设备接入Link Kit SDK

Python SDK

Python SDK

开发环境设置

操作系统要求

Python SDK在下面的操作系统上进行了验证，为了避免开发与运行时出错，请尽量选用与阿里一致的软件环境。

Linux	
Ubuntu 16.04 64-bit	
Windows	
Widows 7 64 bit	
Mac	
High Sierra	

Python版本要求

Python 3.6 版本

安装 python3.6

Linux

```
sudo add-apt-repository ppa:deadsnakes/ppa
sudo apt-get update
sudo apt-get install python3.6
wget https://bootstrap.pypa.io/get-pip.py
sudo python3.6 get-pip.py
python3.6 -m pip install --upgrade pip setuptools wheel
sudo apt-get install python3.6-venv
```

Mac

https://www.python.org/ftp/python/3.6.7/python-3.6.7-macosx10.9.pkg双击安装

windows

根据系统位宽选择安装下面的python文件：

- 32-bit

https://www.python.org/ftp/python/3.6.7/python-3.6.7.exe

- 64-bit:

https://www.python.org/ftp/python/3.6.7/python-3.6.7-amd64.exe

环境配置

创建和激活 VirtualEnvironments

Windows

```
mkdir work_dir
cd work_dir
python3 -m venv test_env
test_env\Scripts\activate.bat
```

Linux

```
mkdir work_dir
cd work_dir
python3 -m venv test_env
source test_env/bin/activate
```

Mac

```
mkdir work_dir
cd work_dir
python3 -m venv test_env
source test_env/bin/activate
```

自动安装linkkit

使用pip来安装linkkit最新版本

```
pip install aliyun-iot-linkkit
```

手动安装paho和linkkit

Link Kit SDK需要使用到开源的MQTT库，点击获取开源MQTT库paho。

点击获取最新版本的Python Link Kit SDK。点击获取exmaple示例代码。

以下以1.1.0版本为例，实际运行时请替换为最新版本。

将 aliyun-iot-linkkit-1.1.0.tar.gz和paho-mqtt-1.4.0.tar.gz 放到work_dir：

Linux

```
tar zxvf paho-mqtt-1.4.0.tar.gz
cd paho-mqtt-1.4.0
python3 setup.py install
cd ..

tar zxvf aliyun-iot-linkkit-1.1.0.tar.gz
cd aliyun-iot-linkkit-1.1.0
python3 setup.py install
cd ..
```

Mac

```
tar zxvf paho-mqtt-1.4.0.tar.gz
cd paho-mqtt-1.4.0
python3 setup.py install
cd ..

tar zxvf aliyun-iot-linkkit-1.1.0.tar.gz
cd aliyun-iot-linkkit-1.1.0
python3 setup.py install
```

```
cd ..
```

Windows

```
解压paho-mqtt-1.4.0.tar.gz
cd paho-mqtt-1.4.0
python setup.py install
cd ..

解压aliyun-iot-linkkit-1.1.0.tar.gz
cd aliyun-iot-linkkit-1.1.0
python setup.py install
```

认证与连接

本文介绍如何初始化设备信息，建立设备与云端的连接

云端域名

阿里云IoT在多个国家与地区部署了服务器，厂商可设置设备需要连接的域名，点击此处了解可用的地域中列出的域名，默认地域为上海。

设备认证

设备的身份认证支持两种方法，不同方法需填写不同信息。

> 若使用一机一密认证方式，需要指定host_name、product_key、device_name和device_secret
>
> 若使用一型一密认证方式，需要制定host_name、product_key、device_name和product_secret，需要设置动态注册回调接口on_device_dynamic_register

> 说明 若使用一型一密认证方式，初始化过程中需设置一型一密动态注册接口。并在控制台开启动态注册。host_name为region信息，各区域信息可查询 https://help.aliyun.com/document_detail/40654.html

- 一机一密

```
from linkkit import linkkit
```

```
lk = linkkit.LinkKit(
host_name="cn-shanghai",
product_key="xxxxxxxxxxx",
device_name="nnnnnnnn",
device_secret="sssssssssssssssssssssssssssssss")
```

如果需要改变MQTT连接的一些默认参数，可以通过config_mqtt 指定端口等连接参数，如下所示：

```
config_mqtt(self, port=1883, protocol="MQTTv311", transport="TCP",
secure="TLS", keep_alive=60, clean_session=True,
max_inflight_message=20, max_queued_message=0,
auto_reconnect_min_sec=1,
auto_reconnect_max_sec=60,
cadata=None):
```

上面的代码示例中设置了端口号、安全协议、保活时间等参数。

- 一型一密

若用户选用一型一密认证方式，首先需要动态注册过程根据ProductKey、DeviceName和ProductSecret去获取DeviceSecret，然后将DeviceSecret保存下来之后再使用一机一密方式进行设备连接。

