1. 简介

1.1 前言

非常感谢您使用我们公司的产品，我们将竭诚为您提供最好的服务。
本手册可能包含技术上不准确的地方或文字错误。
本手册的内容将做定期的更新，恕不另行通知,更新的内容将会在本手册的新版本中加入。
我们随时会改进或更新本手册中描述的产品或程序。

当您阅读该开发手册时，同时应该拿到以下内容：
1.HttpUtillib（以下简称 OpenAPI 安全认证库（C++））开发包，包含依赖库（导出库为 HttpUtillib.dll 和 HttpUtillib.lib），使用说明文档。
2.VS2015 版本 HttpUtillib 的使用 Demo。
3. HttpUtillib 源代码。

1.2 OpenAPI 安全认证库（C++）目录结构

解压 OpenAPI 安全认证库（C++），其目录结构如下图所示：

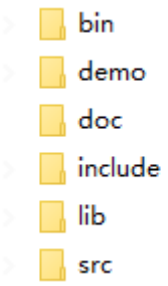

图 1.2-1 OpenAPI 安全认证库（C++）目录结构

各目录存放文件如下表：

表 1.2-1 OpenAPI 安全认证库

文件夹	内容
bin	HttpUtillib.dll 以及相关依赖库、Http_ApiDemo.exe
demo	HttpUtillib demo 源码
doc	OpenAPI 安全认证库（C++）开发指南文档
include	HttpUtillib 导出头文件
lib	HttpUtillib 导出 lib 库
src	HttpUtillib 源码

1.3 OpenAPI 安全认证库（C++）简介

OpenAPI 安全认证库封装了 HTTP/HTTPS 的 POST 方法，提供 C++调 OpenAPI HTTP 接口统一的入口。OpenAPI 安全认证库屏蔽了 OpenAPI HTTP 接口签名细节，降低 C++对接 OpenAPI 的复杂度，使用方只需

引入 OpenAPI 安全认证库（C++）即可方便快速的实现 HTTP 通信。

1.4 运行环境

- 支持 Win7、Win8、Win10 32/64 位操作系统

1.5 约束说明

- 只支持 C++对接

1.6 更新说明

V1.0.0

1、根据 OpenAPI 安全认证协议，封装 HTTP/HTTPS 请求安全认证过程。

V1.1.0

1、优化 HTTP POST 请求性能。

V1.1.1

1、接口内部封装 content-md5 请求头，提升 POST 请求安全性。
2、接口内部新增重定向功能，支持重定向请求。

2. 接口定义

2.1 HTTP POST 字符串

接口名称：

```
string STDCALL HttpPost(string url,
                        map<string, string> headers,
                        string body,
                        string appKey,
                        string appSecret,
                        int timeout,
                        list<string> signHeaderPrefixList);
```

接口描述：

封装 HTTP POST 请求，提供 HTTP POST 请求统一入口。

参数说明：

[in] url：POST 请求的 URL，如 http(s)://10.33.31.9:8001/artemis/api/vms/v1/videoParam
[in] headers：请求头，如可指定 accept 为 application/json，指定 content-type 为 application/json;charset=UTF-8
[in] body：POST 请求体
[in] appKey：合作方 APPKey

[in] appSecret：合作方 APPSecret
[in] timeout：请求超时时间，单位：秒
[in] signHeaderPrefixList：签名附属参数列表，一般不需指定

返回值：
返回请求结果。

备注：
无。

示例（使用 HTTPS 协议）：
详见基于 OpenAPI 安全认证库（C++）接口的使用示例。

3. OpenAPI 安全认证库（C++）使用说明

3.1 使用前提

- 基于 Visual Studio 开发
- 已获取到综合安防管理平台 IP 地址、端口号以及 APPKey、APPSecret。如未获取到相关信息，可联系综合安防管理平台系统管理员获取，或通过如何获取 APPKey 和 APPSecret 获取 APPKey 和 APPSecret。

3.2 使用 OpenAPI 安全认证库（C++）

OpenAPI 安全认证库提供基于 Visual Studio 2015 编译好的 DLL，也提供了源码供使用者自行编译，自行编译时请选择 Release 版本编译。需要注意的是，当 OpenAPI 安全认证库与视频客户端插件配合使用时，若自行修改编译过 OpenAPI 安全认证库源码，请参考附录 4.4 OpenAPI 安全认证库（C++）与视频客户端插件配合使用。本节描述如何使用源码编译成 DLL 以及基于该 DLL 如何使用。

