SDK 中心

Java

腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2025 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

【商标声明】

腾讯云

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

Java

最近更新时间：2025-03-11 17:54:22

简介

欢迎使用腾讯云开发者工具套件（SDK）3.0，SDK3.0是云 API3.0 平台的配套工具。为方便 JAVA 开发者调试和接入腾讯云产品 API，这里向您介绍适用于 Java 的腾讯云开发工具包，并提供首次使用开发工具包的简单示例。让您快速获取腾讯云 Java SDK 并开始调用。

依赖环境

1. JDK 7 版本及以上。
2. 从 腾讯云控制台 开通相应产品。

获取安装

使用 SDK 需要 API 密钥，可前往 腾讯云控制台 – 访问密钥 页面申请，API 密钥包括 SecretID 和 SecretKey，密钥须严格保管，避免泄露。

通过 Maven 安装（推荐）

从 3.1.500 版本开始，本项目使用 KonaJDK 编译发布。
通过 Maven 获取安装是使用 JAVA SDK 的推荐方法，Maven 是 JAVA 的依赖管理工具，支持您项目所需的依赖项，并将其安装到项目中。
关于 Maven 详细可参考 Maven 官网，并下载对应系统 Maven 安装包进行安装。

安装指定产品 SDK（推荐）

例如，安装指定产品包：

```
<dependency>
    <groupId>com.tencentcloudapi</groupId>
    <artifactId>tencentcloud-sdk-java-指定产品包名</artifactId>
    <!-- 如 CVM 产品包：tencentcloud-sdk-java-cvm -->
    <!-- 请到 maven 官网查询 sdk 的所有版本，例如 cvm 的产品包链接为
https://central.sonatype.com/artifact/com.tencentcloudapi/tencentcloud-
sdk-java-cvm/versions -->
    <version>3.1.1000</version>
</dependency>
```

具体产品的包名缩写请参考 products.md 中的包名字段。

安装全产品 SDK

```
<dependency>
    <groupId>com.tencentcloudapi</groupId>
    <artifactId>tencentcloud-sdk-java</artifactId>
    <!-- go to
https://central.sonatype.com/artifact/com.tencentcloudapi/tencentcloud-
sdk-java/versions and get the latest version. -->
    <!-- 请到
https://central.sonatype.com/artifact/com.tencentcloudapi/tencentcloud-
sdk-java/versions 查询所有版本，最新版本如下 -->
    <version>3.1.1000</version>
</dependency>
```

全产品 SDK 包含了所有云产品的调用代码，体积偏大，对体积敏感的场景，推荐安装指定产品 SDK。

注意事项

- 安装全产品 SDK 和安装指定产品的 SDK 两种方式只能选择其中一种。
- 如果同时安装多个产品的包，建议多个产品的包和 common 包保持在同一个版本。
- 项目中添加 Maven 依赖项，只需在 Maven pom.xml 添加以下依赖项即可。注意这里的版本号只是举例，您可以在 Maven 仓库上找到最新的版本（最新版本是 3.1.1000）。请知悉，SDK 是先确认 `mvn deploy` 发布成功后再更新 GitHub tag，但是 Maven 官网索引更新有延迟，导致新发布的版本暂时（约1-2小时）在 Maven 官网搜索不到，实际不影响使用最新版本，您可以正常执行 `mvn compile` 等指令。
- maven 仓库中显示的 4.0.11 是废弃版本，我们已经联系 maven 官方删除 jar 包，但 maven 索引无法清除，请勿使用。
- 无法使用官方源的用户可以使用镜像源加速下载，编辑 maven 的 settings.xml 配置文件，在 mirrors 段落增加镜像配置：

```
<mirror>
  <id>tencent</id>
  <name>tencent maven mirror</name>
  <url>https://mirrors.tencent.com/nexus/repository/maven-public/</url>
  <mirrorOf>*</mirrorOf>
</mirror>
```