下面是动态注册的代码示例：

```
from linkkit import linkkit

lk = linkkit.LinkKit(
host_name="cn-shanghai",
product_key="xxxxxxxxxxx",
device_name="nnnnnnnn",
device_secret="",
product_secret="yyyyyyyyyyyyyyyy")
lk.on_device_dynamic_register = on_device_dynamic_register
def on_device_dynamic_register(rc, value, userdata):
if rc == 0:
print("dynamic register device success, rc:%d, value:%s" % (rc, value))
else:
print("dynamic register device fail,rc:%d, value:%s" % (rc, value))
```

当rc值为0时，动态注册成功, value 为从云端收到的deviceSecret, 需要用户将收到的device_secret存储下来。

注：

- 如果设备已经通过动态注册方式获取到了deviceSecret，后续不能再继续使用动态注册方式去获取deviceSecret，而必须使用已获取到的deviceSecret使用一机一密方式连接阿里云IoT物联网。因此开发者需要保证deviceSecret存储的持久化，不能因为设备重启、重新安装导致deviceSecret丢失。

回调函数

设备连接云端成功后会通过on_connect回调函数通知用户，连接成功以后如果连接断开会通过on_disconnect回调通知用户，用户可以在回调中加入自己的业务处理逻辑。

```
lk.on_connect = on_connect
lk.on_disconnect = on_disconnect
def on_connect(session_flag, rc, userdata):
print("on_connect:%d,rc:%d,userdata:" % (session_flag, rc))
pass
def on_disconnect(rc, userdata):
print("on_disconnect:rc:%d,userdata:" % rc)
```

注：

- 当设备连接到阿里云IoT物联网后，如果因为网络原因连接断开，SDK会自动尝试连接阿里云IoT物联网，用户无需调用API

配置网络接口信息

如果产品生产时错误的将一个三元组烧写到了多个设备，多个设备将会被物联网平台认为是同一个设备，从而出现一个设备上线将另外一个设备的连接断开的情况。用户可以将自己的接口信息上传到云端，那么云端可以通过接口的信息来进行问题定位。

```
lk.config_device_info("Eth|03ACDEFF0032|Eth|03ACDEFF0031")
```

其中接口可取值：

- WiFi
- Eth
- Cellular

如果设备的上行网络接口是WiFi或者Eth（以太网），那么接口会有MAC地址，MAC地址的格式为全大写；

如果是Cellular（即2G、3G、4G蜂窝网接口），那么需要填入的接口数据为：

- IMEI：string
- ICCID: string
- IMSI：string
- MSISDN：string

填入信息格式举例："Cellular|imei_001122|iccid_22334455|imsi_234241|msisdn_53212"

启动连接

在MQTT连接参数配置（可选），回调函数设置（必选），网络接口信息（可选）操作完成后，需要使用connect_async 调用开始进行实际的连接。

```
```

```
lk.connect_async()
```

注：调用该函数之后如果因为网络处于连接断开状态导致连接失败，用户无需再次调用connect_async()、SDK会再次尝试连接云端。

自定义MQTT Topic通信

本文介绍如何直接基于MQTT Topic向云端发送消息，以及从云端接收消息。

从云端接收消息

订阅云端消息

```
rc, mid = lk.subscribe_topic(lk.to_full_topic("user/test"))
```

注：上面的代码示例中lk为调用linkkit.LinkKit()后返回的示例。

订阅结果通过on_subscribe_topic通知用户：

```
lk.on_subscribe_topic = on_subscribe_topic
def on_subscribe_topic(mid, granted_qos, userdata):
print("on_subscribe_topic mid:%d, granted_qos:%s" %
(mid, str(','.join('%s' % it for it in granted_qos))))
pass
```

granded_qos 为订阅topic列表对应的qos返回结果，正常值为0或1，128表示订阅失败

接收与处理来自云端的消息

通过on_topic_message()回调告知用户

```
lk.on_topic_message = on_topic_message
def on_topic_message(topic, payload, qos, userdata):
print("on_topic_message:" + topic + " payload:" + str(payload) + " qos:" + str(qos))
pass
```

发送消息到云端

发送消息

通过调用publish_topic()实现将消息发送到云端：

```
rc, mid = lk.publish_topic(lk.to_full_topic("user/pub"), "123")
```

发布消息结果通知

消息发送后，云端是否成功接收通过on_publish_topic回调通知用户：

```
lk.on_publish_topic = on_publish_topic
def on_publish_topic(mid, userdata):
print("on_publish_topic mid:%d" % mid)
```

注：publish_topic rc返回值为0则表明已经写入到了发送缓冲区，回调on_publish_topic 表明publish成功

取消消息订阅

通过调用unsubscribe_topic()取消对指定topic消息的订阅：

```
rc, mid = lk.unsubscribe_topic(lk.to_full_topic("user/test"))
```