3.2.1 基于源码生成 DLL 文件

OpenAPI 安全认证库（C++）提供基于 Visual Studio 2015 的源码，基于源码生成 DLL 文件步骤如下：

1、 解压 OpenAPI 安全认证库（C++），打开 src 源码目录中 VS2015 下的解决方案文件 HttpUtillib.sln（如打开解决方案文件时因使用方 VS 版本较高需要升级，可能存在警告或者错误的情况，请自行根据升级结果文档解决），如下图：

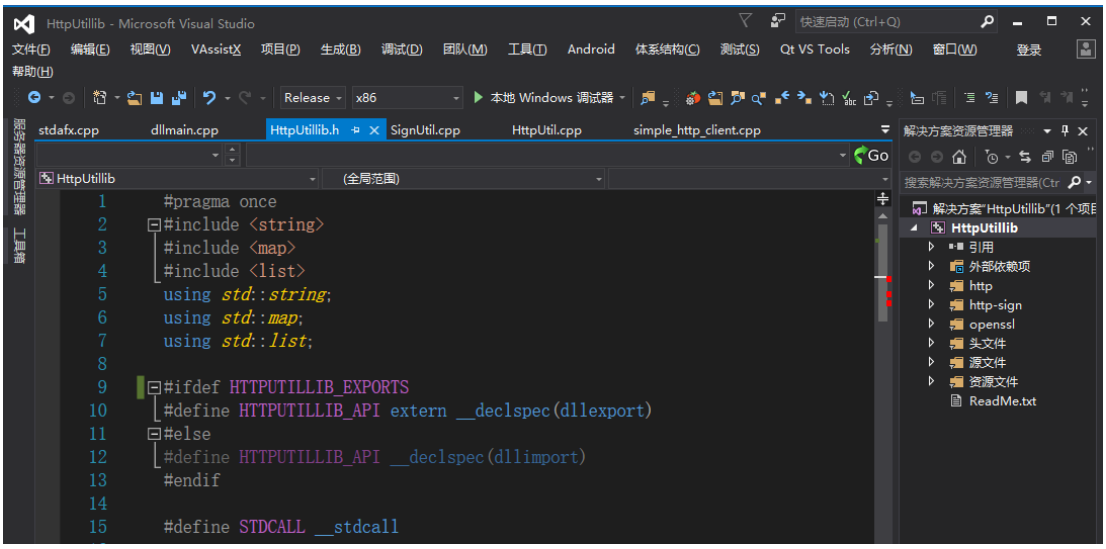

图 3.2.1-1 HttpUtillib 工程

2、 选择 Release 模式下的平台。如果拿到的是 Win32 版本，请选择 x86，如果拿到的是 Win64 版本，请选择 x64。这里选择 x86。选择“生成”->“生成 HttpUtillib”（或英文模式下的“Build”->“Build HttpUtillib”），生成的 DLL 位于 bin 目录下 VS2015 中，生成的 lib 文件位于 lib 文件夹中 VS2015 中。生成 HttpUtillib 以及生成结果如下图：