通过源码包安装

1. 前往 Github 仓库 或者 Gitee 仓库 下载源码压缩包。
2. 将源码包解压到您项目合适的位置。
3. 将 vendor 目录下的 jar 包放在 Java 可以找到的路径中。
4. 引用方法可参考以下示例。

示例

以查询实例接口 DescribeInstances 为例：

简化版

```
import com.tencentcloudapi.common.Credential;
import com.tencentcloudapi.common.exception.TencentCloudSDKException;
import com.tencentcloudapi.cvm.v20170312.CvmClient;
import
com.tencentcloudapi.cvm.v20170312.models.DescribeInstancesRequest;
import
com.tencentcloudapi.cvm.v20170312.models.DescribeInstancesResponse;

public class DescribeInstances {
    public static void main(String[] args) {
        try {
            // 为了保护密钥安全，建议将密钥设置在环境变量中或者配置文件中，请参考本
文凭证管理章节。
            // 硬编码密钥到代码中有可能随代码泄露而暴露，有安全隐患，并不推荐。
            // Credential cred = new Credential("SecretId",
"SecretKey");
            Credential cred = new
Credential(System.getenv("TENCENTCLOUD_SECRET_ID"),
System.getenv("TENCENTCLOUD_SECRET_KEY"));
            CvmClient client = new CvmClient(cred, "ap-shanghai");

            DescribeInstancesRequest req = new
DescribeInstancesRequest();
            DescribeInstancesResponse resp =
client.DescribeInstances(req);


System.out.println(DescribeInstancesResponse.toJsonString(resp));
        } catch (TencentCloudSDKException e) {
            System.out.println(e.toString());
        }
    }
}
```

详细版

```
import com.tencentcloudapi.common.Credential;
import com.tencentcloudapi.common.exception.TencentCloudSDKException;
// 导入对应产品模块的client
import com.tencentcloudapi.cvm.v20170312.CvmClient;
// 导入要请求接口对应的request response类
import
com.tencentcloudapi.cvm.v20170312.models.DescribeInstancesRequest;
import
com.tencentcloudapi.cvm.v20170312.models.DescribeInstancesResponse;
import com.tencentcloudapi.cvm.v20170312.models.Filter;
//导入可选配置类
import com.tencentcloudapi.common.profile.ClientProfile;
import com.tencentcloudapi.common.profile.HttpProfile;
import com.tencentcloudapi.common.profile.Language;

public class DescribeInstances {
    public static void main(String[] args) {
        try {
            // 实例化一个认证对象，入参需要传入腾讯云账户 SecretId，SecretKey。
            // 为了保护密钥安全，建议将密钥设置在环境变量中或者配置文件中，请参考本
文凭证管理章节。
            // 硬编码密钥到代码中有可能随代码泄露而暴露，有安全隐患，并不推荐。
            // Credential cred = new Credential("SecretId",
"SecretKey");
            Credential cred = new
Credential(System.getenv("TENCENTCLOUD_SECRET_ID"),
System.getenv("TENCENTCLOUD_SECRET_KEY"));

            // 实例化一个http选项，可选的，没有特殊需求可以跳过
            HttpProfile httpProfile = new HttpProfile();
            // 从3.0.96版本开始，单独设置 HTTP 代理
            // httpProfile.setProxyHost("真实代理ip");
            // httpProfile.setProxyPort(真实代理端口);
            httpProfile.setReqMethod("GET"); // get请求(默认为post请求)
            httpProfile.setConnTimeout(30); // 请求连接超时时间，单位为秒(默
认60秒)
            httpProfile.setWriteTimeout(30);  // 设置写入超时时间，单位为秒
(默认0秒)
            httpProfile.setReadTimeout(30);  // 设置读取超时时间，单位为秒(默
认0秒)
            httpProfile.setEndpoint("cvm.ap-
shanghai.tencentcloudapi.com"); // 指定接入地域域名(默认就近接入)
```

```
            // 实例化一个client选项，可选的，没有特殊需求可以跳过
            ClientProfile clientProfile = new ClientProfile();
            clientProfile.setSignMethod("HmacSHA256"); // 指定签名算法(默认
为HmacSHA256)
            // 自3.1.80版本开始，SDK 支持打印日志。
            clientProfile.setHttpProfile(httpProfile);
            clientProfile.setDebug(true);
            // 从3.1.16版本开始，支持设置公共参数 Language，默认不传，选择
(ZH_CN or EN_US)
            clientProfile.setLanguage(Language.EN_US);
            // 实例化要请求产品(以cvm为例)的client对象,clientProfile是可选的
            CvmClient client = new CvmClient(cred, "ap-shanghai",
clientProfile);

            // 实例化一个cvm实例信息查询请求对象,每个接口都会对应一个request对象。
            DescribeInstancesRequest req = new
DescribeInstancesRequest();

            // 填充请求参数,这里request对象的成员变量即对应接口的入参
            // 您可以通过官网接口文档或跳转到request对象的定义处查看请求参数的定义
            Filter respFilter = new Filter(); // 创建Filter对象，以zone的维
度来查询cvm实例
            respFilter.setName("zone");
            respFilter.setValues(new String[] { "ap-shanghai-1", "ap-
shanghai-2" });
            req.setFilters(new Filter[] { respFilter }); // Filters 是成
员为Filter对象的列表

            // 通过client对象调用DescribeInstances方法发起请求。注意请求方法名
与请求对象是对应的
            // 返回的resp是一个DescribeInstancesResponse类的实例，与请求对象对
应
            DescribeInstancesResponse resp =
client.DescribeInstances(req);

            // 输出json格式的字符串回包

System.out.println(DescribeInstancesResponse.toJsonString(resp));

            // 也可以取出单个值。
            // 您可以通过官网接口文档或跳转到response对象的定义处查看返回字段的定
义
            System.out.println(resp.getTotalCount());
```

```
        } catch (TencentCloudSDKException e) {
            System.out.println(e.toString());
        }
    }
}
```

