

SDK 中心

Go



腾讯云

【 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 】



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。

Go

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），此 SDK 是云 API 3.0 平台的配套开发工具。

获取安装

依赖环境

1. Go 1.9 版本及以上（如使用 go mod 需要 Go 1.14）。
2. 注意：部分产品需在 [腾讯云控制台](#) 开通服务后才能使用。
3. 在腾讯云控制台 [访问管理](#) 页面获取密钥 SecretID 和 SecretKey，请务必妥善保管，或者使用更安全的临时安全凭证。

通过 go get 安装（推荐）

推荐使用腾讯云镜像加速下载：

1. Linux 或 MacOS:

```
export GOPROXY=https://mirrors.tencent.com/go/
```

2. Windows:

```
set GOPROXY=https://mirrors.tencent.com/go/
```

按需安装（推荐）

请注意，此安装方式仅支持使用 **Go Modules** 模式进行依赖管理，即环境变量 `GO111MODULE=auto` 或者 `GO111MODULE=on`，并且在您的项目中执行了 `go mod init xxx`。

安装指定产品 SDK（推荐）

v1.0.170 后可以按照产品下载，您只需下载基础包和对应的产品包（如 cvm）即可，不需要下载全部的产品，从而加快您构建镜像或者编译的速度：

1. 安装公共基础包：

```
go get -v -u github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common
```

2. 安装对应的产品包（如 cvm）：

```
go get -v -u github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/cvm
```

具体产品的包名缩写请参考 [products.md](#) 中的包名字段。

⚠ 注意：

- 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。
- 为了支持 go mod，SDK 版本号从 v3.x 降到了 v1.x。并于2021年5月10日移除了所有 v3.0.* 和 3.0.* 的 tag，如需追溯以前的 tag，请参考项目根目录下的 commit2tag 文件。

通过源码安装

前往代码托管地址 [Github](#) 或者 [Gitee](#) 下载最新代码，解压后安装到

```
$GOPATH/src/github.com/tencentcloud
```

 目录下。

快速开始

每个接口都有一个对应的 Request 结构和一个 Response 结构。例如云服务器的查询实例列表接口 DescribeInstances 有对应的请求结构体 DescribeInstancesRequest 和返回结构体 DescribeInstancesResponse。

下面以云服务器查询实例列表接口为例，介绍 SDK 的基础用法。

简化版

```
package main

import (
    "fmt"
    "os"
    "github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common"
    "github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common/errors"
    "github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common/profile"
    "github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common/regions"
    cvm "github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/cvm/v20170312"
)
```

```
func main() {  
    // 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。  
    // 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。  
    // credential := common.NewCredential("SecretId", "SecretKey")  
    credential := common.NewCredential(  
        os.Getenv("TENCENTCLOUD_SECRET_ID"),  
        os.Getenv("TENCENTCLOUD_SECRET_KEY"),  
    )  
    client, _ := cvm.NewClient(credential, regions.Guangzhou,  
profile.NewClientProfile())  
  
    request := cvm.NewDescribeInstancesRequest()  
    response, err := client.DescribeInstances(request)  
  
    if _, ok := err.(*errors.TencentCloudSDKError); ok {  
        fmt.Printf("An API error has returned: %s", err)  
        return  
    }  
    if err != nil {  
        panic(err)  
    }  
    fmt.Printf("%s\n", response.ToJsonString())  
}
```

