mqtt协议层信息
object_devic e_id	可选	String	设备影子的目标设备ID。
request_id	必选	String	请求的唯一标识,设备收到的消息带该参数时,响应消息需要将 该参数值返回给平台。
shadow	可选	EN_IOTA_SHADO W_DATA*	服务影子数据I列表
shadow_dat a_count	可选	int	服务影子列表的个数(默认为0)

#### EN\_IOTA\_MQTT\_MSG\_INFO类型:

字段	必选/可选	类型	描述
context	必选	HW_VOID	指向任何应用程序特定上下文的指针。 上下文指针被传递给每个回调函数,以 提供对回调中的上下文信息的访问。当 前预留,默认为NULL。
messageld	必选	HW_INT	消息ID
code	必选	HW_INT	消息码(业务层通知该值默认为0)

EN\_IOTA\_SHADOW\_DATA类型:

字段名	必选/可选	类型	参数描述
service_id	必选	String	设备的服务ID。
desired_prop erties	可选	String	设备影子desired区的属性列表,具 体字段在设备关联的产品模型里定 义。
desired_even t_time	可选	String	desired区设备属性数据的UTC时间 (格式: yyyyMMddTHHmmssZ),如: 20161219T114920Z。
reported_pro perties	可选	String	设备影子reported区的属性列表,具体字段在设备关联的产品模型里定义。
reported_eve nt_time	可选	String	reported区设备属性数据的UTC时间 (格式: yyyyMMddTHHmmssZ),如: 20161219T114920Z。
version	可选	Integer	设备影子版本信息

```
//开发者实现影子数据处理
void HandleDeviceShadowRsp(EN_IOTA_DEVICE_SHADOW *rsp) {
  if (rsp == NULL) {
    return;
  PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleDeviceShadowRsp(), messageId %d \n", rsp-
>mqtt_msg_info->messageId);
  PrintfLog(EN LOG LEVEL INFO, "device demo: HandleDeviceShadowRsp(), request id %s \n", rsp-
>request_id);
  PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleDeviceShadowRsp(), object_device_id %s \n", rsp-
>object_device_id);
  while (rsp->shadow_data_count > 0) {
    PrintfLog(EN LOG LEVEL INFO, "device demo: HandleDeviceShadowRsp(), service id %s \n", rsp-
>shadow[i].service_id);
    PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleDeviceShadowRsp(), desired properties %s \n",
rsp->shadow[i].desired_properties);
    PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleDeviceShadowRsp(), reported properties %s
\n", rsp->shadow[i].reported_properties);
    PrintfLog(EN_LOG_LEVEL_INFO, "device_demo: HandleDeviceShadowRsp(), version %d \n", rsp-
>shadow[i].version);
    rsp->shadow_data_count--;
    i++;
  }
//设置回调函数
IOTA_SetShadowGetCallback(HandleDeviceShadowRsp);
```

## 9 设备时间同步

- 9.1 设备上报时间同步请求
- 9.2 设备接收时间同步响应

## 9.1 设备上报时间同步请求

#### 接口功能

设备向平台发起时间同步请求。

#### 接口描述

HW\_API\_FUNC HW\_INT IOTA\_GetNTPTime(void \*context)

#### 参数说明

字段	必选/可选	类型	描述
context	可选	void *	指向任何数据体的上下文指针,该指针 会在成功或者失败的回调函数中返回。 不使用该指针可以传递NULL。

#### 接口返回值

参见函数标准返回值

#### 示例

// 开发者调用该接口进行时间同步请求 //NTP IOTA\_GetNTPTime(NULL);

## 9.2 设备接收时间同步响应

#### 命令描述

设备接收平台下发的时间同步响应,获取设备准确时间。

### 消息体

#### EN\_IOTA\_EVENT结构体:

字段名	必选/可选	类型	参数描述
mqtt_msg_in fo	必选	EN_IOTA_MQTT_ MSG_INFO*	mqtt协议层信息
object_devic e_id	可选	String	事件对应的最终目标设备,没有 携带则表示目标设备即网关设备
services	可选	EN_IOTA_SERVICE _EVENT*	事件服务列表
services_cou nt	可选	int	事件服务列表的个数