详细版示例代码分步骤介绍

1. 导入待调用的类：

```
import com.tencentcloudapi.common.Credential;
import com.tencentcloudapi.common.exception.TencentCloudSDKException;
// 导入对应产品模块的client
import com.tencentcloudapi.cvm.v20170312.CvmClient;
// 导入要请求接口对应的request response类
import
com.tencentcloudapi.cvm.v20170312.models.DescribeInstancesRequest;
import
com.tencentcloudapi.cvm.v20170312.models.DescribeInstancesResponse;
import com.tencentcloudapi.cvm.v20170312.models.Filter;
//导入可选配置类
import com.tencentcloudapi.common.profile.ClientProfile;
import com.tencentcloudapi.common.profile.HttpProfile;
import com.tencentcloudapi.common.profile.Language;
```

2. 实例化一个认证对象 cred，入参需要传入腾讯云账户密钥 secretId 和 secretKey，您可以前往 API 密钥管理 页面获取密钥。

> **注意：**
> 此处还需注意密钥对的保密。

```
// 为了保护密钥安全，建议将密钥设置在环境变量中或者配置文件中，请参考本文凭证管理章
节。
// 硬编码密钥到代码中有可能随代码泄露而暴露，有安全隐患，并不推荐。
Credential cred = new
Credential(System.getenv("TENCENTCLOUD_SECRET_ID"),
System.getenv("TENCENTCLOUD_SECRET_KEY"));
```