详细版

出于演示的目的，有一些非必要的代码，例如对默认配置的修改，以尽量展示 SDK 的功能。在实际编写代码使用 SDK 的时候，建议尽量使用默认配置，酌情修改。

```
package main  
  
import (  
    "fmt"  
    "os"  
  
    "github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common"  
    "github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common/errors"  
    "github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common/profile"  
)
```

```
"github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/common/regions"

cvm "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/cvm/v20170312"
)

func main() {
    // 必要步骤:
    // 实例化一个认证对象，入参需要传入腾讯云账户密钥对 SecretId, SecretKey。
    // 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。
    // 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。
    // credential := common.NewCredential("SecretId", "SecretKey")
    credential := common.NewCredential(
        os.Getenv("TENCENTCLOUD_SECRET_ID"),
        os.Getenv("TENCENTCLOUD_SECRET_KEY"),
    )

    // 非必要步骤
    // 实例化一个客户端配置对象，可以指定超时时间等配置
    cpf := profile.NewClientProfile()
    // SDK默认使用POST方法。
    // 如果您一定要使用GET方法，可以在这里设置。GET方法无法处理一些较大的请求。
    // 如非必要请不要修改默认设置。
    cpf.HttpProfile.ReqMethod = "POST"
    // SDK有默认的超时时间，如非必要请不要修改默认设置。
    // 如有需要请在代码中查阅以获取最新的默认值。
    cpf.HttpProfile.ReqTimeout = 30
    // SDK会自动指定域名。通常是不需要特地指定域名的，但是如果您访问的是金融区的
    服务，
    // 则必须手动指定域名，例如云服务器的上海金融区域名： cvm.ap-shanghai-
    fsi.tencentcloudapi.com
    cpf.HttpProfile.Endpoint = "cvm.tencentcloudapi.com"
    // SDK默认用TC3-HMAC-SHA256进行签名，它更安全但是会轻微降低性能。
    // 如非必要请不要修改默认设置。
    cpf.SignMethod = "TC3-HMAC-SHA256"
    // SDK 默认用 zh-CN 调用返回中文。此外还可以设置 en-US 返回全英文。
    // 但大部分产品或接口并不支持全英文的返回。
    // 如非必要请不要修改默认设置。
    cpf.Language = "en-US"
    // 打印日志，默认是false
    // cpf.Debug = true
```

量

置，

冲突，先注释掉。

Values是string数组。

```
// 实例化要请求产品 (以cvm为例) 的client对象
// 第二个参数是地域信息，可以直接填写字符串ap-guangzhou，或者引用预设的常量

client, _ := cvm.NewClient(credential, regions.Guangzhou, cpf)
// 实例化一个请求对象，根据调用的接口和实际情况，可以进一步设置请求参数
// 您可以直接查询SDK源码确定DescribeInstancesRequest有哪些属性可以设置，

// 属性可能是基本类型，也可能引用了另一个数据结构。
// 推荐使用IDE进行开发，可以方便的跳转查阅各个接口和数据结构的文档说明。
request := cvm.NewDescribeInstancesRequest()

// 基本类型的设置。
// 此接口允许设置返回的实例数量。此处指定为只返回一个。
// SDK采用的是指针风格指定参数，即使对于基本类型您也需要用指针来对参数赋值。
// SDK提供对基本类型的指针引用封装函数
request.Limit = common.Int64Ptr(1)

// 数组类型的设置。
// 此接口允许指定实例 ID 进行过滤，但是由于和接下来要演示的 Filter 参数冲突，先注释掉。
// request.InstanceIds = common.StringPtrs([]string{"ins-
r8hr2upy"})

// 复杂对象的设置。
// 在这个接口中，Filters是数组，数组的元素是复杂对象Filter，Filter的成员Values是string数组。
request.Filters = []*cvm.Filter{
    &cvm.Filter{
        Name: common.StringPtr("zone"),
        Values: common.StringPtrs([]string{"ap-guangzhou-1"}),
    },
}

// 通过client对象调用想要访问的接口，需要传入请求对象
response, err := client.DescribeInstances(request)
// 处理异常
if _, ok := err.(*errors.TencentCloudSDKError); ok {
    fmt.Printf("An API error has returned: %s", err)
    return
}
// 非SDK异常，直接失败。实际代码中可以加入其他的处理。
if err != nil {
```

```
        panic(err)
    }
    // 打印返回的json字符串
    fmt.Printf("%s\n", response.ToJsonString())
}
```

更多示例参见 [examples](#) 目录。对于复杂接口的 Request 初始化例子，可以参考 [例一](#)。对于使用 json 字符串初始化 Request 的例子，可以参考 [例二](#)。

相关配置

⚠ 注意：

如无特殊需要，建议您使用默认配置。

在创建客户端前，如有需要，您可以通过修改 `profile.ClientProfile` 中字段的值进行一些配置。

```
// 非必要步骤
// 实例化一个客户端配置对象，可以指定超时时间等配置
cpf := profile.NewClientProfile()
```

具体的配置项说明如下：

请求方式

SDK 默认使用 POST 方法。如果您一定要使用 GET 方法，可以在这里设置。GET 方法无法处理一些较大的请求。

```
cpf.HttpProfile.ReqMethod = "POST"
```

