SDK 中心

C++

腾讯云

【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。

C++

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。为方便 C++ 开发者调试和接入腾讯云产品 API，这里向您介绍适用于 C++ 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 C++ SDK 并开始调用。

依赖环境

开通产品和密钥获取

1. 从 腾讯云控制台 开通相应产品。
2. 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。

编译器

请安装支持 C++ 11或更高版本的编译器，建议使用 GCC 4.8或以上版本。

编译工具

CMake

您需要安装 CMake 3.12或更高版本。您可以参考 官方安装文档 进行安装，或者使用包管理器来安装。

- Ubuntu：

```
sudo apt-get install cmake
```

- CentOS：

```
yum install cmake3
```

- macOS：

```
brew install cmake
```

- Windows（使用 Chocolatey）：

```
choco install cmake.install
```

如果您的操作系统版本较低，自带的包管理器可能无法方便地升级 CMake。在这种情况下，您可以考虑使用 pip 进行安装：`pip install --upgrade pip && pip install cmake`。

vcpkg（仅 Windows 平台需要）

tencentcloud-sdk-cpp 在 Windows 平台使用 vcpkg 下载所依赖的程序库，安装说明请参见 安装 vcpkg。

依赖库

libcurl

详情请参见 libcurl。安装示例如下：

> **说明：**
> 建议安装最新版本的 libcurl 库，以避免可能存在的 libcurl 库内存泄漏问题。

- Ubuntu：

```
sudo apt-get install libcurl4-openssl-dev
```

- CentOS：

```
yum install libcurl-devel
```

- macOS（本身自带 curl，这一步非必须）：

```
brew install curl
```

- Windows（根据实际环境选择适当的 CPU 架构）：

```
vcpkg install curl:x64-windows
```

OpenSSL

详情请参见 OpenSSL。安装示例如下：

- Ubuntu：

```
sudo apt-get install libssl-dev
```

- CentOS：

```
yum install openssl-devel
```

- macOS：

```
brew install openssl && brew link openssl
```

- Windows（根据实际环境选择适当的 CPU 架构）：

```
vcpkg install openssl:x64-windows
```

从源代码构建 SDK

安装指定产品 SDK（推荐）

1. 前往 Github 仓库 或者 Gitee 仓库 下载最新代码。
2. 进入 SDK 目录，创建生成所需的构建文件。

Linux / macOS

```
# build
# 通过 BUILD_MODULES 指定产品编译，使用分号;分隔（可选）
./build.sh build -DBUILD_MODULES="cvm;cbs"

# install
./build.sh install
```

Windows (PowerShell)

```
# 允许执行 powershell 脚本
Set-ExecutionPolicy Bypass -Scope Process

# 64位build
# 通过 BUILD_MODULES 指定产品编译，使用分号;分隔（可选）
# 通过 CMAKE_TOOLCHAIN_FILE 指定 vcpkg 目录（必须）
```

```
.\build.ps1 build -DBUILD_MODULES="cvm;cbs" -
DCMAKE_TOOLCHAIN_FILE='[path to vcpkg]/scripts/buildsystems/vcpkg.cmake'

# 32位build
# 通过 BUILD_MODULES 指定产品编译，使用分号;分隔（可选）
# 通过 CMAKE_TOOLCHAIN_FILE 指定 vcpkg 目录（必须）
# 通过 'Visual Studio 17 2022' 指定 visual studio 版本，参考
https://cmake.org/cmake/help/latest/manual/cmake-
generators.7.html#visual-studio-generators
.\build32.ps1 build -DBUILD_MODULES="cvm;cbs" -
DCMAKE_TOOLCHAIN_FILE='[path to vcpkg]/scripts/buildsystems/vcpkg.cmake'
'Visual Studio 17 2022'

# install，需要 Administrator 权限
.\build.ps1 install
```

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

指定参数 `-DBUILD_MODULES_ALL=on` 。
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。
从 3.0.387 版本开始，默认将不再编译所有产品。因为对于低版本编译器将需要约 8GB 内存才能编译完成，且未来随着产品和接口的增长，内存需求会逐渐增加。

