本文主要介绍在已开通ONS服务的基础上，如何使用ONSClient4cpp Linux C++和Windows C++/.NET版本发送和订阅消息，已发送和订阅的消息可以在MQ控制台进行查询。相关MQ产品背景，专业术语，功能特性/业务场景，以及如何开通ONS服务和如何查询消息，请参考：官网介绍。

MQ多语言SDK

目前MQ支持Linux C++，Windows C++, Windows .NET等跨平台多语言版本，SDK下载和使用说明见SDK下载说明

MQ 多语言SDK目录结构

include目录

包含实现发送和订阅功能所需的接口文件，更多接口可以通过查询头文件的方式获取。

> .net SDK不需要`include`目录。

lib目录

- Linux C++版本

  自`2016.12.2`开始Linux CPP版本依赖了高性能boost库(`1.62.0`版本)，不仅降低了CPU资源占用率，而且提高了运行效率;目前主要依赖了`boost_system`，`boost_thread`，`boost_chrono`，`boost_filesystem`四个库；我们提供了静态库和动态库两种解决方案：

  1. 静态解决方案:

     MQ库文件在**lib/lib-boost-static**目录下，boost库静态链接到**libonsclient4cpp.a**中。

  2. 动态解决方案:

     MQ库文件在**lib/lib-boost-share**目录下，需要业务方在生成可执行文件时链接系统上boost动态库和libonsclient4cpp.so。(需要用户安装boost)

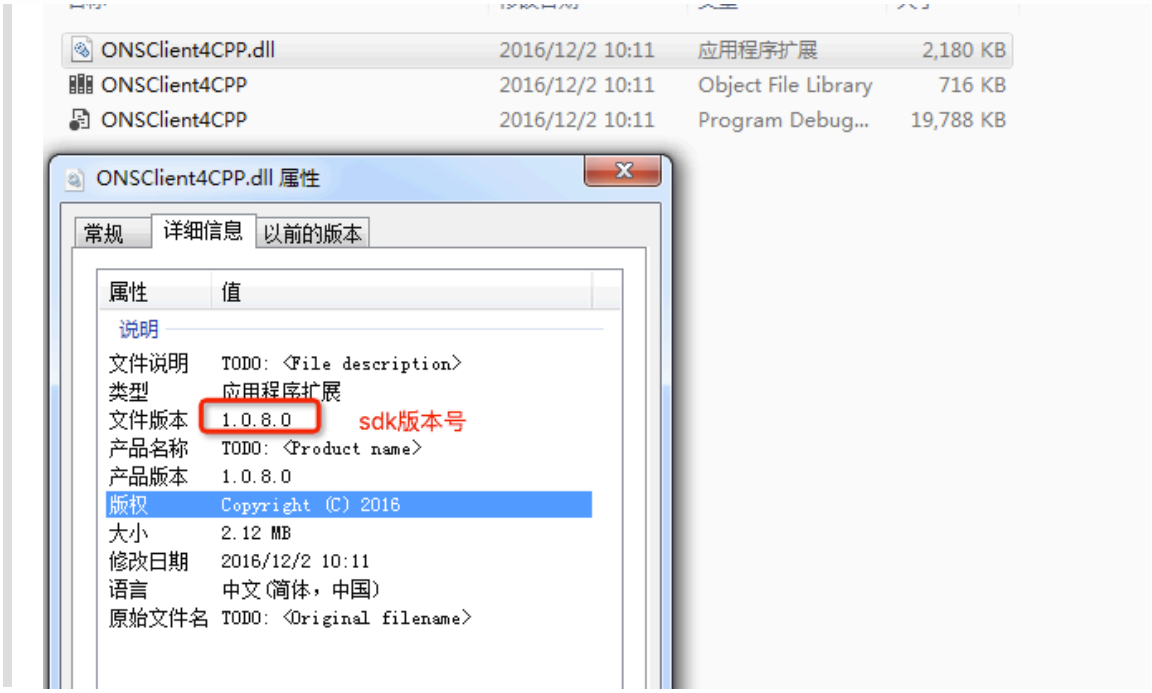
  > 具体的部署方法见下文

- Windows C++版本

  VS2015编译的release 64位版本，如果你是其他VS版本也可以使用, 推荐使用vs2015，共包含下面一些文件:

  1. ONSClient4CPP.lib
  2. ONSClient4CPP.dll
  3. ONSClient4CPP.pdb
  4. vc_redist.x64(Visual c++ 2015的运行时环境，**部署你的机器到其它机器上，切记要安装这个文件**。)

  > 注意: 可以通过右键点击ONSClient4CPP.dll查看sdk的版本号



- Windows .NET版本

  早期的Windows .NET版本是基于CLR对C++进行的托管封装，但是存在的问题较多，基于ASP.NET部署运行的时候不稳定。因此在2016年12月份开始采用基于C++ DLL PINVOKE的方法来调用C++的DLL，基于开源工具SWIG生成。相比于旧版本这种方式部署安装更简单，也更加稳定，并且支持

  1. ONSClient4CPP.lib
  2. ONSClient4CPP.dll
  3. ONSClient4CPP.pdb
  4. vc_redist.x64(Visual c++ 2015的运行时环境)

example目录

包含了普通消息发送，Oneway消息发送、顺序消息发送、普通消息消费、顺序消息消费等例子，Linux下还包含了`Makefile`用于`example`的编译和管理。

changelog

包含了SDK的发布历史，历史发布的每一个版本，包含了版本号、新增的功能、修复的功能、升级的组件、编译环境等等信息。

MQ C++ SDK工程设置

Linux CPP SDK设置

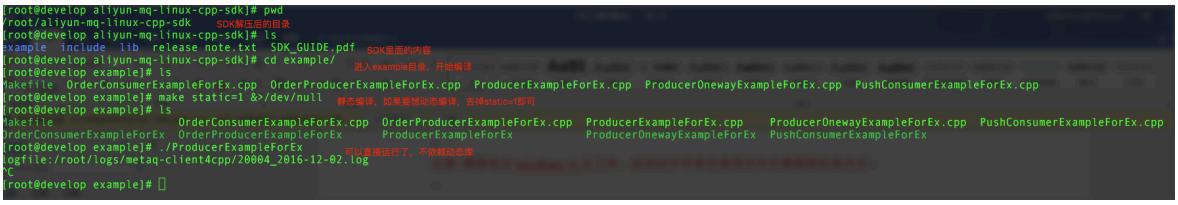
对于没有依赖boost库的业务方，可以直接选用静态库方案，静态库方案中，相应的boost库已经包含在libonsclient4cpp.a中，编译时只需要链接libonsclient4cpp.a即可，无需执行其他操作。

```
g++ -static -I 头文件路径 -L静态库的路径 ProducerExampleForEx.cpp -lonsclient4cpp -lpthread -ldl
```