超时时间

SDK 有默认的超时时间，如非必要请不要修改默认设置。如有需要请在代码中查阅以获取最新的默认值。单位：秒。

```
cpf.HttpProfile.ReqTimeout = 30
```

指定域名

SDK 会自动指定域名。通常是不需要特地指定域名的，但是如果您访问的是金融区的服务，则必须手动指定域名，例如云服务器的上海金融区域名：`cvm.ap-shanghai-fsi.tencentcloudapi.com`。

```
cpf.HttpProfile.Endpoint = "cvm.tencentcloudapi.com"
```

签名方式

SDK 默认用 `TC3-HMAC-SHA256` 进行签名，它更安全但是会轻微降低性能。

```
cpf.SignMethod = "HmacSHA1"
```

DEBUG 模式

DEBUG 模式会打印更详细的日志，当您需要进行详细的排查错误时可以开启。

默认为 `false`。

```
cpf.Debug = true
```

禁用长连接（Keep-alive）

SDK 的每一个 client 默认使用长连接模式，即请求的头部 `Connection` 字段的值为 `keep-alive`。

如果您需要使用短连接，可以按照以下方式进行设置：

```
...
client, _ := cvm.NewClient(credential, regions.Guangzhou, cpf)
tp := &http.Transport{
    DisableKeepAlives: true,
}
client.WithHttpTransport(tp)
...
```

则此 client 发出的每个请求头部的 `Connection` 字段的值为 `close`。

地域容灾

从 `v1.0.227` 开始，腾讯云 GO SDK 支持地域容灾功能：

当请求满足以下条件时：

1. 失败次数 ≥ 5 次
2. 失败率 $\geq 75\%$

SDK 会自动将您请求的地域设置为备选地域。

相关设置如下：

```
// 开启
```

```
cpf.DisableRegionBreaker = false
// 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
cpf.BackupEndpoint = "ap-guangzhou.tencentcloudapi.com"
```

此功能仅支持单个客户端的同步请求。

代理

如果是有代理的环境下，需要设置系统环境变量 `https_proxy`，否则可能无法正常调用，抛出连接超时的异常。或者自定义 Transport 指定代理，通过 `client.WithHttpTransport` 覆盖默认配置。

开启 DNS 缓存

当前 GO SDK 总是会去请求 DNS 服务器，而没有使用到 `nsd` 的缓存，可以通过导出环境变量 `GODEBUG=netdns=cgo`，或者 `go build` 编译时指定参数 `-tags 'netcgo'` 控制读取 `nsd` 缓存。

忽略服务器证书校验

虽然使用 SDK 调用公有云服务时，必须校验服务器证书，以识破他人伪装的服务器，确保请求的安全。但是某些极端情况下，例如测试时，您可能会需要忽略自签名的服务器证书。

以下是其中一种可能的方法：

```
import "crypto/tls"
...
client, _ := cvm.NewClient(credential, regions.Guangzhou, cpf)
tr := &http.Transport{
    TLSClientConfig: &tls.Config{InsecureSkipVerify: true},
}
client.WithHttpTransport(tr)
...
```

⚠ 注意：

再次强调，除非您知道自己在做什么，并明白由此带来的风险，否则不要尝试关闭服务器证书校验。

功能介绍

凭证管理

从版本 `v1.0.217` 开始，腾讯云 GO SDK 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：

```
provider := common.DefaultEnvProvider()
credential, err := provider.GetCredential()
```

2. 配置文件

配置文件路径为：

2.1 环境变量 `TENCENTCLOUD_CREDENTIALS_FILE` 所指定的路径。

2.2 Linux or MacOS: `~/.tencentcloud/credentials`。

2.3 Windows: `c:\Users\NAME\.tencentcloud\credentials`。

配置文件格式如下：

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

相关代码如下：

```
provider := common.DefaultProfileProvider()
credential, err := provider.GetCredential()
```

3. 角色扮演

有关角色扮演的概念请参见 [腾讯云角色概述](#)。

要使用此种方式，您必须在腾讯云访问管理控制台上创建了一个角色，具体创建过程请参见 [腾讯云角色创建](#)。

在您拥有角色后，可以通过如下方式获取凭证：

```
provider := common.DefaultRoleArnProvider(secretId, secretKey,
roleArn)
credential, err := provider.GetCredential()
```

4. 实例角色

有关实例角色的概念请参见 [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。相关代码如下：

```
provider := common.DefaultCvmRoleProvider()
credential, err := provider.GetCredential()
```

