ALIBABA CLOUD

阿里云

iOS SDK

文档版本：20201123

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.工程配置

您可以使用Link SDK提供的iOS语言版本，搭建设备与云端的双向数据通道。SDK包含初始化、去初始化、数据上下行以及订阅/取消订阅接口等内容，目前最新的版本为1.2.1。

本章节中SDK 采用 cocoapods 管理依赖，建议采用 cocoapods 1.2.0 或以上版本。在xcode工程Podfile中添加以下行，集成SDK。

```
source 'https://github.com/CocoaPods/Specs.git'
source 'https://github.com/aliyun/aliyun-specs.git'
target "necslinkdemo" do
pod 'IotLinkKit', '1.2.1'
end
```

iOS Linkkit SDK Demo

iOS SDK提供Demo，供您参考使用；下载本Demo将默认您同意本软件许可协议。

单击下载iOS SDK Demo。

2.认证与连接

本文介绍如何进行 SDK 初始化，建立设备与云端的连接。

物联网平台对设备的身份认证支持设备证书、X.509、ID2等方式，对于设备证书方式还增加动态注册机制用于简化设备的证书烧录。Java SDK目前只支持设备证书的认证方式，并且暂不支持“动态注册”；客户可以访问“设备安全认证”了解这些认证方式的具体定义与区别。

设备证书的方式指设备具备阿里云物联网平台颁发的

- ProductKey：产品的型号定义
- DeviceName：设备的唯一标识
- DeviceSecret：设备的共享密钥

设备连接物联网平台时需要上报自己的ProductKey、DeviceName以及使用DeviceSecret计算的签名数据，物联网平台将会根据收到的ProductKey和DeviceName以及其它数据进行签名计算，如果验证设备上报的签名正确，则认为设备是合法设备；因此Java SDK在初始化时需要提供设备的证书信息。

另外一个比较重要的参数是物联网平台的域名。物联网平台在国内的多个阿里云地域（比如上海、深圳、北京）、以及多个国家和地区（比如日本、新加坡、美国等等）进行了部署，另外物联网平台还分为公共实例与企业实例的形态，设备在连接的时候需要指定设备需要连接的物联网平台服务端的域名：

- 公共实例

  公共实例的域名结构为：{*productKey*}.iot-as-mqtt.*RegionID*.aliyuncs.com"，其中的RegionID即代表相应的站点，其取值可以从“地域和可用区”中查看，需要注意的是物联网平台这个服务并不是在所有的阿里云地域都进行了部署，因此这里填入的RegionID一定要和产品创建的地域相同，比如“华东2（上海）”的RegionID是cn-shanghai，假设产品的productKey是abc，那么最终的域名为：abc.iot-as-mqtt.cn-shanghai.aliyuncs.com。SDK中的默认域名为华东2，若设备接入阿里云华东2站点的公共实例，那么可以不用指定域名。

- 企业实例

  用户需要在企业实例中查看实例的接入URL，请参见文档“实例管理：

  

  上图中圈选部分则为物联网平台企业实例的MQTT接入域名。

SDK 初始化

SDK初始化时，会初始化跟云端的MQTT长连接通道。初始化之前请检查网络情况，如果未连接网络请提示用户先连接网络。

请注意初始化时的 'resultBlock' 回调成功并不代表MQTT长连接通道建立成功，MQTT长连接通道连接是否成功请先调用此方法 '[[LinkKitEntry sharedKit] registerChannelListener:self];'，根据注册成功的 listener的 [onConnectStateChange:state:] 回调方法来判断通道连接情况。