> 注意: 完全的静态链接请确保机器上安装了libstdc++，pthread等相关的静态库，默认安装的libstdc++是没有安装静态库的，所以需要通过yum或者apt-get来安装相关的静态库。
>
> 使用如上方式会出现一些Warnning如下:
>
> warning: Using 'gethostbyaddr' in statically linked applications requires at runtime the shared libraries from the glibc version used for linking
>
> 建议最佳的方式，不要使用完全的静态链接，而是只静态链接lonsclient4cpp，其他库动态链接即可

```
g++ -Wl,-static -lonsclient4cpp -L$(TOPDIR)/lib/lib-boost-static/ -Wl,-Bdynamic  -lpthread -ldl -lrt  ProducerExampleForEx.cpp
```

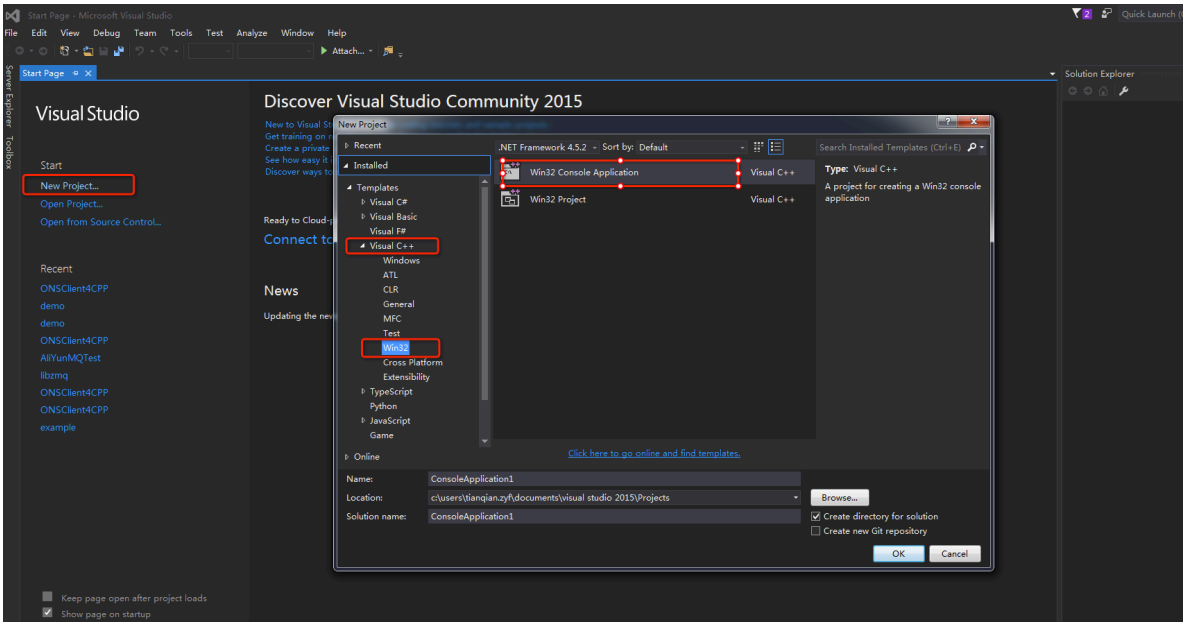
> 注意: 最简单的编译安装方式，下载SDK后，进入example目录，直接执行make即可。