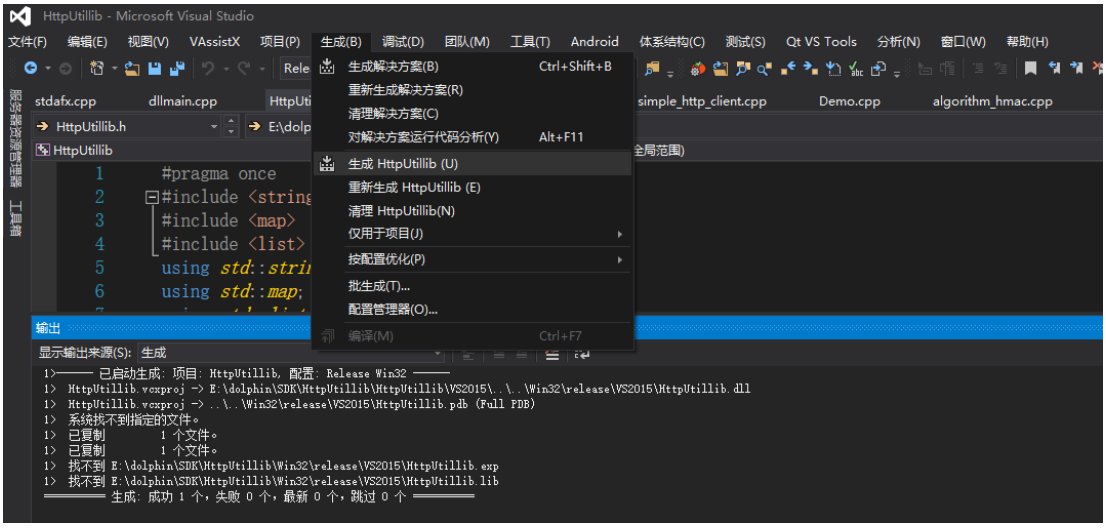

图 3.2.1-2 生成 HttpUtillib

3.2.2 使用 Demo

OpenAPI 安全认证库（C++）提供基于 Visual Studio 2015 的 demo 源码，并已配置好依赖文件（依赖 lib 目录下的 lib 文件以及 include 目录下的头文件）以及输出目录（输出目录位于 bin 的 VS2015 下）。基于 Demo 源码编译步骤如下：

1、 解压 OpenAPI 安全认证库（C++），选择 demo 目录 VS2015 下的 demo 源码，这里选择 VS2015 版本的源文件，在 VS2015 下 HttpUtillib 目录中打开解决方案文件 Http_ApiDemo.sln（如打开解决方案文件时因使用方 VS 版本较高需要升级，可能存在警告或者错误的情况，请自行根据升级结果文档解决），如下图：

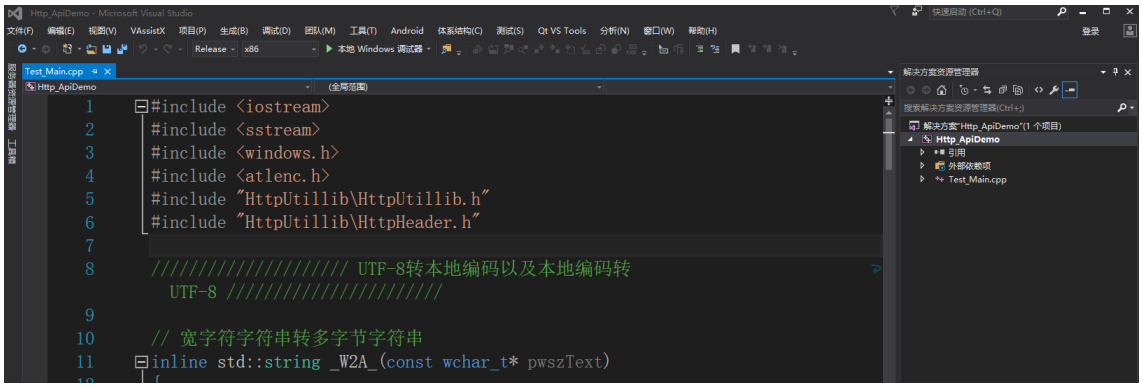

图 3.2.2-1 HttpUtillib Demo 工程

2、 选择 Release 模式下的平台。如果拿到的是 Win32 版本，请选择 x86，如果拿到的是 Win64 版本，请选择 x64。这里选择 x86。选中“解决方案 Http_ApiDemo”下的项目“Http_ApiDemo”，右键后在弹出的菜单中选择“属性”，如下图：