3. 实例化一个 http 选项，若没有特殊需求可参照简化版示例代码跳过设置。若有需求可以参照下方示例代码设置 http 选项中的参数。

```
HttpProfile httpProfile = new HttpProfile();
//  从3.1.16版本开始，单独设置 HTTP 代理
// httpProfile.setProxyHost("真实代理ip");
// httpProfile.setProxyPort(真实代理端口);
// get请求(默认为post请求)
httpProfile.setReqMethod("GET"); // get请求(默认为post请求)
httpProfile.setProtocol("https://");  // 合法值：https:// 或者 http://
，公有云只能使用 https。
httpProfile.setConnTimeout(30); // 请求连接超时时间，单位为秒(默认60秒)
httpProfile.setWriteTimeout(30);  // 设置写入超时时间，单位为秒(默认0秒)
httpProfile.setReadTimeout(30);  // 设置读取超时时间，单位为秒(默认0秒)
httpProfile.setEndpoint("cvm.ap-shanghai.tencentcloudapi.com"); // 指
定接入地域域名(默认就近接入)
```

4. 实例化一个 client 选项，若没有特殊需求可参照简化版示例代码跳过设置。若有需求可以参照下方示例代码设置 client 选项中的参数。

```
ClientProfile clientProfile = new ClientProfile();
clientProfile.setSignMethod("HmacSHA256"); // 指定签名算法(默认为
HmacSHA256)
// 自3.1.80版本开始，SDK 支持打印日志。
clientProfile.setHttpProfile(httpProfile);
clientProfile.setDebug(true);
// 从3.1.16版本开始，支持设置公共参数 Language，默认不传，选择(ZH_CN or
EN_US)
clientProfile.setLanguage(Language.EN_US);
```

5. 实例化要请求产品的 client 对象，示例代码以 CVM 产品为例，实例化 CvmClient 对象。

```
CvmClient client = new CvmClient(cred, "ap-shanghai", clientProfile);
```

6. 实例化一个 CVM 实例信息查询请求对象 req，每个接口都会对应一个 request 对象，若有需要可以参照官网文档设置对象中的参数。

```
DescribeInstancesRequest req = new DescribeInstancesRequest();
// 填充请求参数,这里request对象的成员变量即对应接口的入参
// 您可以通过官网接口文档或跳转到request对象的定义处查看请求参数的定义
Filter respFilter = new Filter(); // 创建Filter对象，以zone的维度来查询
cvm实例
respFilter.setName("zone");
```

```
respFilter.setValues(new String[] { "ap-shanghai-1", "ap-shanghai-2"
});
req.setFilters(new Filter[] { respFilter }); // Filters 是成员为Filter
对象的列表
```

7. 通过 client 对象调用您想调用的接口，返回对应的 Response 类的实例。

```
// 通过client对象调用DescribeInstances方法发起请求。注意请求方法名与请求对象是
对应的
// 返回的resp是一个DescribeInstancesResponse类的实例，与请求对象对应
DescribeInstancesResponse resp = client.DescribeInstances(req);
```

8. 可以打印输出返回的数据值。

```
// 输出json格式的字符串回包
System.out.println(DescribeInstancesResponse.toJsonString(resp));
// 也可以取出单个值。
// 您可以通过官网接口文档或跳转到response对象的定义处查看返回字段的定义
System.out.println(resp.getTotalCount());
}
```

9. 使用 catch 处理报错，错误码详细内容请参考官网产品文档对应的错误码章节。

```
catch (TencentCloudSDKException e) {
    System.out.println(e.toString());
}
```

更多示例

您可以在 GitHub 中 examples 目录下找到更多详细的示例。

相关配置

代理

从 3.0.96 版本开始，可以单独设置 HTTP 代理：

```
HttpProfile httpProfile = new HttpProfile();
httpProfile.setProxyHost("代理ip或者域名");
httpProfile.setProxyPort(代理端口);
httpProfile.setProxyUsername("代理用户名");
```

```
httpProfile.setProxyPassword("代理密码");
```

语言

从 3.1.16 版本开始，我们添加了对公共参数 Language 的支持，以满足部分产品国际化的需求。和以前一样，Language 默认不传，行为由各产品接口自行决定。通常情况下，接口的默认语言是中文，但也有一些接口的默认语言是英文。目前可选值为中文或者英文，如果您需要设置 Language 参数，可使用以下方法：

```
import com.tencentcloudapi.common.profile.ClientProfile;
import com.tencentcloudapi.common.profile.Language;
...
    ClientProfile clientProfile = new ClientProfile();
    clientProfile.setLanguage(Language.ZH_CN);
```