取消订阅的结果通过on_unsubscribe_topic回调通知用户：

```
lk.on_unsubscribe_topic = on_unsubscribe_topic
def on_unsubscribe_topic(mid, userdata):
print("on_unsubscribe_topic mid:%d" % mid)
pass
```

unsubscribe_topic 返回值rc 为0表明请求已写入缓存区，其它值失败。当回调on_unsubscribe_topic时表明取消成功

物模型开发

设备可以使用物模型功能，实现属性上报、事件上报和服务调用

配置物模型文件

用户需要从云端控制台下载物模型文件，该文件需要集成到应用工程中，这样对应的topic才能正确的接收和发

送消息

```
lk.thing_setup("tsl.json")
```

注：该设置需要在连接云端之前调用。

物模型功能可用时通过on_thing_enable通知用户，然后用户可以进行属性上报，事件上报，服务响应：

```
lk.on_thing_enable = on_thing_enable
...
def on_thing_enable(self, userdata):
print("on_thing_enable")
```

物模型功能不可用时通过 on_thing_disable 通知用户，属性上报，事件上报，服务响应不可用

```
lk.on_thing_disable = on_thing_disable
...
def on_thing_disable(self, userdata):
print("on_thing_disable")
```

属性上报

通过thing_post_property上报属性，传入参数为与物模型定义中属性对应的字典对象

```
prop_data = {
"abs_speed": 11,
"power_stage": 10
}
rc, request_id = lk.thing_post_property(prop_data)
```

服务端对上报的属性做出处理发出响应，SDK通过on_thing_prop_post 通知用户

```
lk.on_thing_prop_post = on_thing_prop_post
...
def on_thing_prop_post(self, request_id, code, data, message,userdata):
print("on_thing_prop_post request id:%s, code:%d, data:%s message:%s" %
(request_id, code, str(data), message))
```

thing_post_property返回值rc为0时表明请求写入发送缓冲区成功，rc为其它值为写入发送缓冲失败，当on_thing_prop_post 调用时，表明结果从云端返回了请求结果，code为200时表明解析成功，其它值为失败，失败信息可在message中查看

事件上报

通过thing_trigger_event上报事件，传入参数为事件identifier和物模型中定义事件对应的字典对象，如下所示：

```
event_data = {
"power": 10,
"power_style": 1
}
rc, request_id = lk.thing_trigger_event(("power_state", event_data))
```

服务端对上报的事件处理后发出响应，SDK通过on_thing_event_post通知用户

```
lk.on_thing_event_post = on_thing_event_post
...
def on_thing_event_post(self, event, request_id, code, data, message, userdata):
print("on_thing_event_post event:%s,request id:%s, code:%d, data:%s, message:%s" %
(event, request_id, code, str(data), message))
```

设置属性

服务端发送设置属性消息后，SDK通过设置的回调函数on_thing_prop_changed通知用户，回调函数中params为包含属性名与值的字典对象，用户需要对接收到的新属性进行处理：

```
lk.on_thing_prop_changed = on_thing_prop_changed
...
def on_thing_prop_changed(self, params, userdata):
print("on_thing_prop_changed params:" + str(params))
```

注: SDK不会主动上报属性变化，如需要修改后再次上报云端，需要用户自行调用thing_post_property()发送

服务响应（异步）

服务端发送服务请求消息后，SDK通过设置的回调函数on_thing_call_service通知用户：

注：所有的服务均通过on_thing_call_service进行通知用户，通过identifier区分不同服务

```
lk.on_thing_call_service = on_thing_call_service
...
def on_thing_call_service(self, identifier, request_id, params, userdata):
print("on_thing_call_service identifier:%s, request id:%s, params:%s" %
(identifier, request_id, params))
```

identifier 对应物模型中服务对应的identifier, request_id为区分每次调用的id，params为服务调用参数

设备端对服务做出返回响应

```
lk.thing_answer_service(identifier, request_id, code, params)
```

identifer为物模型中服务对应的identifier,request_id为on_thing_call_service中传递的reques_id，code为返回码，200为本地处理成功，params为字典类型的参数，和物模型中服务返回参数进行对应

物模型-自定义协议

如果想使用自定义格式传输数据，则将物模型文件参数为空即可

```
lk.thing_setup()
```

设备上行数据，通过thing_raw_post_data向服务端发送自定义格式数据

```
params = {
"prop_int16": 11
}
payload = self.protocolToRawData(params)
lk.thing_raw_post_data(payload)
```

服务端收到自定义格式数据后给出回复,SDK通过on_thing_raw_data_post通知用户

```
lk.on_thing_raw_data_post = on_thing_raw_data_post
def on_thing_raw_data_post(self, payload, userdata):
print("on_thing_raw_data_post: %s" % str(payload))
```

服务端发送数据后，SDK通过on_thing_raw_data_arrived通知用户

```
lk.on_thing_raw_data_arrived = on_thing_raw_data_arrived
def on_thing_raw_data_arrived(self, payload, userdata):
print("on_thing_raw_data_arrived:%r" % payload)
print("prop data:%r" % self.rawDataToProtocol(payload))
```

设备标签

功能介绍

物联网平台的设备标签是给设备添加自定义的标识。您可以使用标签功能来灵活管理产品、设备和分组。

设备标签的结构为键值对, Key:Value。

您可以根据设备的特性为设备添加特有的标签，方便对设备进行管理。例如，为房间 201 的智能电表定义一个标签为room:201。您可以在控制台管理设备标签，也可以通过Python SDK API 管理设备标签。