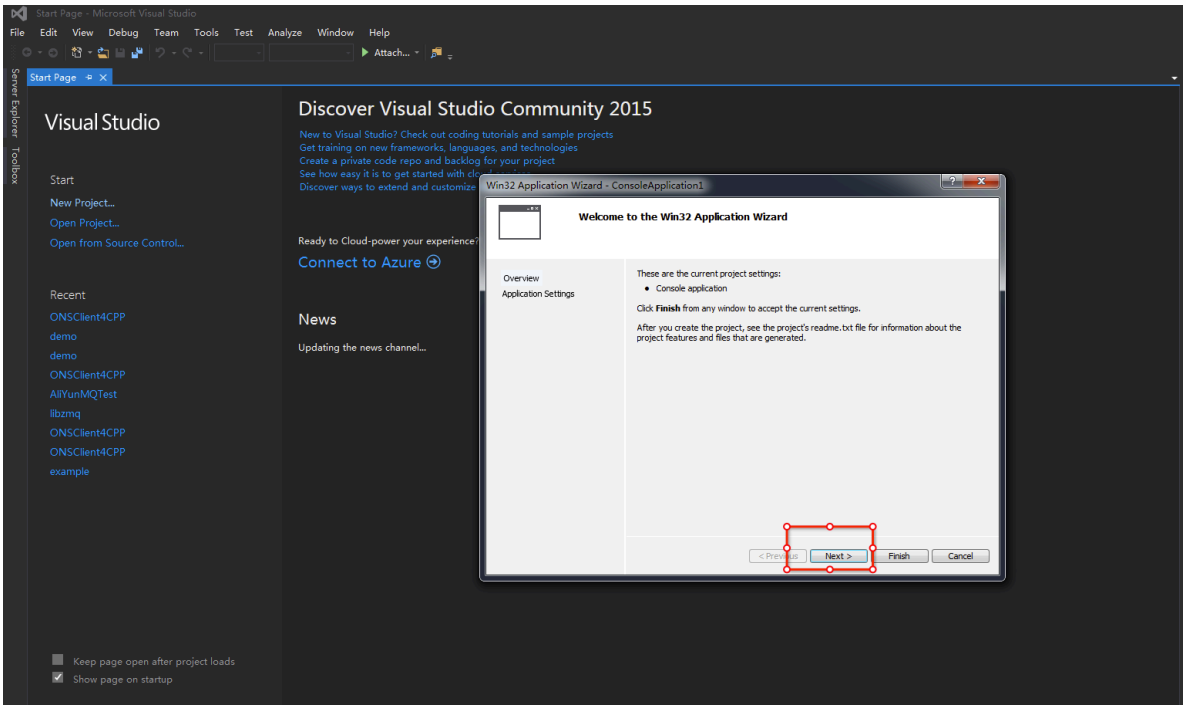


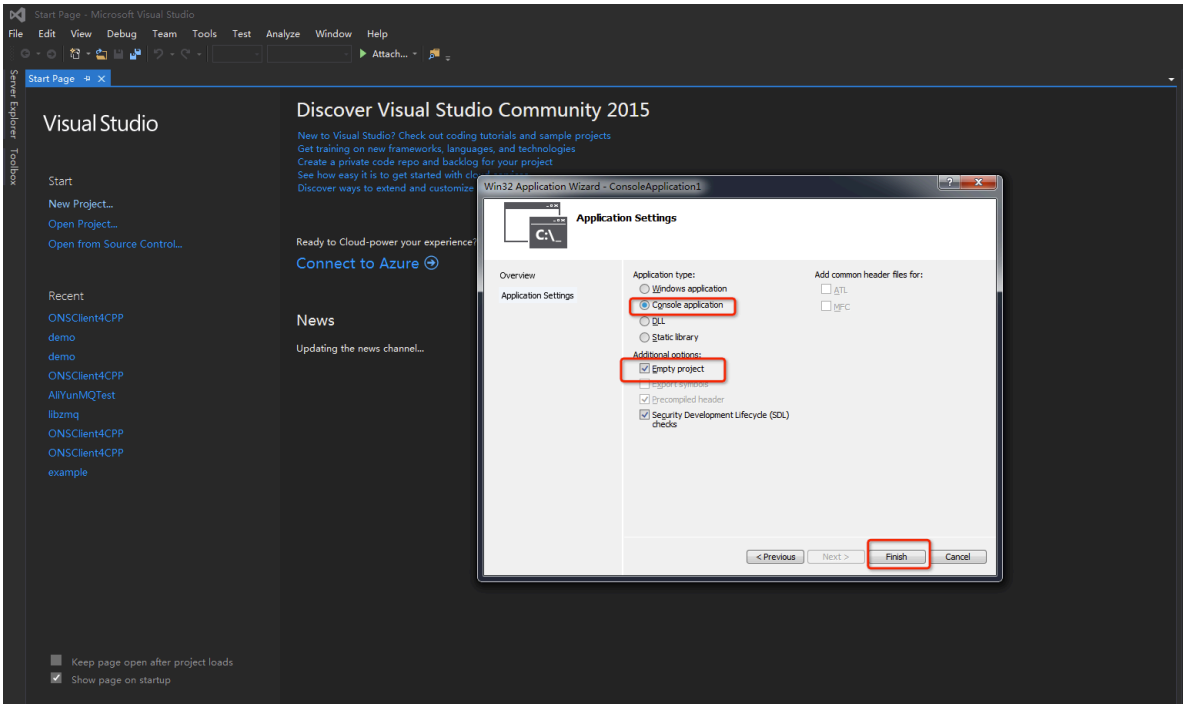
Windows C++ SDK设置

- **Visual Studio 2015 环境下使用 C++ SDK**

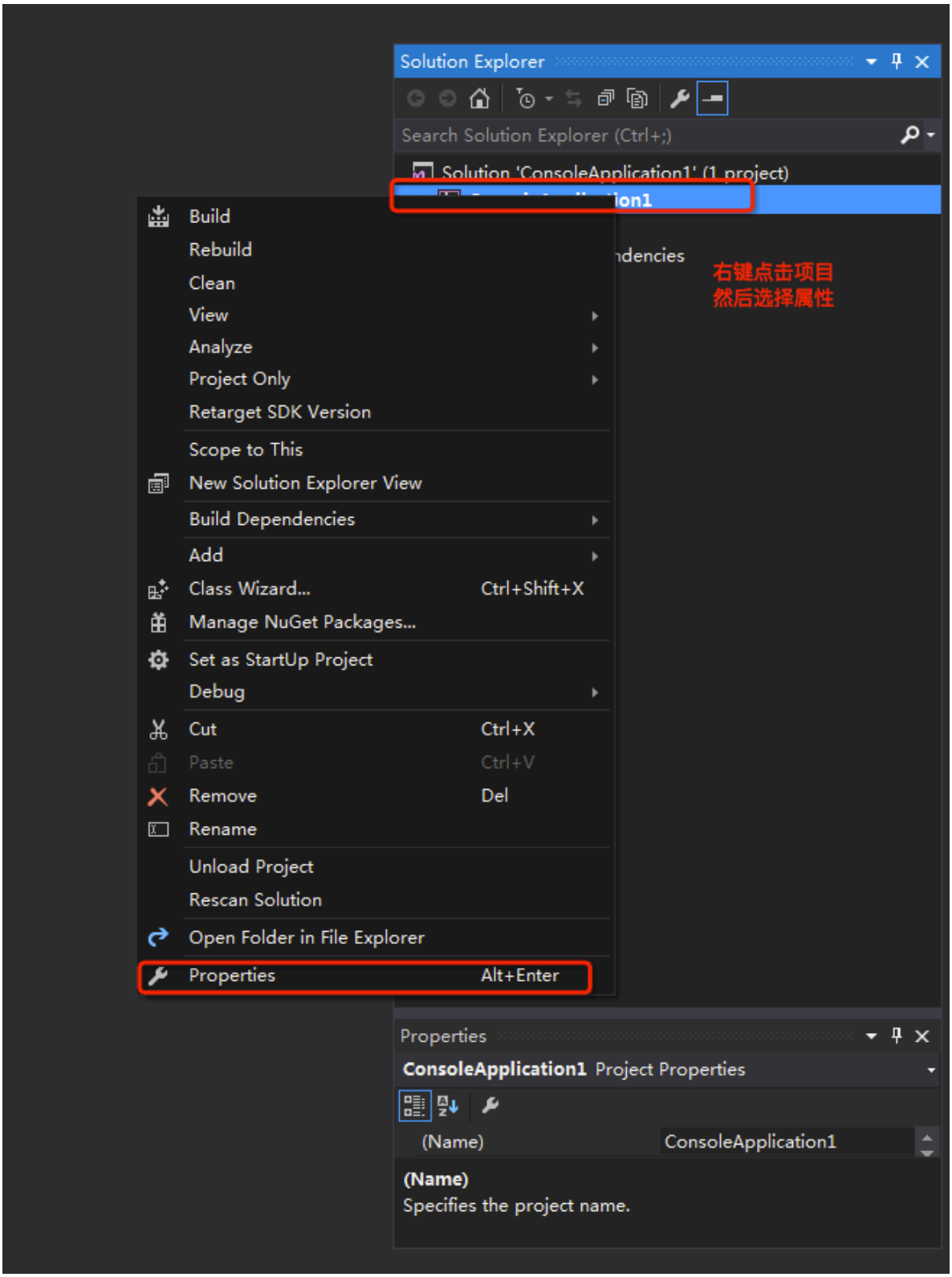
1. 使用 `Visual Studio 2015` 创建自己的项目。

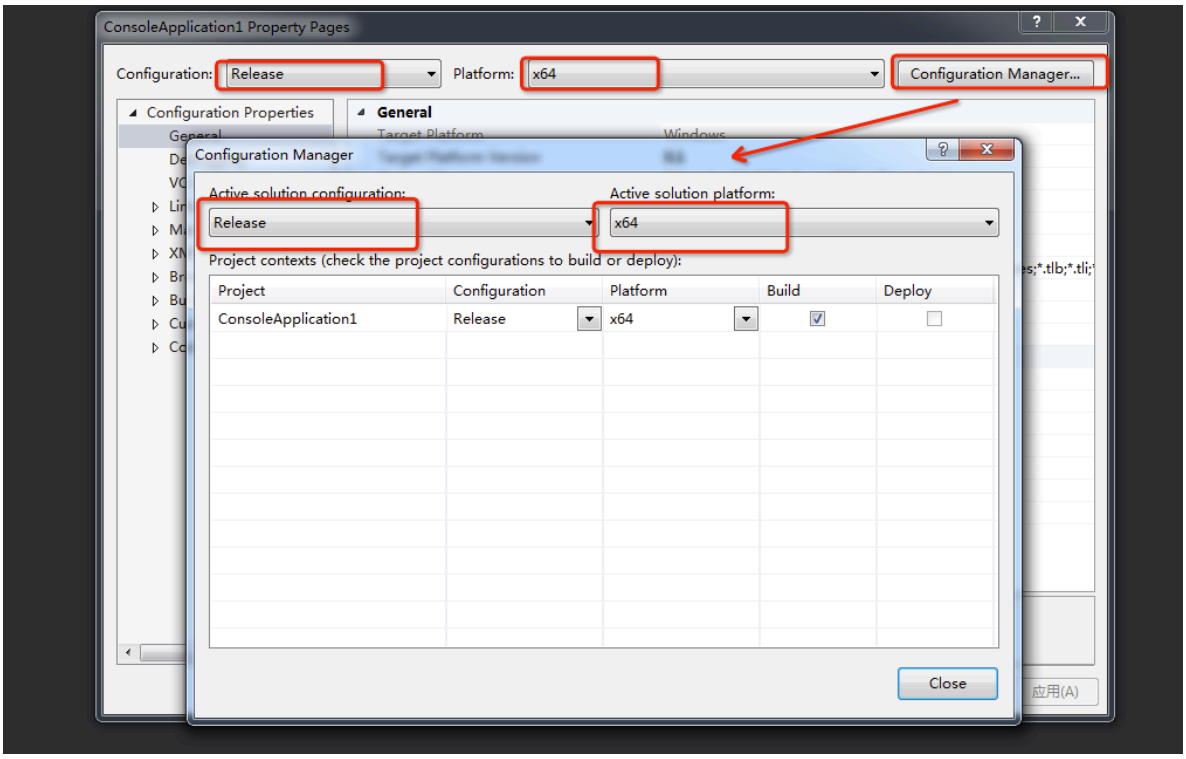




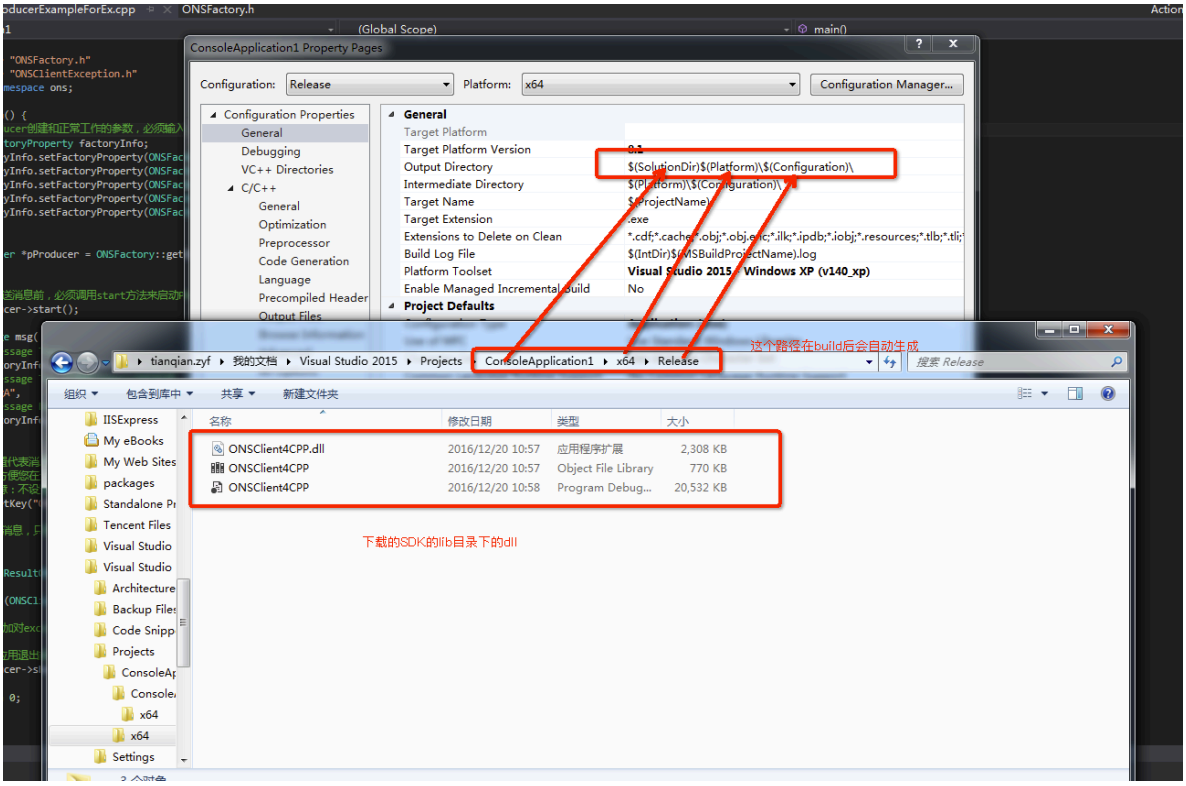


2. 右键单击项目选择**属性>配置管理器** 设置活动解决方案配置为**release**，设置活动解决方案平台为**x64**。

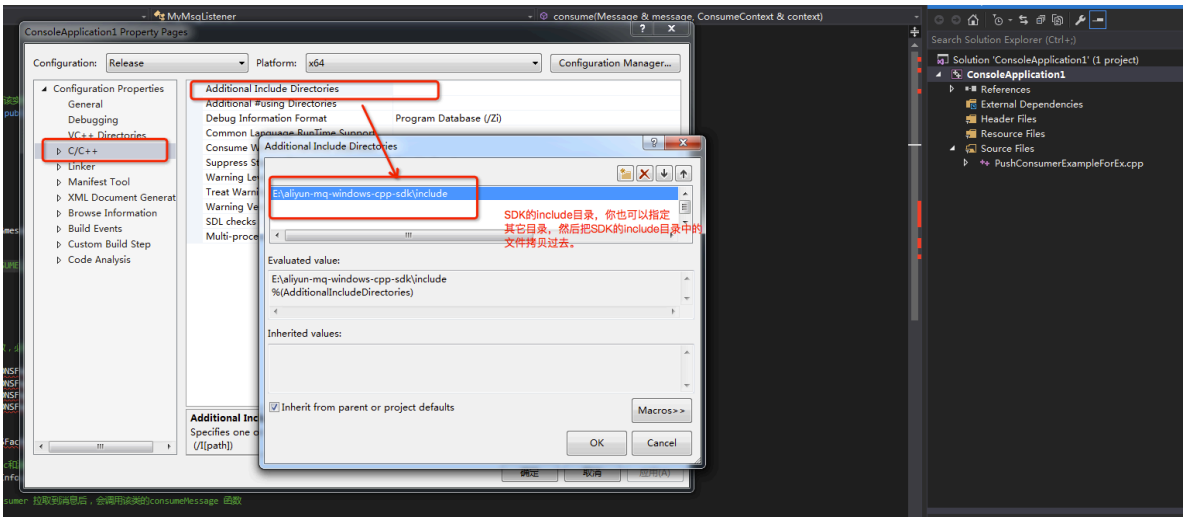

Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'ConsoleApplication1' (1 project)
右键点击项目
然后选择属性
Build
Rebuild
Clean
View
Analyze
Project Only
Retarget SDK Version
Scope to This
New Solution Explorer View
Build Dependencies
Add
Class Wizard... Ctrl+Shift+X
Manage NuGet Packages...
Set as StartUp Project
Debug
Cut Ctrl+X
Paste Ctrl+V
Remove Del
Rename
Unload Project
Rescan Solution
Open Folder in File Explorer
Properties Alt+Enter
Properties
ConsoleApplication1 Project Properties
(Name) ConsoleApplication1
(Name)
Specifies the project name.

