ALIBABA CLOUD

阿里云

Python SDK

文档版本：20210302

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.开发环境设置

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 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 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
```

2.认证与连接

本文介绍如何初始化设备信息，并建立设备与云端的连接。

云端域名

设备可以接入物联网平台的公共实例或者企业实例，SDK初始化时需要根据实例类型来指定设备连接的域名：

- 公共实例

  若设备接入公共实例，需要指定公共实例所在的阿里云站点。需要注意的是物联网平台这个服务并没有在所有的阿里云地域都进行了部署，因此开发者需要先查看自己创建的产品所在的物联网平台地域，然后再从“地域和可用区中查看对应的RegionID，比如“华东2（上海）”的RegionID为cn-shanghai。

- 企业实例

  用户需要在企业实例中查看实例的接入域名，请参见文档“实例管理，下图圈选部分即是企业实例的接入域名：

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，需要指定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，各地域信息可查询 https://help.aliyun.com/document_detail/40654.html。

- 一机一密

下面的示例代码配置设备的设备证书以及连接的公共示例的RegionID，其中RegionID以华东2（上海）为例：

```
from linkkit import linkkit
lk = linkkit.LinkKit(
  host_name="cn-shanghai",
  product_key="xxxxxxxxxx",
  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。

> **说明** 目前Python SDK只在华东2（上海）站点支持一型一密。

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="xxxxxxxxxx",
  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地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"。

企业实例域名配置的更改

针对企业实例，需要调用config_mqtt接口，填入企业实例域名。以 examples/mqtt_connect_TCP.py为例：

在原有的lk.config_mqtt(secure="")的基础上， 加上企业实例域名的信息，调整为lk.config_mqtt(secure="", endpoint="${instance-endpoint}")，其中${instance-endpoint}为前面章节描述的企业实例域名。

```
# 修改如下所示：# - 标识为原有配置，+ 标识为更改后的配置。
# # 具体为把lk.config_mqtt(secure="") 这一句, 替换为lk.config_mqtt(secure="", endpoint="${instance-endpoint}"), 其中${instance-endpoint}为企业实例域名。
-lk.config_mqtt(secure="")
+lk.config_mqtt(secure="", endpoint="${instance-endpoint}")
 lk.connect_async()
```

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()
```

> **注意** 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。