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](#)。

示例代码请参考 `examples/ssm/v20190923/get_secret_value.go`。

```
provider, err := common.DefaultTkeOIDCRoleArnProvider()
credential, err := provider.GetCredential()
```

6. 凭证提供链

腾讯云 GO SDK 提供了 凭证提供链，它会默认以 **环境变量->配置文件->实例角色** 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：

```
provider := common.DefaultProviderChain()
credential, err := provider.GetCredential()
```

您也可以自定义自己的凭证提供链，从而改变其调用顺序：

```
provider1 := common.DefaultCvmRoleProvider()
provider2 := common.DefaultEnvProvider()
customProviderChain := []common.Provider{provider1, provider2}
provider := common.NewProviderChain(customProviderChain)
credential, err := provider.GetCredential()
```

更详细的使用方式请参考示例：[使用ProviderChain](#)。

错误处理

从 `v1.0.181` 开始，腾讯云 GO SDK 会将各个产品的返回的错误码定义为常量，您可以直接调用处理，无需手动定义。如果您使用 IDE（如 Goland）进行开发，可以使用他们的代码提示功能直接选择。例如：

```
...//Your other go code

// Handling errors
response, err := client.DescribeInstances(request)
if terr, ok := err.(*errors.TencentCloudSDKError); ok {
    code := terr.GetCode()
    if code == cvm.FAILEDOPERATION_ILLEGALTAGKEY{
        fmt.Printf("Handling error: FailedOperation.IllegalTagKey,%s",
err)
    }else if code == cvm.UNAUTHORIZEDOPERATION{
        fmt.Printf("Handling error: UnauthorizedOperation,%s", err)
    }else{
```

```
    fmt.Printf("An API error has returned: %s", err)
}
return
}
```

...

同时，各个接口函数的注释部分也列出了此接口可能会返回的错误码，方便您进行处理：

```
// DescribeInstances
// 本接口 (DescribeInstances) 用于查询一个或多个实例的详细信息。
//
//
//
// * 可以根据实例`ID`、实例名称或者实例计费模式等信息来查询实例的详细信息。过滤信息详
细请见过滤器`Filter`。
//
// * 如果参数为空，返回当前用户一定数量（`Limit`所指定的数量，默认为20）的实例。
//
// * 支持查询实例的最新操作（LatestOperation）以及最新操作状态
(LatestOperationState)。
//
// 可能返回的错误码：
//  FAILEDOPERATION_ILLEGALTAGKEY = "FailedOperation.IllegalTagKey"
//  FAILEDOPERATION_ILLEGALTAGVALUE = "FailedOperation.IllegalTagValue"
//  FAILEDOPERATION_TAGKEYRESERVED = "FailedOperation.TagKeyReserved"
//  INTERNALSERVERERROR = "InternalServerError"
//  INVALIDFILTER = "InvalidFilter"
//  INVALIDFILTERVALUE_LIMITEXCEEDED =
"InvalidFilterValue.LimitExceeded"
//  INVALIDHOSTID_MALFORMED = "InvalidHostId.Malformed"
//  INVALIDINSTANCEID_MALFORMED = "InvalidInstanceId.Malformed"
//  INVALIDPARAMETER = "InvalidParameter"
//  INVALIDPARAMETERVALUE = "InvalidParameterValue"
//  INVALIDPARAMETERVALUE_IPADDRESSMALFORMED =
"InvalidParameterValue.IPAddressMalformed"
//  INVALIDPARAMETERVALUE_INVALIDIPFORMAT =
"InvalidParameterValue.InvalidIpFormat"
//  INVALIDPARAMETERVALUE_INVALIDVAGUENAME =
"InvalidParameterValue.InvalidVagueName"
//  INVALIDPARAMETERVALUE_LIMITEXCEEDED =
"InvalidParameterValue.LimitExceeded"
//  INVALIDPARAMETERVALUE_SUBNETIDMALFORMED =
"InvalidParameterValue.SubnetIdMalformed"
```

```
// INVALIDPARAMETERVALUE_TAGKEYNOTFOUND =  
"InvalidParameterValue.TagKeyNotFound"  
// INVALIDPARAMETERVALUE_VPCIDMALFORMED =  
"InvalidParameterValue.VpcIdMalformed"  
// INVALIDSECURITYGROUPID_NOTFOUND = "InvalidSecurityGroupId.NotFound"  
// INVALIDSGID_MALFORMED = "InvalidSgId.Malformed"  
// INVALIDZONE_MISMATCHREGION = "InvalidZone.MismatchRegion"  
// RESOURCENOTFOUND_HPCCLUSTER = "ResourceNotFound.HpcCluster"  
// UNAUTHORIZEDOPERATION_INVALIDTOKEN =  
"UnauthorizedOperation.InvalidToken"  
func (c *Client) DescribeInstances(request *DescribeInstancesRequest)  
(response *DescribeInstancesResponse, err error){  
    ...  
}
```