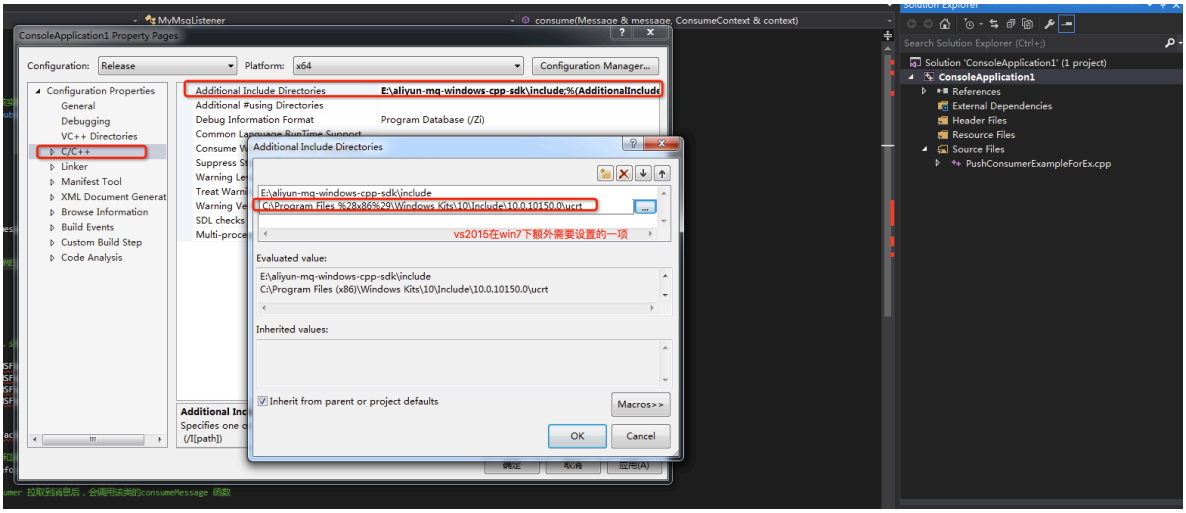


3. 右键单击项目选择**属性>配置属性>常规>输出目录**：**/A**。按照**活动解决方案平台**的设置，拷贝 64 位 lib 目录下的所有文件到输出目录 **/A**。