关于标签的更多介绍详见 **用户指南/标签** 章节。

SDK使用

版本需求

Aliyun IoT Python SDK version >= 1.0.1

更新标签

通过thing_update_tags的接口可以添加以及更新标签，范例

```
tags = {
"floor": "2f",
"room": "201"
}
rc, request_id = linkkit.thing_update_tags(tags)
if rc == 0:
printf("success")
```

该接口主要异步化向云平台提交一个更新标签的请求，返回rc, request_id。rc为0标识成功，request_id为提交的请求id, 可以在异步的回调函数中关联该id获取最终的结果。

执行结果将异步返回，可以通过设置回调函数on_thing_device_info_update 获得相关结果，范例如下:

```
linkkit.on_thing_device_info_update = on_thing_device_info_update

...

def on_thing_device_info_update(self, request_id, code, data, message, userdata):
print("on_thing_device_info_update: request_id:%s, code:%s, data:%s, message:%s" % (request_id, code, data,
message))
```

标签更新成功可以在控制台查看

删除标签

通过thing_remove_tags的接口可以添加以及更新标签，范例

```
tags = ["floor", "room"]
rc, request_id = linkkit.thing_remove_tags(tags)
if rc == 0:
printf("success")
```

该接口主要异步化向云平台提交一个删除标签的请求，返回rc, request_id。rc为0标识成功，request_id为提交的请求id, 可以在异步的回调函数中关联该id获取最终的结果。

执行结果将异步返回，可以通过设置回调函数on_thing_device_info_delete 获得相关结果，范例如下:

```
linkkit.on_thing_device_info_delete = on_thing_device_info_delete

...

def on_thing_device_info_delete(self, request_id, code, data, message, userdata):
print("on_thing_device_info_delete: request_id:%s, code:%s, data:%s, message:%s" % (request_id, code, data,
message))
```

设备影子

功能介绍

如果开启高级版，推荐使用高级版物模型能力，已经提供了更完整的能力用于替代**设备影子**功能。

设备影子是一个 JSON 文档，用于存储设备上报状态、应用程序期望状态信息。

- 每个设备有且只有一个设备影子，设备可以通过MQTT获取和设置设备影子以此来同步状态，该同步可以是影子同步给设备，也可以是设备同步给影子。
- 应用程序通过物联网平台的SDK获取和设置设备影子，获取设备最新状态或者下发期望状态给设备。

具体影子的详细介绍见**物联网平台/设备端开发指南/设备影子**章节，
https://help.aliyun.com/document_detail/53930.html

SDK使用

版本需求

Aliyun IoT Python SDK version >= 1.1.0

主动更新影子

通过thing_update_shadow 的接口可以更新影子状态

```
reported = {"color":"red"}
# reported - 上报的影子数据
# version - 影子数据的版本号，例子中为1
```

```
res = linkkit.thing_update_shadow(reported, 1)
if res == 0:
print('success')
```

该接口主要异步化向云平台上报影子数据。

执行结果将异步返回，可以通过设置回调函数on_thing_shadow_get 获得相关结果，范例如下:

```
linkkit.on_thing_shadow_get = on_thing_shadow_get

...

def on_thing_shadow_get(self, payload, userdata):
print("on_thing_shadow_get:", payload)
```

如果影子设置成功，回调中payload对象数据如下:

```
{
"method": "reply",
"payload": {
"status": "success",
"version": 1
},
"timestamp": 1544686266
}
```

查询影子数据

通过thing_get_shadow的接口可以查询最新的影子数据，范例如下：

```
res = linkkit.thing_get_shadow()
if res == 0:
print('success')
```

该接口主要异步化向云平台提交一个查询影子的请求，返回rc。rc为0标识请求成功， 可以在异步的回调函数中关联该id获取最终的结果。

执行结果将异步返回，可以通过设置回调函数on_thing_shadow_get 获得相关结果，范例如下:

```
linkkit.on_thing_shadow_get = on_thing_shadow_get

...

def on_thing_shadow_get(self, payload, userdata):
print("on_thing_shadow_get:", payload)
```

对于正确的get操作，异步返回的on_thing_shadow_get回调中payload对象数据范例如下:

```
{
"method": "reply",
"payload": {
"status": "success",
"state": {
"reported": {
"color": "red"
}
},
"metadata": {
"reported": {
"color": {
"timestamp": 1544701176
}
}
}
},
"timestamp": 1544784614,
"version": 1
}
```

为了更方便的获取影子信息，SDK也会本地缓存一份/shadow/get/{pk}/{dn} topic里面的影子数据，可以通过**local_get_latest_shadow**来读取。

监听影子变更

影子的作用是允许云端去更新影子desired状态，设备端可以通过设置回调函数on_thing_shadow_get 获得desired状态的变更执行结果将异步返回，可以通过设置回调函数on_thing_shadow_get 获得相关结果，范例如下:

```
linkkit.on_thing_shadow_get = on_thing_shadow_get

...

def on_thing_shadow_get(self, payload, userdata):
print("on_thing_shadow_get:", payload)
```