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。
- 通过指定编译选项，可以控制部分编译行为。注意：某些选项的变更需要删除 `sdk_build` 目录后才会生效。更多选项可以参考 `CMakeLists.txt` 文件。
- 生成静态库文件： `cmake -DBUILD_SHARED_LIBS=off ..`
- 如果产品列表过长，以上参数都不方便使用，可以直接编辑根路径下的 `CMakeLists.txt` 文件，关闭不需要产品的编译，例如云服务器 cvm，用 `#` 符号注释掉 `add_subdirectory(cvm)` 及附近相关代码，改为 `#add_subdirectory(cvm)` 。

使用 C++ SDK 示例

本文以云服务器 CVM 产品的 DescribeInstances 接口为例：

简化版

```
#include <tencentcloud/core/TencentCloud.h>
```

```cpp
#include <tencentcloud/core/Credential.h>
#include <tencentcloud/cvm/v20170312/CvmClient.h>
#include <tencentcloud/cvm/v20170312/model/DescribeInstancesRequest.h>
#include <tencentcloud/cvm/v20170312/model/DescribeInstancesResponse.h>
#include <tencentcloud/cvm/v20170312/model/Instance.h>

#include <iostream>
#include <string>
#include <cstdlib>

using namespace TencentCloud;
using namespace TencentCloud::Cvm::V20170312;
using namespace TencentCloud::Cvm::V20170312::Model;
using namespace std;

int main()
{
    TencentCloud::InitAPI();

    // 前往 https://console.cloud.tencent.com/cam/capi 获取 API 密钥
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。
    Credential cred = Credential(getenv("TENCENTCLOUD_SECRET_ID"),
                                 getenv("TENCENTCLOUD_SECRET_KEY"));

    DescribeInstancesRequest req = DescribeInstancesRequest();

    CvmClient cvm_client = CvmClient(cred, "ap-guangzhou");

    auto outcome = cvm_client.DescribeInstances(req);
    if (!outcome.IsSuccess())
    {
        cout << outcome.GetError().PrintAll() << endl;
        TencentCloud::ShutdownAPI();
        return -1;
    }

    DescribeInstancesResponse rsp = outcome.GetResult();
    cout<<"RequestId="<<rsp.GetRequestId()<<endl;
    cout<<"TotalCount="<<rsp.GetTotalCount()<<endl;
    if (rsp.InstanceSetHasBeenSet())
    {
```

```
        vector<Instance> instanceSet = rsp.GetInstanceSet();
        for (auto itr=instanceSet.begin(); itr!=instanceSet.end();
++itr)
        {
            cout<<(*itr).GetPlacement().GetZone()<<endl;
        }
    }

    TencentCloud::ShutdownAPI();

    return 0;
}
```

详细版

```
#include <tencentcloud/core/TencentCloud.h>
#include <tencentcloud/core/profile/HttpProfile.h>
#include <tencentcloud/core/profile/ClientProfile.h>
#include <tencentcloud/core/Credential.h>
#include <tencentcloud/core/NetworkProxy.h>
#include <tencentcloud/core/AsyncCallerContext.h>
#include <tencentcloud/cvm/v20170312/CvmClient.h>
#include <tencentcloud/cvm/v20170312/model/DescribeInstancesRequest.h>
#include <tencentcloud/cvm/v20170312/model/DescribeInstancesResponse.h>
#include <tencentcloud/cvm/v20170312/model/Instance.h>

#include <cstdlib>
#include <iostream>
#include <string>

using namespace TencentCloud;
using namespace TencentCloud::Cvm::V20170312;
using namespace TencentCloud::Cvm::V20170312::Model;
using namespace std;

int main()
{
    TencentCloud::InitAPI();

    // use the sdk
    // 前往 https://console.cloud.tencent.com/cam/capi 获取 API 密钥
SecretId SecretKey
```

```
    // 实例化一个认证对象，入参需要传入腾讯云账户 SecretId 和 SecretKey，切勿将密
钥泄露给他人
    // 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。
    Credential cred = Credential(getenv("TENCENTCLOUD_SECRET_ID"),
                                 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 = HttpProfile();
    httpProfile.SetKeepAlive(true);  // 状态保持，默认是False
    httpProfile.SetEndpoint("cvm.ap-guangzhou.tencentcloudapi.com");  //
指定接入地域域名(默认就近接入)
    httpProfile.SetReqTimeout(30);  // 请求超时时间，单位为秒(默认60秒)
    httpProfile.SetConnectTimeout(30); // 响应超时时间，单位是秒(默认是60秒)

    ClientProfile clientProfile = ClientProfile(httpProfile);

    DescribeInstancesRequest req = DescribeInstancesRequest();

    Filter respFilter;
    respFilter.SetName("zone");
    respFilter.SetValues({ "ap-guangzhou-1", "ap-guangzhou-2" });
    req.SetFilters({ respFilter });
    req.SetOffset(0);
    req.SetLimit(5);

    CvmClient cvm_client = CvmClient(cred, "ap-guangzhou",
clientProfile);

    // set proxy
    // NetworkProxy proxy = NetworkProxy(NetworkProxy::Type::HTTP,
"localhost.proxy.com", 8080);
    // cvm_client.SetNetworkProxy(proxy);

    auto outcome = cvm_client.DescribeInstances(req);
    if (!outcome.IsSuccess())
    {
        cout << outcome.GetError().PrintAll() << endl;
        TencentCloud::ShutdownAPI();
        return -1;
    }
    DescribeInstancesResponse rsp = outcome.GetResult();
    cout<<"RequestId="<<rsp.GetRequestId()<<endl;
```

```
    cout<<"TotalCount="<<rsp.GetTotalCount()<<endl;
    if (rsp.InstanceSetHasBeenSet())
    {
        vector<Instance> instanceSet = rsp.GetInstanceSet();
        for (auto itr=instanceSet.begin(); itr!=instanceSet.end();
++itr)
        {
            cout<<(*itr).GetPlacement().GetZone()<<endl;
        }
    }

    TencentCloud::ShutdownAPI();

    return 0;
}
```