支持 http

SDK 支持 HTTP 协议和 HTTPS 协议，通过设置 HttpProfile 的 setProtocol() 方法可以实现协议间的切换：

```
HttpProfile httpProfile = new HttpProfile();
httpProfile.setProtocol("http://"); //http协议
httpProfile.setProtocol("https://"); //https协议
```

支持打印日志

自 3.1.80 版本开始，SDK 支持打印日志。您可以通过以下步骤来设置：
首先，在创建 CLientProfile 对象时，设置 debug 模式为真，这将打印 SDK 异常信息和流量信息。

```
ClientProfile clientProfile = new ClientProfile();
clientProfile.setDebug(true);
```

腾讯云 Java SDK 使用 commons.logging 类进行打印日志，示例如下：

```
九月 10, 2020 5:14:30 下午 com.tencentcloudapi.cvm.v20170312.CvmClient
info
信息: send request, request url: https://cvm.ap-
shanghai.tencentcloudapi.com/?
Nonce=367595572&Action=DescribeInstances&Filters.0.Values.1=ap-shanghai-
2&Version=2017-03-12&Filters.0.Values.0=ap-shanghai-
1&SecretId=**********************************&Filters.0.Name=zone&Requ
estClient=SDK_JAVA_3.1.129&Region=ap-
```

```
shanghai&SignatureMethod=HmacSHA256&Timestamp=1599729270&Signature=DcGRP
dquMZZRPj1NFXP5bsOGnRlaT2KXy7aegNhZa00%3D. request headers information:
九月 10, 2020 5:14:32 下午 com.tencentcloudapi.cvm.v20170312.CvmClient
info
信息:  recieve response, response url: https://cvm.ap-
shanghai.tencentcloudapi.com/?
Nonce=367595572&Action=DescribeInstances&Filters.0.Values.1=ap-shanghai-
2&Version=2017-03-12&Filters.0.Values.0=ap-shanghai-
1&SecretId=**********************************&Filters.0.Name=zone&Requ
estClient=SDK_JAVA_3.1.129&Region=ap-
shanghai&SignatureMethod=HmacSHA256&Timestamp=1599729270&Signature=DcGRP
dquMZZRPj1NFXP5bsOGnRlaT2KXy7aegNhZa00%3D, response headers: Server:
nginx;Date: Thu, 10 Sep 2020 09:14:32 GMT;Content-Type:
application/json;Content-Length: 103;Connection: keep-alive;OkHttp-
Selected-Protocol: http/1.1;OkHttp-Sent-Millis: 1599729271230;OkHttp-
Received-Millis: 1599729272020;,response body information:
com.squareup.okhttp.internal.http.RealResponseBody@8646db9
```

您可以根据自己的需要配置日志打印类，例如 log4j。配置方法如下：

- 配置 pom 文件，设置 log4j 版本。

```
    <dependency>
      <groupId>log4j</groupId>
      <artifactId>log4j</artifactId>
      <version>1.2.17</version>
    </dependency>
```

- 设置环境变量为 log4j，并创建 log 类。

```
System.setProperty("org.apache.commons.logging.Log",
"org.apache.commons.logging.impl.Log4JLogger");
Log logger = LogFactory.getLog("TestLog");
logger.info("hello world");
```