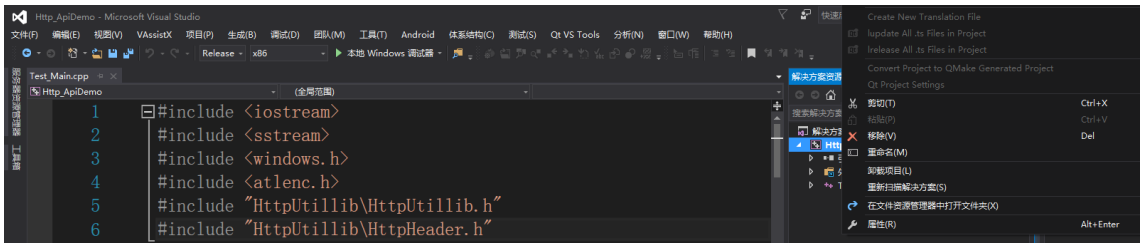

图 3.2.2-2 HttpUtillib Demo 工程属性

3、 配置依赖头文件目录。在项目属性页中选择“C/C++”->“常规”->“添加包含目录”（或英文模式下的“C/C++”->“General”->“Additional Include Directories”）添加依赖头文件目录（“.\..\..include”为使用相对目录，“.\”表示 Http_ApiDemo.vcxproj 文件同级目录），如下图所示：

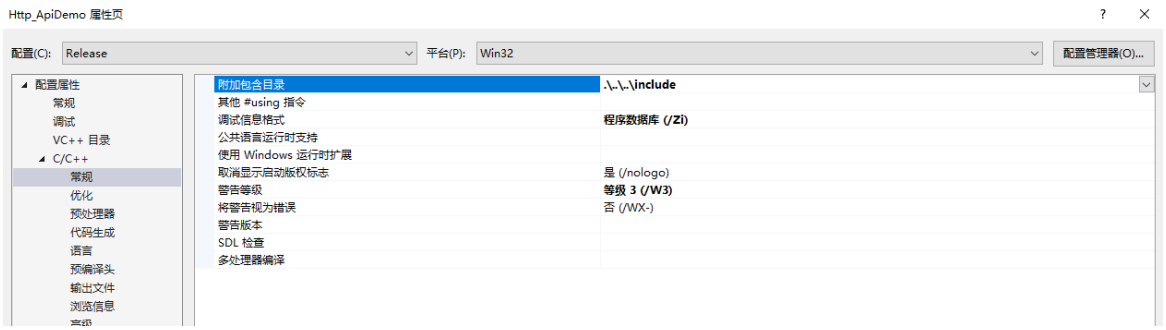

图 3.2.2-3 HttpUtillib Demo 工程属性配置附加包含目录

4、 配置依赖 lib 库文件目录。在项目属性页中选择“链接器”->“常规”->“附加库目录”（或英文模式下的“Linker”->“General”->“Additional Library Directories”）添加依赖 lib 库文件目录（“.\..\..\lib\VS2015”为使用相对目录，“.\”表示 Http_ApiDemo.vcxproj 文件同级目录），如下图所示：

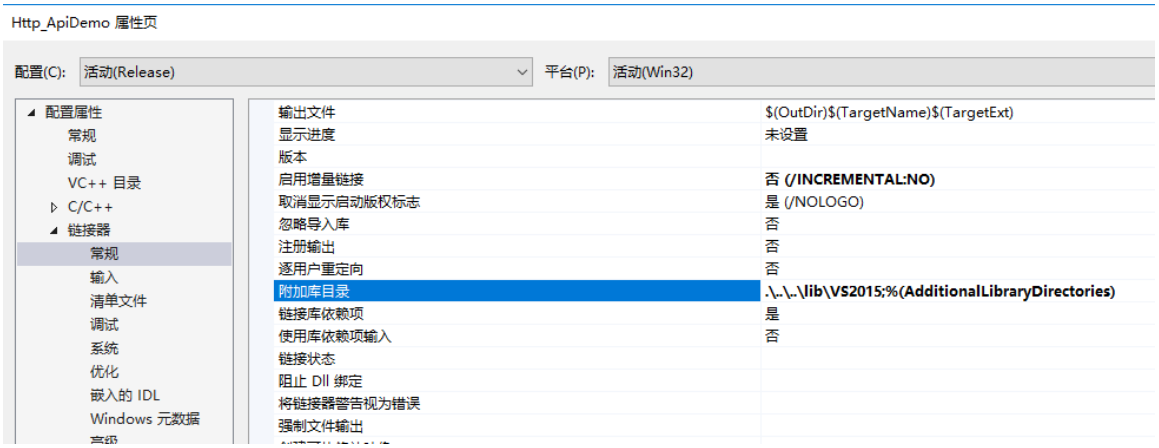

图 3.2.2-4 HttpUtillib Demo 工程属性配置附加库目录

5、 配置依赖库。在项目属性页中选择“链接器”->“输入”->“附加依赖项”（或英文模式下的“Linker”->“Input”->“AdditionalDependencies”）配置依赖 lib 库名称，如下图所示：