#### EN\_IOTA\_MQTT\_MSG\_INFO类型:

字段	必选/可选	类型	描述
context	必选	HW_VOID	指向任何应用程序特定上下文的指针。 上下文指针被传递给每个回调函数,以 提供对回调中的上下文信息的访问。当 前预留,默认为NULL。
messageId	必选	HW_INT	消息ID
code	必选	HW_INT	消息码(业务层通知该值默认为0)

#### EN\_IOTA\_SERVICE\_EVENT类型:

字段名	必选/可选	类型	参数描述
service_id	必选	int	EN_IOTA_EVENT_TIME_SYNC(枚 举值为2)
event_type	必选	int	EN_IOTA_EVENT_GET_TIME_SYNC_ RESPONSE(枚举值为5)
event_time	可选	String	事件时间

字段名	必选/可选	类型	参数描述
paras	必选	EN_IOTA_DEVI CE_PARAS*	子设备事件参数,此时为NULL
ota_paras	可选	EN_IOTA_OTA _PARAS*	ota事件参数,此时为NULL
ntp_paras	必选	EN_IOTA_NTP _PARAS*	NTP事件参数
gtw_add_de vice_paras	可选	EN_IOTA_GTW _ADD_DEVICE _PARAS	网关新增子设备事件,此时为NULL
gtw_del_devi ce_paras	可选	EN_IOTA_GTW _DEL_DEVICE_ PARAS	网关删除子设备事件,此时为NULL
device_log_p aras	可选	EN_IOTA_DEVI CE_LOG_PARA S*	设备日志事件参数,此时为NULL

#### EN\_IOTA\_NTP\_PARAS结构体:

字段名	必选/可 选	类型	参数描述
device_real_tim e	必选	HW_LLONG	设备真实时间

#### 示例

请参考5.6 接收平台新增子设备通知示例。

## 10设备信息上报

#### 10.1 设备信息上报

### 10.1 设备信息上报

#### 接口功能

设备向平台上报设备信息(设备连接上平台后,会自动上报一次携带sdk版本号的消息)。

#### 接口描述

HW\_API\_FUNC HW\_INT IOTA\_ReportDeviceInfo(ST\_IOTA\_DEVICE\_INFO\_REPORT \*device\_info\_report, void \*context)

#### 参数说明

字段	必选/可选	类型	描述
device_info _report	可选	ST_IOTA_DE VICE_INFO_ REPORT *	设备相关信息结构体
context	可选	void *	指向任何数据体的上下文指针,该指针 会在成功或者失败的回调函数中返回。 不使用该指针可以传递NULL。

#### ST\_IOTA\_DEVICE\_INFO\_REPORT 类型:

字段名	必选/可选	类型	参数描述
object_devic e_id	可选	String	事件对应的最终目标设备,没有携带则表示目标设备即topic中指定的设备

字段名	必选/可选	类型	参数描述
event_time	可选	String	事件时间
device_sdk_v ersion	可选	String	格式为:接入方式_版本号,例如 C_v0.5.0, JAVA_v0.5.0,Tiny SDK_v1.0.0等
sw_version	可选	String	软件版本
fw_version	可选	String	固件版本

#### 接口返回值

#### 参见函数标准返回值

#### 示例

```
// 开发者调用该接口上报设备信息
//report device log
long long timestamp = getTime();
char timeStampStr[14];
sprintf(timeStampStr,"%lld",timestamp);
IOTA_ReportDeviceLog("DEVICE_STATUS", "device log", timeStampStr, NULL);
```