Common Client

从 v1.0.189 开始，腾讯云 GO SDK 支持使用 泛用型的 API 调用方式 (Common Client) 进行请求。您只需安装 common 包，即可向任何产品发起调用。

注意，您必须明确知道您调用接口的参数内容，否则会调用失败。

目前仅支持使用 POST 方式发送请求，且签名方法必须使用签名方法 v3。

详细使用请参阅示例：[使用 Common Client 进行调用](#)。

自定义 Header

RunInstancesRequest 示例

```
request := cvm.NewRunInstancesRequest()  
request.SetHeader(map[string]string{  
    "X-TC-TraceId": "ffe0c072-8a5d-4e17-8887-a8a60252abca",  
})
```

CommonRequest 示例

```
request := tchttp.NewCommonRequest("cvm", "2017-03-12",  
"DescribeZones")  
request.SetHeader(map[string]string{  
    "X-TC-TraceId": "ffe0c072-8a5d-4e17-8887-a8a60252abca",  
})
```

http 代理

DescribeInstances 示例

```
// with authentication
clientProfile.HttpProfile.Proxy =
"http://username:password@127.0.0.1:1080"
// without authentication
clientProfile.HttpProfile.Proxy = "http://127.0.0.1:1080"
```

请求重试

网络错误重试

当发生临时网络错误或超时，SDK可以被配置为自动重试。默认不开启。

通过 `ClientProfile` 配置重试次数和重试间隔时间。

❗ 说明：

通过反射检查 `Request` 结构体是否存在 `ClientToken` 字段，存在该字段则认为是幂等请求。幂等请求才会在网络错误时自动重试，非幂等请求会抛出异常，防止请求多次重放造成结果不一致。

```
package main

import (
    "github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common"
    "github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common/profile"
    "github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common/regions"
    cvm "github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/cvm/v20170312"
)

func main() {
    credential := common.NewCredential("secretId", "secretKey")
    prof := profile.NewClientProfile()
    prof.NetworkFailureMaxRetries = 3 // 定义最大重试次数
    prof.NetworkFailureRetryDuration = profile.ExponentialBackoff // 定义重试间隔时间
    client, _ := cvm.NewClient(credential, regions.Guangzhou, prof)

    // ...
```

```
}
```

更多用法参考 [测试文件](#)。

限频重试

当发生 API 限频时，SDK 可以被配置为自动重试。默认不开启。

通过 `ClientProfile` 配置重试次数和重试间隔时间。

```
package main

import (
    "github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common"
    "github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common/profile"
    "github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common/regions"
    cvm "github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/cvm/v20170312"
)

func main() {
    // ...

    prof := profile.NewClientProfile()
    prof.RateLimitExceededMaxRetries = 3
    // 定义最大重试次数
    prof.RateLimitExceededRetryDuration = profile.ExponentialBackoff
    // 定义重试间隔时间
    client, _ := cvm.NewClient(credential, regions.Guangzhou, prof)

    // ...
}
```

幂等标识符

当网络超时重试或限频重试开启时，会自动向请求中注入 `ClientToken` 参数（如果请求存在 `ClientToken` 字段且为空）。

当用户手动指定 `ClientToken` 时，会跳过注入流程。



说明：

注入的 `ClientToken` 在 100000/s 并发量以下提供全局唯一性。

空数组和 omitempty

SDK 使用 omitempty 标签来序列化您的 request 对象，这样可以避免上报空数组或空对象。然而，对于某些接口而言，长度为 0 的数组和 nil 数组是有区别的。如果您希望在请求中携带空数组，需要使用 Common Client 来发送请求，请参考 [示例](#)。

支持产品列表

参见 [产品列表文档](#)。