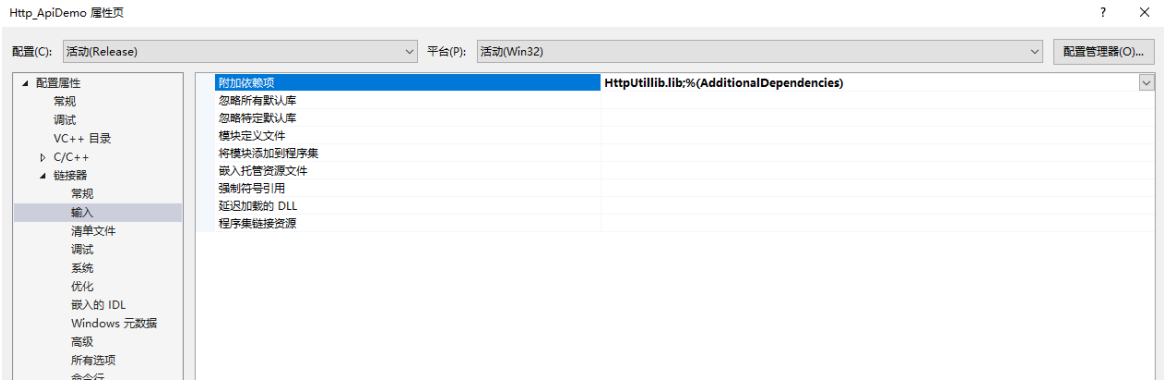

图 3.2.2-5 HttpUtillib Demo 工程属性配置附加依赖库

6、 选择“生成”->“生成 Http_ApiDemo”（或英文模式下的）“Build”->“Build Http_ApiDemo”，生成的 EXE 位于 bin 目录下 VS2015/中。生成 Http_ApiDemo 以及生成结果如下图：