地域容灾

从 3.1.779 版本开始，腾讯云的 Java SDK 支持地域容灾功能。当满足以下条件时，SDK 会自动将您的请求设置为备选地域：

1. 失败次数 ≥ 5次
2. 失败率 ≥ 75%

您可以通过以下设置来配置相关功能：

```
// 设置备用请求地址，不需要指定服务，SDK 会自动在头部加上服务名(如cvm)
// 例如，设置为 ap-guangzhou.tencentcloudapi.com，则最终的请求为 cvm.ap-guangzhou.tencentcloudapi.com
clientProfile.setBackupEndpoint("ap-guangzhou.tencentcloudapi.com");

// 自定义断路器条件
CircuitBreaker.Setting setting = new CircuitBreaker.Setting();
setting.maxFailNum = 6;
setting.maxFailPercentage = 0.8f;
CircuitBreaker rb = new CircuitBreaker(setting);
client.setRegionBreaker(rb);
```

请注意，此功能仅支持单个客户端的同步请求。

Common Client

从3.1.303版本开始，腾讯云 Java SDK 支持使用泛用性的 API 调用方式 (Common Client) 进行请求。您只需要安装 Common 包，即可向任何产品发起调用。

> **注意：**
> 您必须明确了解您调用的接口所需的参数，否则可能会导致调用失败。

目前仅支持使用 POST 方式，且签名方法必须使用签名方法 v3。详细使用请参见示例 使用 Common Client 进行调用 。

支持重试请求

从 3.1.310 版本开始，腾讯云 Java SDK 支持重试请求。您可以为每个请求设置重试次数，如果接口请求未成功，SDK 会进行重试，直到请求成功或达到设置的重试次数。待设置的重试次数最大为10，最小为0，每次重试失败需要睡眠1秒钟。详细使用请参见示例 使用 retry 进行重试请求 。

凭证管理

腾讯云 Java SDK 目前支持以下几种方式进行凭证管理：

1. 环境变量

   默认读取环境变量 `TENCENTCLOUD_SECRET_ID` 和 `TENCENTCLOUD_SECRET_KEY` 获取 secretId 和 secretKey。相关代码如下：

   ```
   Credential cred = new EnvironmentVariableCredentialsProvider().getCredentials();
   ```

2. 配置文件

配置文件路径要求为：

- Windows： `c:\Users\NAME\.tencentcloud\credentials`
- Linux： `~/.tencentcloud/credentials` 或 `/etc/tencentcloud/credentials`

配置文件格式如下：

```
[default]
secret_id = xxxxx
secret_key = xxxxx
```

相关代码如下：

```
Credential cred = new ProfileCredentialsProvider().getCredentials();
```

3. 角色扮演

有关角色扮演的相关概念请参见 腾讯云角色概述。如果使用此种方式，您需要在腾讯云访问管理控制台上创建一个角色，具体创建过程请参见 腾讯云角色创建。
在您拥有角色后，可以通过持久密钥和 roleArn 获取临时凭证，SDK 会自动刷新临时凭证，相关代码如下：

```
Credential cred = new STSCredential("secretId", "secretKey",
"roleArn", "roleSessionName");
```

4. 实例角色

有关实例角色的相关概念请参见 腾讯云实例角色。
在您为实例绑定角色后，您可以在实例中访问相关元数据接口获取临时凭证，SDK 会自动刷新临时凭证。相关代码如下：

```
Credential cred = new CvmRoleCredential();
```

5. TKE OIDC 凭证

有关 TKE OIDC 凭证的相关示例请参见 Pod 使用 CAM 对数据库身份验证。

```
OIDCRoleArnProvider provider = new OIDCRoleArnProvider();
Credential credential = provider.getCredentials();
```

6. 凭证提供链

腾讯云 Java SDK 提供了凭证提供链，它会按照**环境变量 > 配置文件 > 实例角色 > TKE OIDC 凭证**的顺序尝试获取凭证，并返回第一个获取到的凭证。相关代码如下：

```
Credential cred = new DefaultCredentialsProvider().getCredentials();
```

凭证管理详细使用请参见 使用凭证提供链 。

自定义 SSLSocketFactory 和 X509TrustManager

```
ClientProfile cpf = new ClientProfile();
cpf.getHttpProfile().setSslSocketFactory(new MySSLSocketFactoryImpl());
cpf.getHttpProfile().setX509TrustManager(new MyX509TrustManagerImpl());
```