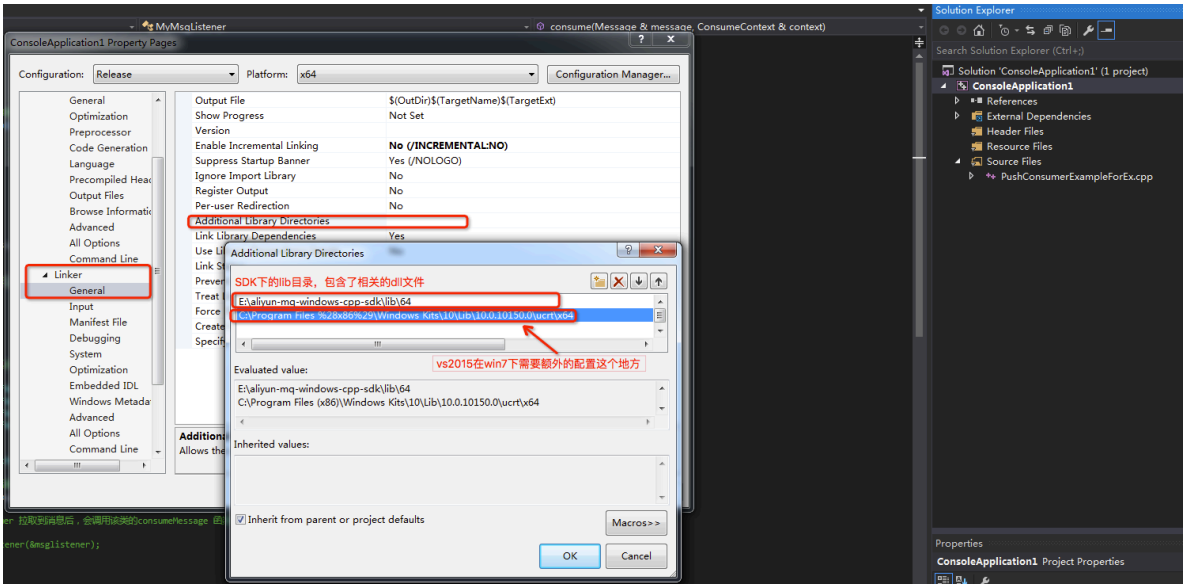


4. 右键单击项目选择**属性>配置属性>C/C++-常规>附加包含目录**：**/B**。拷贝 include 目录下的头文件到包含目录: **/B**。

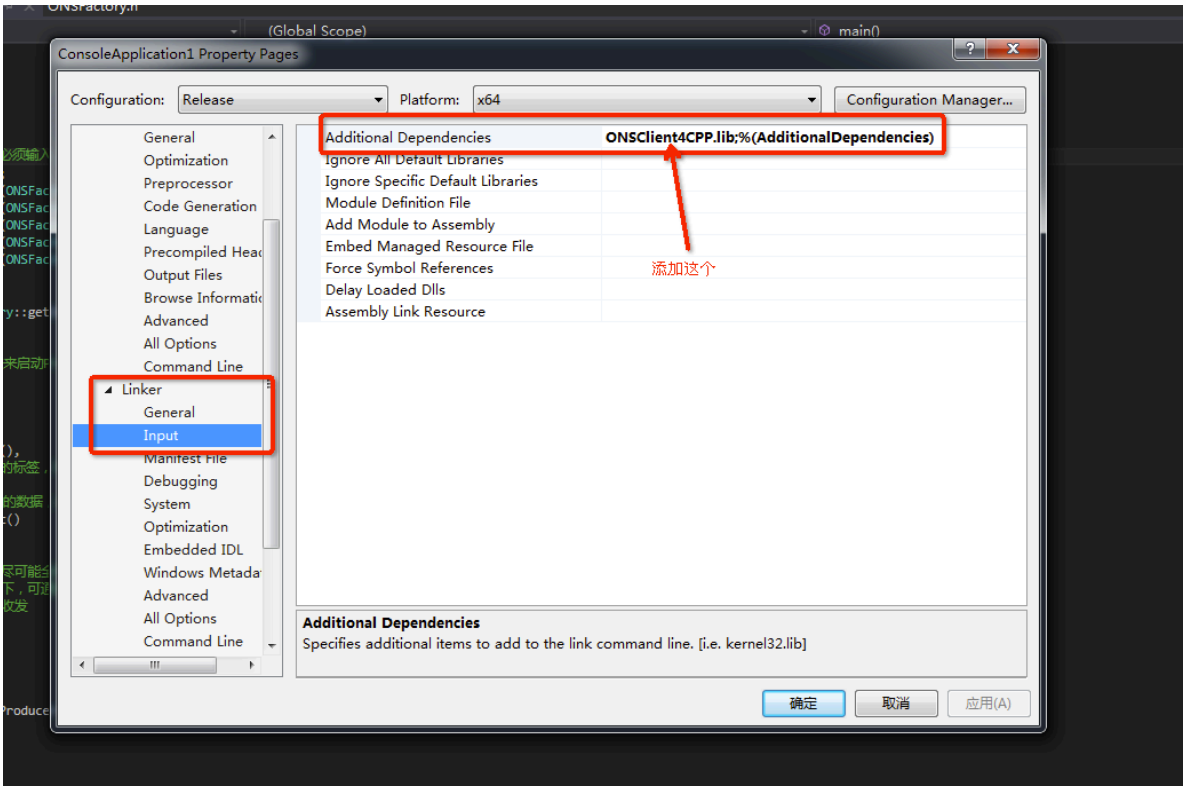




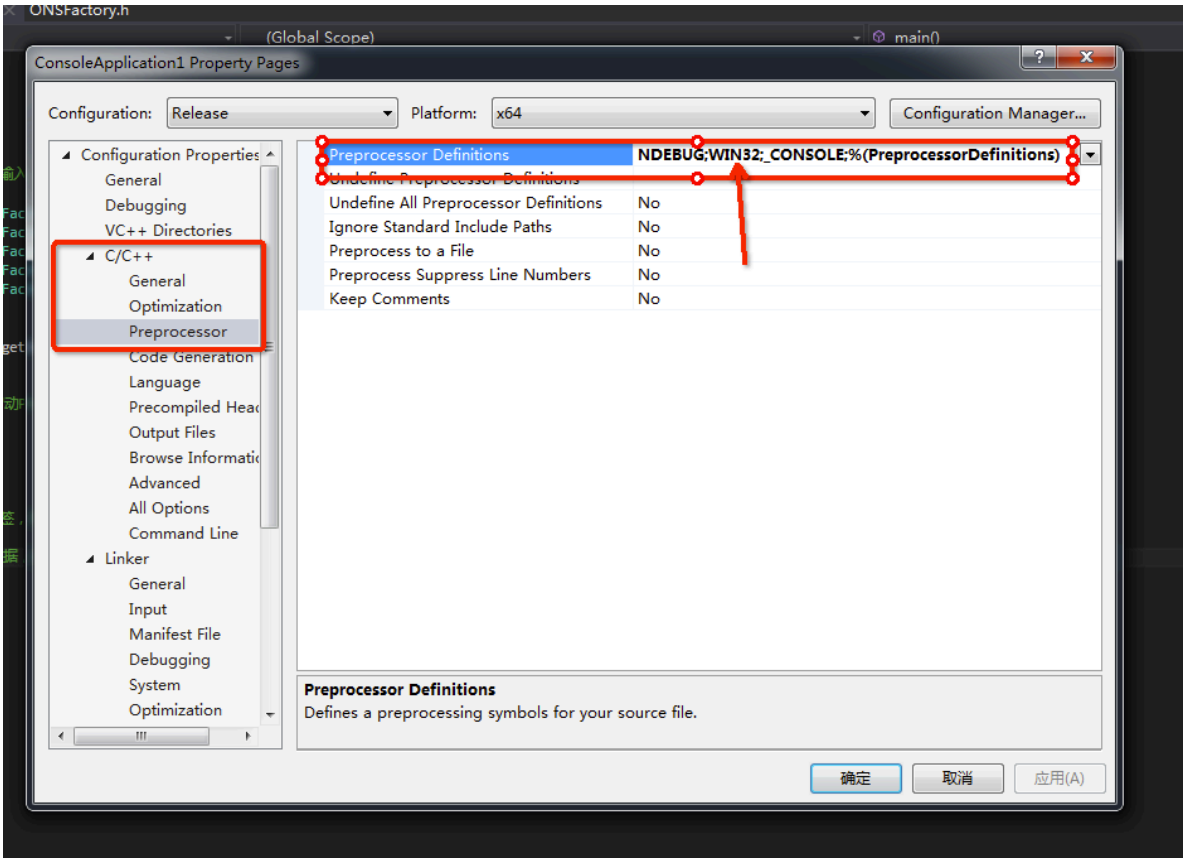
5. 右键单击项目选择**属性>配置属性>链接>常规>附加库目录：/A。**



6. 右键单击项目选择**属性>配置属性>链接>输入>附加依赖项：ONSClient4CPP.lib。**



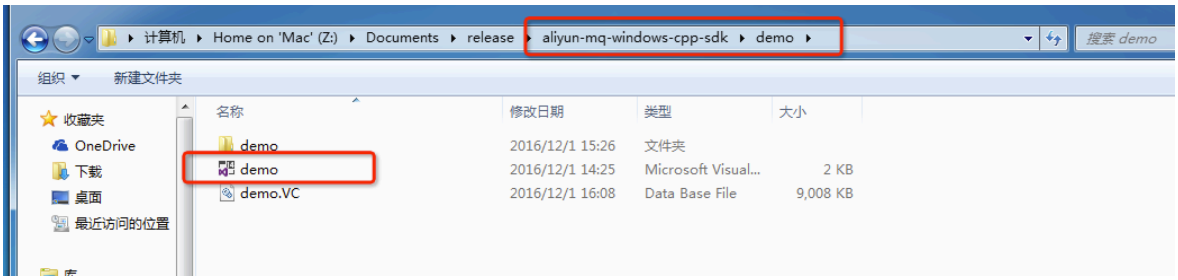
7. 右键单击项目选择**属性>配置属性>>C/C++-常规>>预处理器定义**: 添加**WIN32**宏