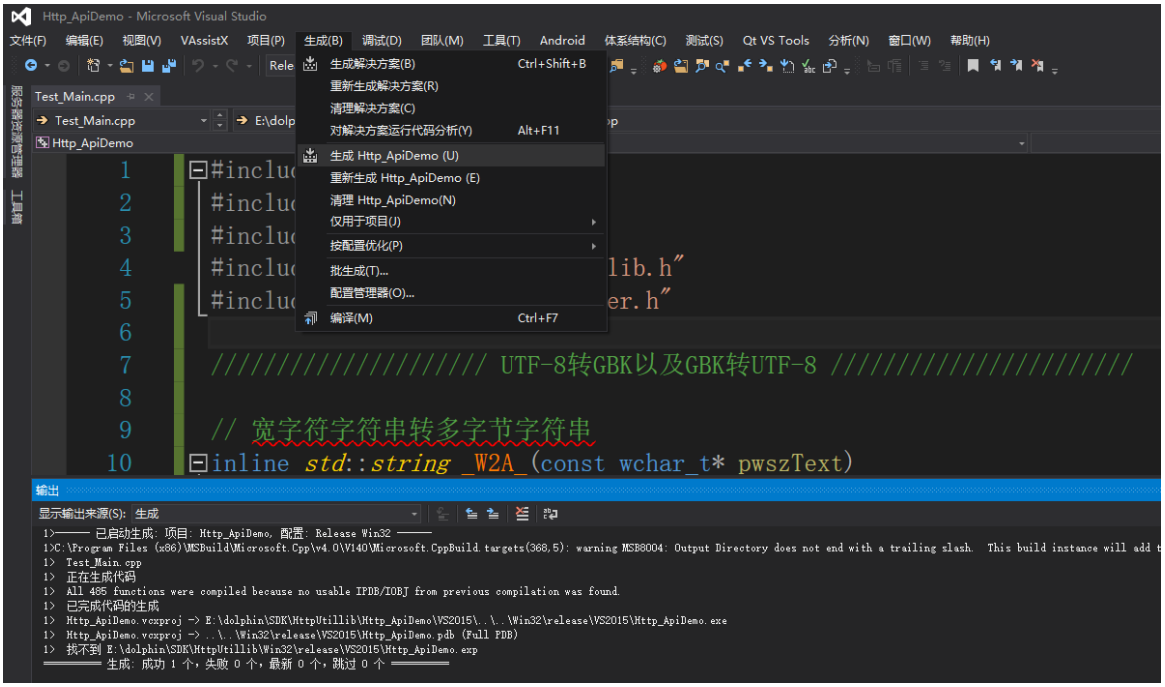

图 3.2.2-2 生成 HttpUtillib Demo

3.3 基于 OpenAPI 安全认证库（C++）接口的使用示例

以 OpenAPI 中“根据编号获取监控点信息”接口（接口详细信息请访问：https://open.hikvision.com/docs/93d926415b62e0f76290e6a63cd6facb#b0bf8cf1）为例，展示如何使用 OpenAPI 安全认证库（C++）HttpPost 接口来根据监控点编号获取监控点信息。完整示例代码详见 Http_ApiDemo。

1、 引入头文件

```
#include <windows.h>
#include <iostream>
#include <sstream>
#include "HttpUtillib\HttpUtillib.h"
#include "HttpUtillib\HttpHeader.h"
```

2、 根据监控点编号获取监控点信息

```
// Step1：根据实际综合安防管理平台情况设置IP地址、端口号、APPKey和APPSecret（如何获取APPKey和APPSecret）
Unit_query_t query;
query.appKey = "123456";          // APPKey
query.appSecret = "123456789";    // APPSecret
query.artemisIp = "10.10.10.10";// 综合安防管理平台IP地址
query.artemisPort = 443;          // 综合安防管理平台端口（使用HTTP或HTTPS协议端口，默认端口分别为80和443）

// Step2：组装POST请求URL（以HTTPS协议为例）
std::stringstream  ss;
ss <<"https://"<<query.artemisIp<<":"<<query.artemisPort<< "/artemis/api/resource/v1/cameras/indexCode";
std::string szUrl =ss.str();
```

```
// Step3：根据期望的Response内容类型组装请求头
map<string, string> headers;
headers.insert(std::make_pair(HttpHeader::HTTP_HEADER_ACCEPT, "application/json"));
headers.insert(std::make_pair(HttpHeader::HTTP_HEADER_CONTENT_TYPE, "application/json;charset=UTF-8"));

// Step4：请求超时时间与自定义参与签名参数列表
int iTimeOut = 15;
list<string> signHeaderPrefixList;

// Step5：组装body(此处直接组装字符串，也可以使用json来组装，报文较复杂时，建议使用json来组装)
string szCameraIndexCode = "748d84750e3a4a5bbad3cd4af9ed5101";
string szBody = "{\"cameraIndexCode\": \"" + szCameraIndexCode + "\"}";

// Step6：对body转成UTF-8编码
string szUtf8Body = _A2U8(szBody.c_str());   // 宏_A2U8详见本地编码转换UTF-8编码示例

// Step7：发起POST请求
std::string szResponse = HttpPost(szUrl, headers, szBody, query.appKey, query.appSecret, iTimeOut,
signHeaderPrefixList);
// 或使用 HttpPost 的另一种形式：std::string szResponse = HttpPost(szUrl, headers, szBody.c_str(),
query.appKey, query.appSecret, iTimeOut, signHeaderPrefixList);

// Step8：将响应转成本地编码（中文操作系统下认为是 GBK）
string szLocal = _U82A(szResponse.c_str());  // 宏_U82A 详见 UTF-8 编码转换本地编码示例

// Step9：解析 szLocal 中的 json 报文
```

4. 附录

4.1 如何获取 APPKey 和 APPSecret

请查看综合安防管理平台“文档中心”（“http(s):\\IP:PORT/artemis-portal/document”，其中 IP 为综合安防管理平台 IP 地址，端口 PORT 为综合安防管理平台端口，如 http://10.19.132.186/artemis-portal/document）的“开发前准备”章节。