跳过证书校验

```
ClientProfile cpf = new ClientProfile();
// 创建一个信任所有证书的 TrustManager
TrustManager[] trustAllCerts = new TrustManager[] {
        new X509TrustManager() {
            @Override
            public void checkClientTrusted(X509Certificate[] chain,
String authType) throws CertificateException {
            }

            @Override
            public void checkServerTrusted(X509Certificate[] chain,
String authType) throws CertificateException {
            }

            @Override
            public X509Certificate[] getAcceptedIssuers() {
                return new X509Certificate[0];
            }
        }
};
SSLContext sslContext = SSLContext.getInstance("TLS");
sslContext.init(null, trustAllCerts, new java.security.SecureRandom());
SSLSocketFactory sslSocketFactory = sslContext.getSocketFactory();
httpProfile.setSslSocketFactory(sslSocketFactory);
httpProfile.setX509TrustManager((X509TrustManager) trustAllCerts[0]);
httpProfile.setHostnameVerifier(new HostnameVerifier() {
    @Override
    //创建一个不进行主机名验证的 HostnameVerifier
    public boolean verify(String hostname, SSLSession session) {
```

```
        return true;
    }
});
```

自定义 Header

DescribeInstancesRequest 示例

```
ClientProfile cpf = new ClientProfile();
// 自定义 Header 需要使用 v3 签名方式
cpf.setSignMethod(ClientProfile.SIGN_TC3_256);

DescribeInstancesRequest request = new DescribeInstancesRequest();
Map<String, String> header = new HashMap<String, String>();
header.put("X-TC-TraceId","ffe0c072-8a5d-4e17-8887-a8a60252abca");
request.SetHeader(header);
```

CommonClientRequest 示例

```
ClientProfile cpf = new ClientProfile();
// 自定义 Header 需要使用 v3 签名方式
cpf.setSignMethod(ClientProfile.SIGN_TC3_256);

CommonClientRequest request = new CommonClientRequest();
Map<String, String> header = new HashMap<String, String>();
header.put("X-TC-TraceId","ffe0c072-8a5d-4e17-8887-a8a60252abca");
request.SetHeader(header);
```

其他问题

版本升级

> **注意：**
> 从 3.0.x 版本升级到 3.1.x 版本有兼容性问题，对于 Integer 字段的使用修改为了 Long 类型，需要重新编译项目。

证书问题

证书问题通常是客户端环境配置错误导致的。SDK 没有对证书进行操作，依赖的是 Java 运行环境本身的处理。出现证书问题后，您可以使用 `-Djavax.net.debug=ssl` 开启详细日志辅助判断。

有用户报告使用 IBM JDK 1.8出现证书报错：`javax.net.ssl.SSLHandshakeException: Received fatal alert: handshake_failure`，使用 Oracle JDK 后问题消失。

kotlin 问题

部分用户可能使用时遇到报错：`java.lang.NoSuchMethodError: kotlin.collections.ArraysKt.copyInto`。这是因为 kotlin 运行环境版本较低所导致的，您可以尝试升级 kotlin 版本来解决此问题。

java.lang.NoSuchMethodError: xxx.setSkipSign 问题

部分用户可能使用时遇到报错：`java.lang.NoSuchMethodError: xxx.setSkipSign`。这是因为 `tencentcloud-sdk-java-common` 包和其他产品（如 `tencentcloud-sdk-java-cvm`）的版本不一致导致的。
该问题可能是 pom 中指定的 `common` 包版本有误，也可能是因为引用了其他第三方 sdk 而间接引用了不匹配的 `common` 版本导致的。
解决方式是在 pom.xml 中显式指定相同版本的 `common` 包版本。

```
<dependency>
    <groupId>com.tencentcloudapi</groupId>
    <artifactId>tencentcloud-sdk-java-common</artifactId>
    <version>3.1.1000</version>
</dependency>
```