- 11.1 平台下发日志收集通知
- 11.2 设备上报日志内容到平台

## 11.1 平台下发日志收集通知

#### 命令描述

用于平台下发日志收集通知给设备。

#### 消息体

EN\_IOTA\_EVENT结构体:

字段名	必选/可选	类型	参数描述
mqtt_msg_in fo	必选	EN_IOTA_MQTT_ MSG_INFO*	mqtt协议层信息
object_devic e_id	可选	String	事件对应的最终目标设备,没有 携带则表示目标设备即网关设备
services	可选	EN_IOTA_SERVICE _EVENT*	事件服务列表
services_cou nt	可选	int	事件服务列表的个数

EN\_IOTA\_MQTT\_MSG\_INFO类型:

字段	必选/可选	类型	描述
context	必选	HW_VOID	指向任何应用程序特定上下文的指针。 上下文指针被传递给每个回调函数,以 提供对回调中的上下文信息的访问。当 前预留,默认为NULL。
messageld	必选	HW_INT	消息ID
code	必选	HW_INT	消息码(业务层通知该值默认为0)

#### EN\_IOTA\_SERVICE\_EVENT类型:

字段名	必选/可选	类型	参数描述
service_id	必选	int	EN_IOTA_EVENT_DEVICE_LOG(枚 举值为3)
event_type	必选	int	EN_IOTA_EVENT_LOG_CONFIG (枚举值为8)
event_time	可选	String	事件时间
paras	可选	EN_IOTA_DEVI CE_PARAS*	子设备事件参数,此时为NULL
ota_paras	可选	EN_IOTA_OTA _PARAS*	ota事件参数,此时为NULL
ntp_paras	可选	EN_IOTA_NTP _PARAS*	NTP事件参数,此时为NULL
gtw_add_de vice_paras	可选	EN_IOTA_GTW _ADD_DEVICE _PARAS*	网关新增子设备事件,此时为NULL
gtw_del_devi ce_paras	可选	EN_IOTA_GTW _DEL_DEVICE_ PARAS*	网关删除子设备事件,此时为NULL
device_log_p aras	必选	EN_IOTA_DEVI CE_LOG_PARA S*	设备日志事件参数

EN\_IOTA\_DEVICE\_LOG\_PARAS结构体:

字段名	必选/可 选	类型	参数描述
log_switch	可选	String	设备侧日志收集开关 on: 开启设备侧日志收集功能 off: 关闭设备侧日志收集开关
end_time	可选	String	日志收集结束时间 yyyy-MM-dd'T'HH:mm:ss'Z'

请参考5.6 接收平台新增子设备通知示例。

## 11.2 设备上报日志内容到平台

#### 接口功能

日志收集开关开启时设备使用该接口向平台上报日志内容(在设备连接上平台后, SDK 默认自动上报一条登录成功日志到平台),最大不超过1MB。

#### 接口描述

 $\label{thm:char} \mbox{HW\_API\_FUNC HW\_INT IOTA\_ReportDeviceLog(HW\_CHAR *type, HW\_CHAR *content, HW\_CHAR *timestamp, void *context)} \\$ 

#### 参数说明

字段	必选/可选	类型	描述
type	必选	String	日志类型:
			DEVICE_STATUS: 设备状态
			DEVICE_PROPERTY: 设备属性
			DEVICE_MESSAGE: 设备消息
			DEVICE_COMMAND:设备命令
content	必选	String	日志内容。
timestamp	可选	String	日志产生时间戳,精确到毫秒
context	可选	void *	指向任何数据体的上下文指针,该指针 会在成功或者失败的回调函数中返回。 不使用该指针可以传递NULL。

#### 接口返回值

参见**函数标准返回值** 

```
// 开发者调用该接口进行时间同步请求
//report device log
long long timestamp = getTime();
char timeStampStr[14];
sprintf(timeStampStr,"%lld",timestamp);
IOTA_ReportDeviceLog("DEVICE_STATUS", "device log", timeStampStr, NULL);
```