对于desired的变更, on_thing_shadow_get回调中payload对象数据范例如下:

```
{
"method": "control",
"payload": {
"status": "success",
"state": {
"reported": {
"color": "red"
},
"desired": {
```

```
"color": "green"
}
},
"metadata": {
"reported": {
"color": {
"timestamp": 1544701176
}
},
"desired": {
"color": {
"timestamp": 1544702121
}
}
}
},
"timestamp": 1544702121,
"version": 3
}
```


RRPC能力

功能介绍

MQTT协议是基于PUB/SUB的异步通信模式，这种通讯模型不适用于服务端同步控制设备端返回结果的场景。物联网平台基于MQTT协议制定了一套请求和响应的同步机制，无需改动MQTT协议即可实现同步通信。物联网平台提供API给服务端，设备端只需要按照固定的格式回复PUB消息，服务端使用API，即可同步获取设备端的响应结果。

详细关于RRPC的介绍请见阿里云物联网平台文档 **用户指南/RRPC** 章节。
https://help.aliyun.com/document_detail/90567.html

RRPC是指客户云端通过云端API发起一个RRPC调用，该调用将同步返回设备的响应。设备端会收到一个同步请求的topic, 格式如/ext/rrpc/{messageId}/{rrpc_topic}, 设备端接收到该消息后，进行处理，并将处理结果以Message的方式publish到/ext/rrpc/{messageId}/{rrpc_topic}。Python的SDK已经提供了相应的细节封装。

云端有两种场景会涉及到RRPC的调用

- 消息通信API - RRpc (https://help.aliyun.com/document_detail/69797.html)

> 该API会发送一个RRPC请求，需要在设备端实现RRPC调用。

- 设备管理API - InvokeThingsService (https://help.aliyun.com/document_detail/96242.html)

> 如果使用高级版并登记为同步类型的服务，调用其服务时会采用RRpc模式。

SDK使用

版本需求

Aliyun IoT Python SDK version >= 1.1.0

RRPC使用 - 普通RRPC Topic

通过设置on_topic_rrpc_message 的回调来处理RRPC的Topic请求

```
linkkit.on_topic_rrpc_message = on_topic_rrpc_message

...

def on_topic_rrpc_message(self, id, topic, payload, qos, userdata):
print("on_topic_rrpc_message: id:%s, topic:%s, payload:%s" % (id, topic, payload))
self.linkkit.thing_answer_rrpc(id, payload)
```

所有的RRPC请求处理完成后，必须通过 thing_answer_rrpc 进行回应，id为rrpc请求的id, payload为返回报文的payload。

针对此类的RRPC，云端SDK可以通过Rrpc接口进行调用，并获得同步的返回结果

RRPC使用 - 物模型服务

通过设置on_thing_call_service 的回调来处理同步类型的service请求

```
linkkit.on_thing_call_service = on_thing_call_service

...

def on_thing_call_service(self, identifier, request_id, params, userdata):
print("on_thing_call_service: identifier:%s, request_id:%s, params:%s" % (identifier, request_id, params))
...
self.linkkit.thing_answer_service(identifier, request_id, 200, {})
```

所有的service请求处理完成后，必须通过 thing_answer_service 进行回应，request_id为请求的request_id。

针对此类的Service，云端SDK可以通过InvokeThingsService接口进行调用，并获得同步的返回结果

API列表

Init & Config

enable_logger

```
enable_logger(level)
```

Description:

enable linkkit sdk internal log

Available Working State:

- LinkKit.LinkKitState.INITIALIZED

Input Parameter

level

type:logging.CRITICALlogging.FATALlogging.ERRORlogging.WARNINGlogging.WARNlogging.INFOlogging.DEBUG

Return Value

None

Exception

No Exception

disable_logger

```
disable_logger()
```

Description:

disable linkkit sdk internal log

Available Working State:

- LinkKit.LinkKitState.INITIALIZED

Input Parameter

No input parameter

Return Value

None

Exception

No Exception

LinkKit

```
LinkKit(self, host_name, product_key, device_name, device_secret,
product_secret=None,user_data=None)
```

Description:

construct a LinkKit object

Available Working State:

None

Input Parameter

host_name

- type:string
- description:\${Region ID} in 地域和可用区, such as “cn-shanghai”

product_key

- type:string
- description:key of product,get from IoT Console

device_name

- type:string

- description:name of device,get from IoT Console

device_secret

- type:string
- description:secret of device,get from IoT Console,or use dynamic register device

product_secret

- type:string
- description:secret of product,get from IoT Console

user_data

- type:object
- description:user data,LinkKit all callback function give this user_data as last parameter

Return Value

LinkKit object

Exception

- type:ValueError

config_mqtt

```
config_mqtt(self, port=1883, protocol="MQTTv311", transport="TCP",
secure="TLS", keep_alive=60, clean_session=True,
max_inflight_message=20, max_queued_message=0,
auto_reconnect_min_sec=1,
auto_reconnect_max_sec=60,
cadata=None):
```