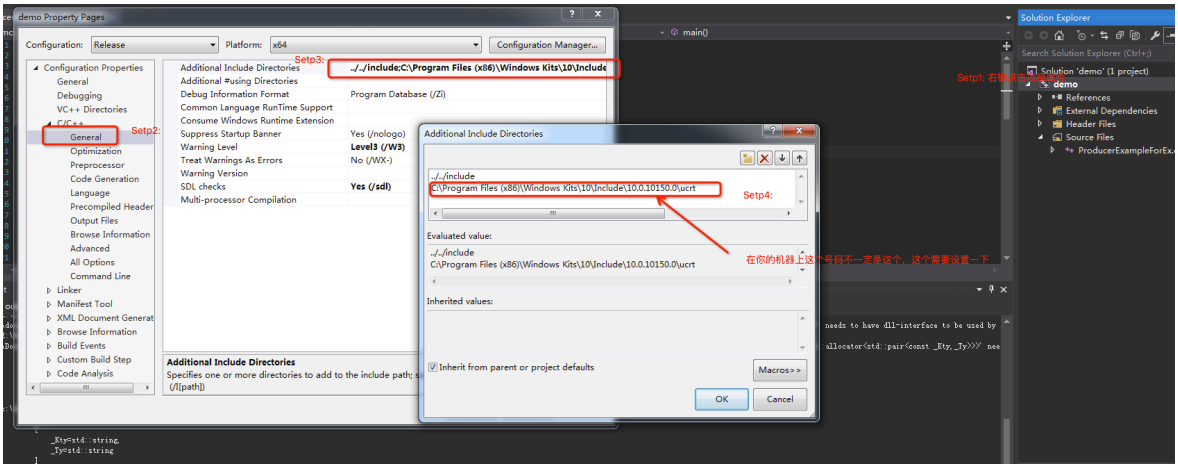


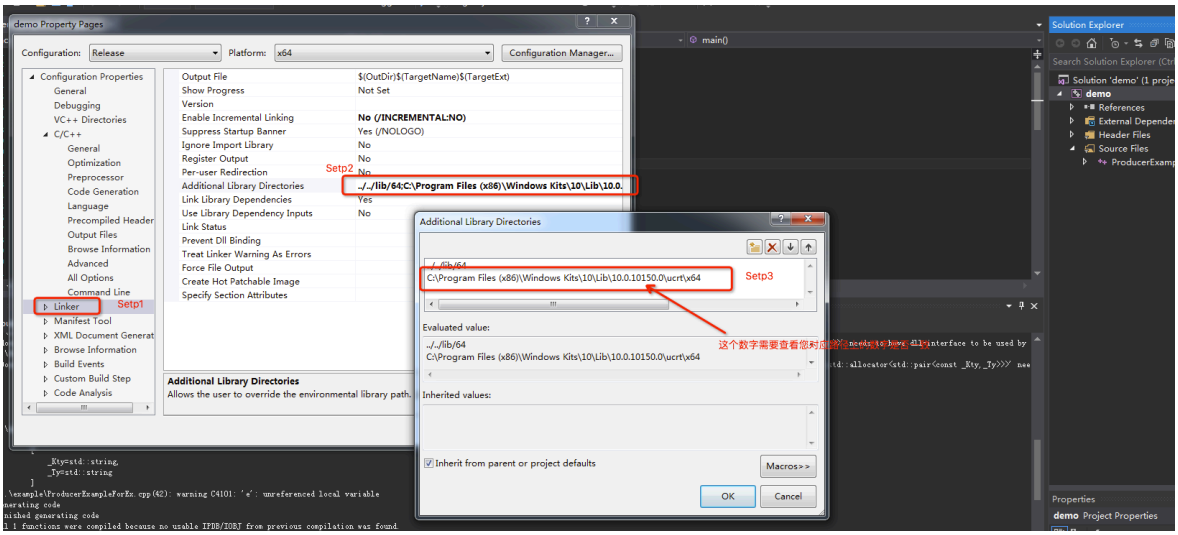
- 非 Visual Studio 2013环境下使用 C++ SDK**

1. 首先需要按照 Visusal Studio 2015 的环境来配置，配置过程同上。
2. 安装`vc_redist.x64`这是`Visual C++ 2015`的运行时环境。