下面的代码示例如何初始化SDK接入物联网平台在华东2（上海）的公共示例：

```
#import <IotLinkKit/IotLinkKit.h>
  //初始化前请先注册listener侦听长连接通道的连接状态变化
  [[LinkKitEntry sharedKit] registerChannelListener:self];
  ////输入设备三元组信息
  self.productKey = self.textFieldProductKey.text ;
  self.deviceName = self.textFieldDeviceName.text  ;
  self.deviceSecret = self.textFieldDeviceSecret.text;
  LinkkitChannelConfig * channelConfig = [[LinkkitChannelConfig alloc] init];
  channelConfig.productKey = self.productKey;
  channelConfig.deviceName = self.deviceName;
  channelConfig.deviceSecret = self.deviceSecret;
  channelConfig.cleanSession = (self.cleanSession == 1);
  //channelConfig.server = @"your custom server url";//不设置server，表示使用默认的server地址
  //channelConfig.port = 1883;
  LinkkitSetupParams * setupParams = [[LinkkitSetupParams alloc] init];
  setupParams.appVersion = self.appVersion;
  setupParams.channelConfig = channelConfig;
  [[LinkKitEntry sharedKit] setup:setupParams resultBlock:^(BOOL succeeded, NSError * _Nullable error) {
    LinkkitLogDebug(@"setup error : %@", error);
    dispatch_async(dispatch_get_main_queue(), ^{
      [self ims_showHUDWithMessage:[NSString stringWithFormat:@"Linkkit 初始化 : %@",
                    succeeded ? @"成功" : @"失败"]];
    });
  }];
```

若设备接入非华东2(上海)的物联网平台的公共实例，或者接入物联网平台的企业实例，那么需要把上面示例代码中的channelConfig.server设置为前面所描述的物联网平台服务端的域名。

SDK 反初始化

反初始化时，会断开跟云端的MQTT长连接。如果需要注销初始化，调用如下接口。

```
  [[LinkKitEntry sharedKit] destroy:^(BOOL succeeded, NSError * _Nullable error) {
    LinkkitLogDebug(@"kit destroy error : %@", error);
    dispatch_async(dispatch_get_main_queue(), ^{
      [self ims_showHUDWithMessage:[NSString stringWithFormat:@"Linkkit 去初始化 : %@",
                    succeeded ? @"成功" : @"失败"]];
    });
  }];
```

其他设置

日志开关

打开SDK内部日志输出开关：

```
#import <IMSLog/IMSLog.h>
NSString * const IMS_DEMO_TAG = @"LinkkitDemo";
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
  // 添加以下代码打开 SDK 的控制台Log
  [IMSLog setAllTagsLevel:IMSLogLevelAll];
  //创建日志面板助手，注入到日志框架
  //[IMSLog addAssistant:[[IMSDashboardLogAssistant alloc] init]];
  //设置在控制台显示日志
  [IMSLog showInConsole:YES];
  //设置 Demo 的日志 tag
  [IMSLog registerTag:IMS_DEMO_TAG];
  return YES;
}
```

MQTT长连接通道状态变化 listener

实现示例代码如下：

```
- (void)onConnectStateChange:(nonnull NSString *)connectId state:(LinkkitChannelConnectState)state {
    NSString * connectTip = nil;
    if (state == LinkkitChannelStateConnected) {
        connectTip = @"已连接";
    } else if (state == LinkkitChannelStateDisconnected) {
        connectTip = @"未连接";
    } else {
        connectTip = @"连接中";
    }
    dispatch_async(dispatch_get_main_queue(), ^{
        self.labelConnectState.text = connectTip;
    });
}
- (void)onNotify:(nonnull NSString *)connectId topic:(nonnull NSString *)topic data:(id _Nullable)data {
    NSString * downData = [NSString stringWithFormat:@"收到下推，topic : %@ \r\n", topic];
    downData = [downData stringByAppendingString:[NSString stringWithFormat:@"\r\n数据 : %@", data]];
    LinkkitLogDebug(@"kit recv  topic : %@", topic);
    dispatch_async(dispatch_get_main_queue(), ^{
        self.textViewDownData.text = downData;
    });
}
- (BOOL)shouldHandle:(nonnull NSString *)connectId
            topic:(nonnull NSString *)topic {
    return YES;
}
```