Description:

config mqtt connect parameter

Available Working State:

- LinkKit.LinkKitState.INITIALIZED

Input Parameter

port

- type:int

- description:port of mqtt broker

protocol

- type:string
- description:version of mqtt protocol.available:”MQTTv311”,”MQTTv31”

transport

- type:string
- description:transport layer type:only “TCP” support currently

secure

- type:string
- description:secure type,available:”TLS”,use TLS1.2;”” :use bare TCP,no secure

keep_alive

- type:int
- description: heartbeat of mqtt connect,unit is second

clean_session

- type:bool
- description:is clean session

max_inflight_message

- type:int
- description:max inflight message count in sdk internal buffer

max_queued_message

- type:int
- description:max received message count in sdk internal buffer,0 mean no limit

auto_reconnect_min_sec

- type:int
- description:auto reconnect wait min seconds,available range(1~120*60)

auto_reconnect_max_sec

- type:int
- description:auto reconnect wait max seconds,available range(1~120*60);
- auto_reconnect_max_sec must >= auto_reconnect_min_sec

cadata

- type:string
- description:ca file content use in SSL secure context

Return Value

None

Exception

- type:ValueError

config_device_info

```
config_device_info(interface_info)
```

Description:

config device info,only support interface info currently

Available Working State:

- LinkKit.LinkKitState.INITIALIZED

Input Parameter

interface_info

- type:string
- description:interface info

Return Value

rcrc:type:int0 success1 interface info too long,must less than 160 chars

Exception

ValueError: not string

destruct

```
destruct()
```

Description:

destruct a LinkKit object,will block wait for internal thread and object exit and destroy finish

Available Working State:

Any except LinkKit.LinkKitState.DESTRUCTED

Input Parameter

None

Return Value

None

Exception

None

connect_async

```
connect_async()
```

Description:

connect async,start internal working thread

Available Working State:

- LinkKit.LinkKitState.INITIALIZED

Input Parameter

None

Return Value

- rc
- rc:int 0 start working thread success 1 working thread already running

Exception

LinkKit.StateError

disconnect

```
disconnect()
```

Description:

disconnect mqtt

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Input Parameter

None

Return Value

None

Exception

LinkKit.StateError

Topic Management

publish_topic

```
publish_topic(topic, payload=None, qos=1)
```

Description:

publish a topic message to broker

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Input Parameter

topic

- type:string
- description:mqtt topic

payload

- type:string
- description:mqtt message send data

qos

- type:int
- description:mqtt message qos value,available 0,1

Return Value

- rc, mid
- rc:int 0 publish topic success 1 publish fail
- mid:int message id

Exception

- LinkKit.StateError
- ValueError

subscribe_topic

```
subscribe_topic(topic, qos=1)
```

Description:

subscribe topic

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Input Parameter

topic

- type:string
- description: mqtt topic or
- type:list(topic,qos)
- description:when subscribe multiple topics,can use list parameter to topic,qos will ignore

qos

- type:int
- description:mqtt message qos value,available 0,1 or
- if topic is list,this value ignore

Return Value

- rc, mid
- rc:int 0 subscribe topic success, 1 subscribe fail
- mid:int message id

Exception

- LinkKit.StateError
- ValueError

unsubscribe_topic

```
unsubscribe_topic(topic)
```

Description:

unsubscribe topics

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Input Parameter

topic

- type:string
- description: mqtt topic or
- type:list(topic)
- description:when unsubscribe multiple topics,can use list parameter to topic

Return Value

- rc, mid
- rc:int 0 unsubscribe topic success, 1 unsubscribe fail
- mid:int message id

Exception

- LinkKit.StateError
- ValueError

check_state

```
check_state()
```

Description:

get current working state

Available Working State:

Any

Input Parameter

None

Return Value

- LinkKit.LinkKitState.INITIALIZED
- LinkKit.LinkKitState.CONNECTING
- LinkKit.LinkKitState.CONNECTED
- LinkKit.LinkKitState.DISCONNECTING
- LinkKit.LinkKitState.DISCONNECTED
- LinkKit.LinkKitState.DESTRUCTING
- LinkKit.LinkKitState.DESTRUCTED

Exception

None

Thing Management

thing_setup

```
thing_setup(file=None)
```

Description:

setup thing model

Available Working State:

- LinkKit.LinkKitState.INITIALIZED

Input Parameter

file

- type:string
- description:thing model json file path,if set to None,thing only support custom protocol

Return Value

rc

- type:int
- 0 success,1 already set,2 file open fail

Exception

- LinkKit.StateError

thing_trigger_event

```
thing_trigger_event(event_tuple)
```

Description:

send a event to broker

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Input Parameter

event_tuple

- type:tuple(event, params)
- description:event is event identifier in things model,params is dict object and structure corresponding to thing model

Return Value

- rc, request_id
- rc:int 0 success,1 fail
- request_id:string when rc is 0,this value represent this request