4.2 UTF-8 编码转换成本地编码示例

```
// 宽字符字符串转多字节字符串
inline std::string _W2A_(const wchar_t* pwszText)
{
    if (pwszText == NULL || wcslen(pwszText) == 0)
    {
        return std::string();
```

```
    }
    int iSizeInBytes = WideCharToMultiByte(CP_ACP, 0, pwszText, -1, NULL, 0, NULL, NULL);
    char* pMultiByte = new(std::nothrow) char[iSizeInBytes];
    if (pMultiByte == NULL)
    {
        return std::string();
    }

    memset(pMultiByte, 0, iSizeInBytes);
    WideCharToMultiByte(CP_ACP, 0, pwszText, -1, pMultiByte, iSizeInBytes, NULL, NULL);

    std::string strResult = std::string(pMultiByte);
    delete[] pMultiByte;
    pMultiByte = NULL;
    return strResult;
}

// UTF-8字符串转宽字符字符串
inline std::wstring _U82W_(const char* pszText)
{
    if (pszText == NULL || strlen(pszText) == 0)
    {
        return std::wstring();
    }
    int iSizeInChars = MultiByteToWideChar(CP_UTF8, 0, pszText, -1, NULL, 0);
    wchar_t* pWideChar = new(std::nothrow) wchar_t[iSizeInChars];
    if (pWideChar == NULL)
    {
        return std::wstring();
    }

    wmemset(pWideChar, 0, iSizeInChars);
    MultiByteToWideChar(CP_UTF8, 0, pszText, -1, pWideChar, iSizeInChars);

    std::wstring strResult = std::wstring(pWideChar);
    delete[] pWideChar;
    pWideChar = NULL;
    return strResult;
}

// UTF-8字符串转多字节字符串
inline std::string _U82A_(const char* pszText)
{
    return _W2A_(_U82W_(pszText).c_str());
```

```
}

// 定义 UTF-8 转换至多字节宏_U82A
#define _U82A(lpu8Text) (const_cast<char*>(_U82A_(static_cast<const char*>(lpu8Text)).c_str()))
```

4.3 本地编码转换 UTF-8 编码示例

```
// 多字节字符串转宽字符字符串
inline std::wstring _A2W_(const char* pszText)
{
    if (pszText == NULL || strlen(pszText) == 0)
    {
        return std::wstring();
    }
    int iSizeInChars = MultiByteToWideChar(CP_ACP, 0, pszText, -1, NULL, 0);
    wchar_t* pWideChar = new(std::nothrow) wchar_t[iSizeInChars];
    if (pWideChar == NULL)
    {
        return std::wstring();
    }

    wmemset(pWideChar, 0, iSizeInChars);
    MultiByteToWideChar(CP_ACP, 0, pszText, -1, pWideChar, iSizeInChars);

    std::wstring strResult = std::wstring(pWideChar);
    delete[] pWideChar;
    pWideChar = NULL;
    return strResult;
}

// 宽字符字符串转UTF-8字符串
inline std::string _W2U8_(const wchar_t* pwszText)
{
    if (pwszText == NULL || wcslen(pwszText) == 0)
    {
        return std::string();
    }
    int iSizeInBytes = WideCharToMultiByte(CP_UTF8, 0, pwszText, -1, NULL, 0, NULL, NULL);
    char* pUTF8 = new(std::nothrow) char[iSizeInBytes];
    if (pUTF8 == NULL)
    {
        return std::string();
    }
```

```
    memset(pUTF8, 0, iSizeInBytes);
    WideCharToMultiByte(CP_UTF8, 0, pwszText, -1, pUTF8, iSizeInBytes, NULL, NULL);

    std::string strResult = std::string(pUTF8);
    delete[] pUTF8;
    pUTF8 = NULL;
    return strResult;
}

// 多字节字符串转UTF-8字符串
inline std::string _A2U8_(const char* pszText)
{
    return _W2U8_(_A2W_(pszText).c_str());
}

// 定义多字节转换至 UTF-8 宏_A2U8
#define _A2U8(lpszText) (const_cast<char*>(_A2U8_(static_cast<const char*>(lpszText)).c_str()))
```

4.4 OpenAPI 安全认证库（C++）与视频客户端插件配合使用

视频客户端插件（详见 https://open.hikvision.com/download/5c67f1e2f05948198c909700?type=10）已封装了基于 VS2015 的 OpenAPI 安全认证库（C++）Win32 版本，如需使用自行编译的安全认证库，请基于 Win32 的 VS2015 版本编译，并保证接口不变，将编译后的 dll 替换至解压视频客户端插件后的 bin 目录中。