demo 用例编译执行：

```
cd example/cvm/v20170312
mkdir build
cd build
# centos 下使用 cmake3 ..
cmake ..
make
./DescribeInstances
```

如果在执行过程中遇到动态库找不到的错误，请指定动态库路径。例如 libtencentcloud-sdk-cpp-core.so 安装到了 `/usr/local/lib` 路径下（在 CentOS 下可能是 `/usr/local/lib64` ），请将该路径追加到 `LD_LIBRARY_PATH` 环境变量中：

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib
./DescribeInstances
```

如果安装的是静态库，您需要修改 `example/cvm/v20170312/CMakeLists.txt` 文件，在链接库的配置代码中追加链接库：

```
target_link_libraries(DescribeInstances tencentcloud-sdk-cpp-cvm
tencentcloud-sdk-cpp-core -lcrypto -lcurl)
target_link_libraries(DescribeInstancesAsync tencentcloud-sdk-cpp-cvm
tencentcloud-sdk-cpp-core -lcrypto -lcurl)
```

更多示例请参见 example 目录。

Common Client 调用方式

从版本3.0.297开始，腾讯云 C++ 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 tencentcloud-sdk-cpp-core 基础依赖库，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。

Common Client 的示例请参见 example。

C++ SDK 支持压缩协议使用方式

从版本 3.0.622 开始，C++ SDK 已支持压缩协议请求，如需启用，请按照以下步骤进行操作：

1. 安装通用压缩库，安装示例如下：
   - Ubuntu

```
sudo apt-get install zlib1g-dev
```

   - CentOS

```
yum install -y zlib zlib-devel
```

   - macOS

```
brew install zlib
```

   - Windows（根据实际环境选择适当的 CPU 架构）

```
vcpkg install zlib:x64-windows
```

2. 通过指定编译选项，可以选择压缩模块是否进行编译，默认关闭。如需开启，将对应模块打开即可，如下所示：

```
cmake -DENABLE_COMPRESS_MODULE=on ..
```

单元测试

依赖库 gtest

安装示例如下：

```
git clone https://github.com/google/googletest
cd googletest
checkout release-1.10.0
mkdir build && cd build
# centos 改为 cmake3 ..
cmake ..
make
sudo make install
```

配置环境变量

环境变量	描述	获取方式
TENCENTCLOUD_SECRET_ID	云 API 密钥 SecretId	请前往 API 密钥管理 获取
TENCENTCLOUD_SECRET_KEY	云 API 密钥 SecretKey	请前往 API 密钥管理 获取

测试

执行以下脚本：

```
sh function_test.sh
```

> ⚠ **注意：**
> 如果您使用的是 CentOS 操作系统，请将 `function_test.sh` 中的 `cmake` 改为 `cmake3`，或者在环境变量中设置别名 `alias`。