Exception

- LinkKit.StateError

thing_post_property

```
thing_post_property(property_data)
```

Description:

upload property values to broker

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Input Parameter

property_data

- type:dict
- description:property_data structure corresponding to thing model

Return Value

- rc, request_id
- rc:int 0 success,1 fail
- request_id:string when rc is 0,this value represent this request

Exception

- LinkKit.StateError

thing_post_property

```
thing_post_property(property_data)
```

Description:

upload property values to broker

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Input Parameter

property_data

- type:dict
- description:property_data structure corresponding to thing model

Return Value

- rc, request_id
- rc:int 0 success,1 fail
- request_id:string when rc is 0,this value represent this request

Exception

- LinkKit.StateError

thing_update_tags

```
thing_update_tags(tag_map)
```

Description:

update tags of the thing

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Input Parameter

tag_map

- type:dict
- description: tag map

Return Value

- rc, request_id
- rc:int 0 success,1 fail
- request_id:string when rc is 0,this value represent this request

Exception

- LinkKit.StateError

thing_remove_tags

```
thing_remove_tags(tag_list)
```

Description:

remove tags of the thing

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Input Parameter

tag_list

- type:list
- description: the list of the tag keys

Return Value

- rc, request_id
- rc:int 0 success,1 fail
- request_id:string when rc is 0,this value represent this request

Exception

- LinkKit.StateError

thing_update_shadow

```
thing_update_shadow(reported, version)
```

Description:

update shadow of the thing

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Input Parameter

reported

- type: dict
- description: reported items
- example

```
reported = {"color":"red"}
```

version

- type: int
- description: the version of the shadow

Return Value

- rc, message_id
- rc:int 0 success,1 fail
- message_id:string when rc is 0,this value represent this request

Exception

- LinkKit.StateError

thing_get_shadow

```
thing_get_shadow()
```

Description:

get shadow of the thing

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Input Parameter

Return Value

- rc, message_id
- rc:int 0 success,1 fail
- message_id:string when rc is 0,this value represent this request

Exception

- LinkKit.StateError

local_get_latest_shadow

```
local_get_latest_shadow()
```

Description:

get shadow of the thing from the local cache

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Input Parameter

Return Value

- shadow data

Exception

- LinkKit.StateError

subscribe_rrpc_topic

```
subscribe_rrpc_topic(topic)
```

Description:

subscribe the RRPC topic

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Input Parameter

topic

- type: string
- description: topic name

Return Value

- rc, message_id
- rc:int 0 success,1 fail
- message_id:string when rc is 0,this value represent this request

Exception

- LinkKit.StateError

unsubscribe_rrpc_topic

```
unsubscribe_rrpc_topic(topic)
```

Description:

unsubscribe the RRPC topic

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Input Parameter

topic

- type: string
- description: topic name

Return Value

- rc, message_id
- rc:int 0 success,1 fail
- message_id:string when rc is 0,this value represent this request

Exception

- LinkKit.StateError

thing_answer_service

```
thing_answer_service(identifier, request_id, code, data=None)
```

Description:

answer service reply

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Input Parameter

identifier

- type:string
- description:service identifier of thing model

request_id

- type:string
- description:request id string for this call

code

- type:string
- description:broker response code for this request

data

- type:string
- description:answer service data

Return Value

rc

- type:int
- description:0 success, 1 fail

Exception

- LinkKit.StateError

thing_answer_rrpc

```
thing_answer_rrpc(identifier, id, response)
```

Description:

reply the response of RRPC request

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Input Parameter

id

- type:string
- description: request id

response

- type:string
- description: the payload of the response

Return Value

rc

- type:int
- description:0 success, 1 fail

Exception

- LinkKit.StateError

thing_raw_post_data

```
thing_raw_post_data(payload)
```

Description:

send raw bytes data to broker

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Input Parameter

payload

- type:string
- description:raw data to broker

Return Value

rc

- type:int
- description:0 success, 1 fail

Exception

- LinkKit.StateError

thing_raw_data_reply

```
thing_raw_data_reply(payload)
```

Description:

send reply through raw data to broker

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Input Parameter

payload

- type:string
- description:raw data to broker for reply

Return Value

rc

- type:int
- description:0 success, 1 fail

Exception

- LinkKit.StateError

on_publish_topic

```
on_publish_topic(mid, userdata)
```

Description:

callback after publish_topic call

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Callback Parameter

mid

- type:int
- description:publish message id

userdata

- type:
- description:same as LinkKit input parameter user_data

Return Value

None

Exception

None

on_subscribe_topic

```
on_subscribe_topic(mid, granted_qos, userdata)
```

Description:

callback after subscribe_topic call

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Callback Parameter

mid

- type:int
- description:publish message id

granted_qos

- type:list(int)
- description: corresponding to subscribe_topic parameter topic,0 represent qos=0,1 represent qos=1,128 represent subscribe error

userdata

- type:
- description:same as LinkKit input parameter user_data

Return Value

None

Exception

None

on_unsubscribe_topic

```
on_unsubscribe_topic(mid, userdata)
```

Description:

callback after unsubscribe topic

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Callback Parameter

mid

- type:int
- description:publish message id

userdata

- type:
- description:same as LinkKit input parameter user_data

Return Value

None

Exception

None

on_connect

```
on_connect(session_flag, rc, userdata)
```

Description:

callback after connect_async

Available Working State:

- LinkKit.LinkKitState.INITIALIZED

Callback Parameter

session_flag

- type:int
- description:is previous connect session,0 new session; 1 previous session

rc

- type:int
- description:

```
0: Connection successful
1: Connection refused - incorrect protocol version
2: Connection refused - invalid client identifier
3: Connection refused - server unavailable
4: Connection refused - bad username or password
5: Connection refused - not authorised
6: SSL wrong - ca file/data wrong
7: MQTT parameter wrong
8: Connect Timeout
9: network error
```

userdata

- type:
- description:same as LinkKit input parameter user_data

Return Value

None

Exception

None

on_device_dynamic_register

```
on_device_dynamic_register(rc, value, userdata)
```

Description:

dynamic register callback

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Callback Parameter

rc

- type:int
- description:0 register success,1 register fail

value

- type:string
- description:when rc==0,value is device secret,when rc !=0,value is return message

userdata

- type:
- description:same as LinkKit input parameter user_data

Return Value

None

Exception

None

on_thing_raw_data_post

```
on_thing_raw_data_post(payload, userdata)
```

Description:

callback after thing_raw_data_post done

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Callback Parameter

payload

- type:string
- description:mqtt message received data

userdata

- type:
- description:same as LinkKit input parameter user_data

Return Value

None

Exception

None

on_thing_raw_data_arrived

```
on_thing_raw_data_arrived(payload, userdata)
```

Description:

callback when raw data received

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Callback Parameter

payload

- type:string
- description:mqtt message received data

userdata

- type:
- description:same as LinkKit input parameter user_data

Return Value

None

Exception

None

on_thing_event_post

```
on_thing_event_post(event, request_id, code, data, message, userdata)
```

Description:

callback after thing_trigger_event call

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Callback Parameter

event

- type:string
- description:event is event identifier in things model

request_id

- type:string
- description: request_id same as return value of thing_trigger_event

code

- type:int
- description: broker response code,200 success,other value wrong occur

data

- type:dict
- description: broker response data

message

- type:string
- description: message corresponding broker response code

userdata

- type:
- description:same as LinkKit input parameter user_data

Return Value

None

Exception

None

on_thing_prop_post

```
on_thing_prop_post(request_id, code, data, message,userdata)
```

Description:

callback after thing_post_property

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Callback Parameter

request_id

- type:string
- description: request_id same as return value of thing_post_property

code

- type:int
- description: broker response code,200 success,other value wrong occur

data

- type:dict
- description: broker response data

message

- type:string
- description: message corresponding broker response code

userdata

- type:
- description:same as LinkKit input parameter user_data

Return Value

None

Exception

None

on_disconnect

```
on_disconnect(rc)
```

Description:

callback after connection disconnect

Available Working State:

- LinkKit.LinkKitState.DISCONNECTED

Callback Parameter

rc

- type:int
- description:0 success call for disconnect,1 network error

userdata

- type:
- description:same as LinkKit input parameter user_data

Return Value

None

Exception

None

on_thing_enable

```
on_thing_enable(userdata)
```

Description:

callback for things api able to call

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Callback Parameter

userdata

- type:
- description:same as LinkKit input parameter user_data

Return Value

None

Exception

None

on_thing_disable

```
on_thing_disable(userdata)
```

Description:

callback when things model disabled

Available Working State:

- LinkKit.LinkKitState.DISCONNECTED

Callback Parameter

userdata

- type:
- description:same as LinkKit input parameter user_data

Return Value

None

Exception

None

on_thing_call_service

```
on_thing_call_service(identifier, request_id, params, userdata)
```

Description:

callback after receiver call service request

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Callback Parameter

identifier

- type:string
- description:identifier is service identifier in things model

request_id

- type:string
- description: request_id represent this service call

params

- type:dict
- description: params structure corresponding to thing model service define

userdata

- type:
- description:same as LinkKit input parameter user_data

Return Value

Exception

on_thing_prop_changed

```
on_thing_prop_changed(params, userdata)
```

Description:

callback after receive broker set property message

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Callback Parameter

params

- type:dict
- description: params structure corresponding to thing model property define

Return Value

None

Exception

None

on_thing_shadow_get

```
on_thing_shadow_get(payload, userdata)
```

Description:

callback after get the shadow

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Callback Parameter

payload

- type:dict
- description: payload of the shadow

Return Value

None

Exception

None

on_topic_rrpc_message

```
on_topic_rrpc_message(id, topic, payload, qos, userdata)
```

Description:

callback for rrpc messages

Available Working State:

- LinkKit.LinkKitState.CONNECTED

Callback Parameter

id

- type: string
- description: request id

topic

- type:string
- description: the topic name

qos

- type:int
- description: QoS value

Return Value

None

Exception

None