> 注意: 不想进行繁琐的设置话，你可以使用设置好的SDK demo，下载SDK后 进行解压，然后进入demo目录使用vs2015打开工程。





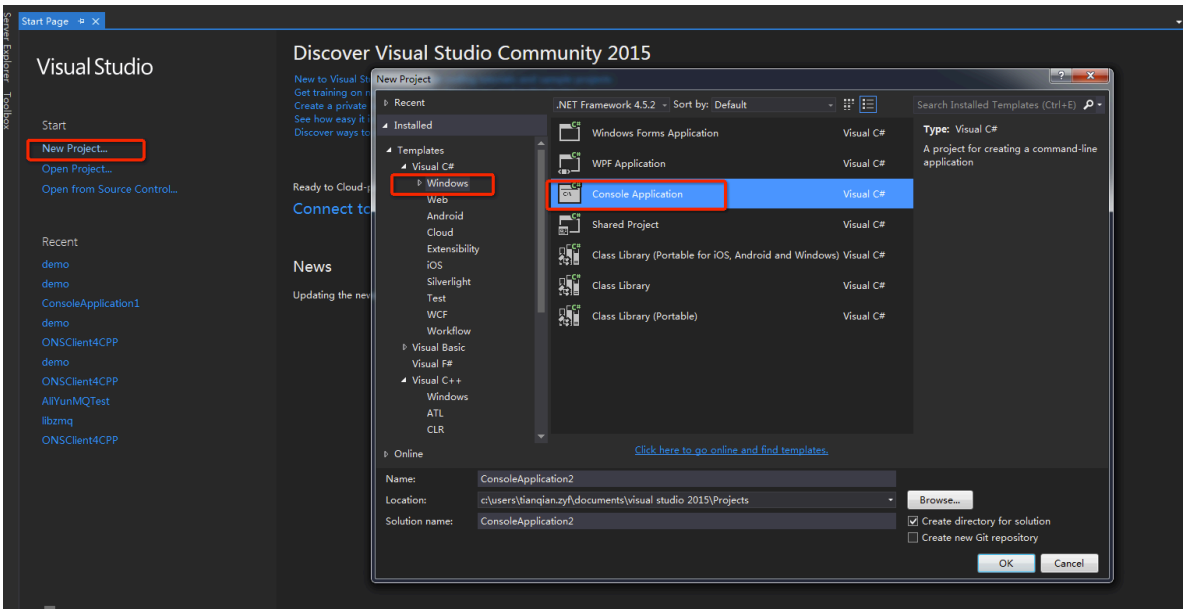


到此为此就设置好编译环境了，点击Build，即可编译出可执行的程序，然后拷贝dll到可执行程序的目录下即可运行，或者拷贝dll到系统目录下。

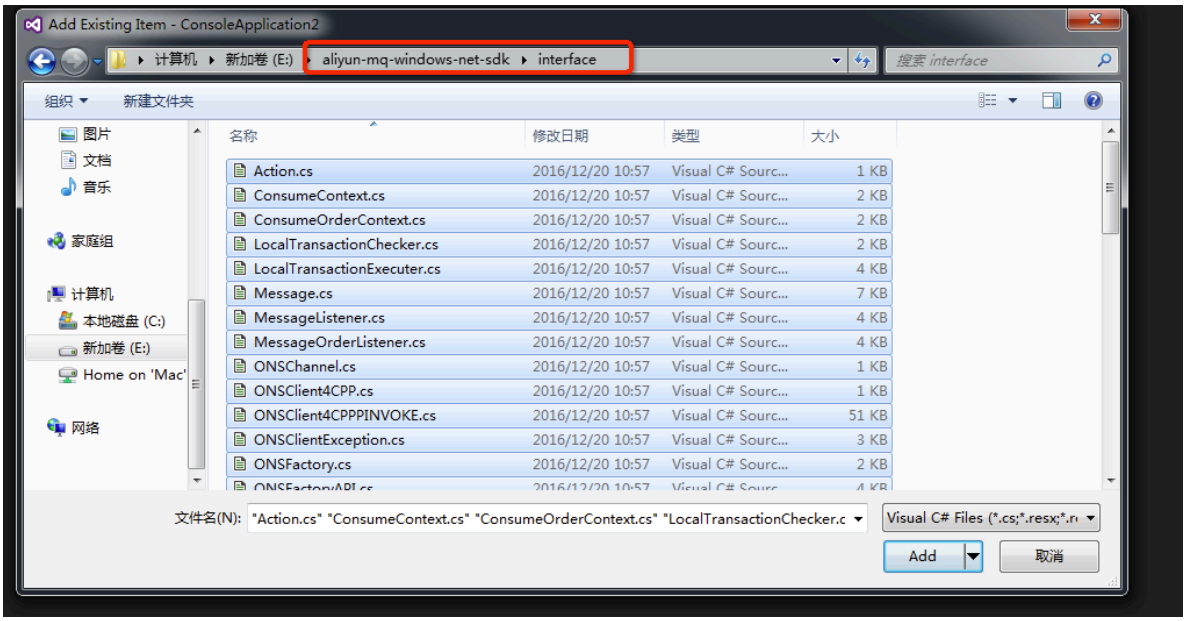
MQ .NETSDK设置

- **Visual Studio 2015 使用 .NET SDK 配置说明**

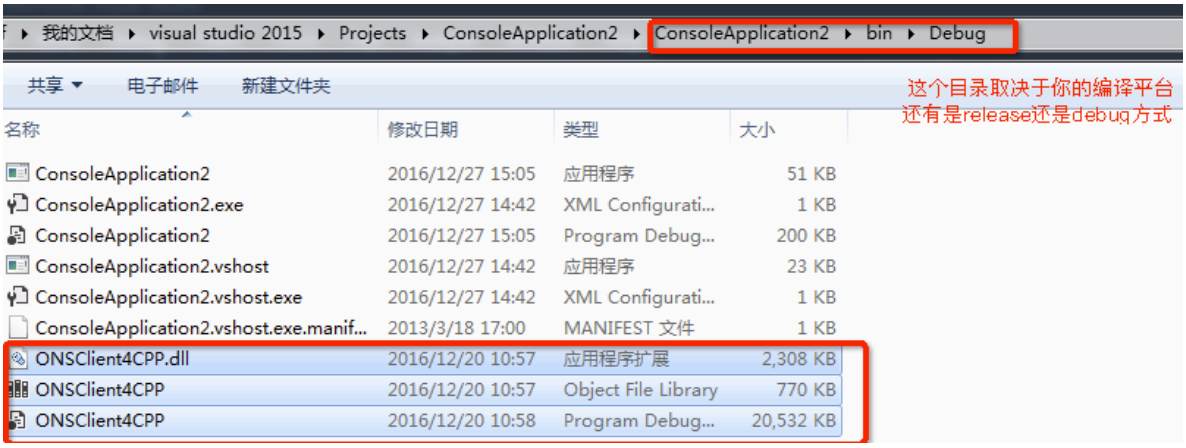
1. 使用 `Visual Studio 2015` 创建自己的项目。



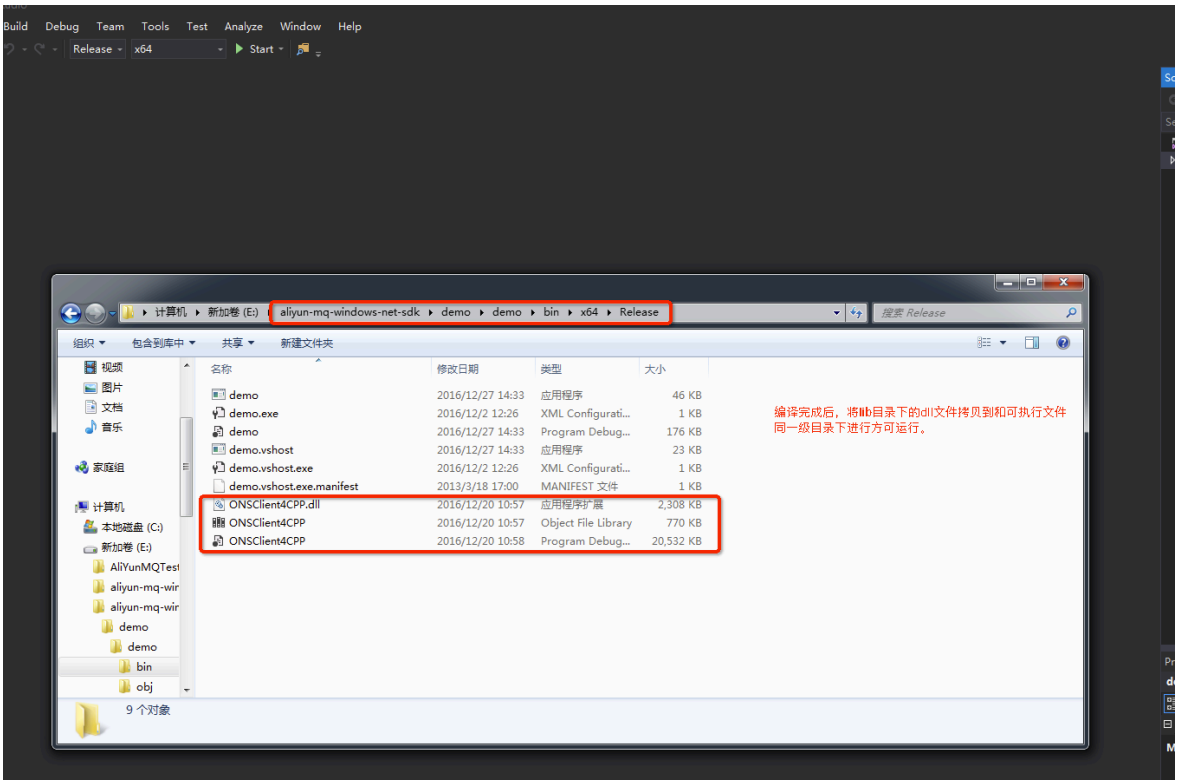
2. 右键单击项目选择**添加>现存在项** 将下载的SDK中的interface目录下的所有文件都添加进去。



3. 编写测试程序，然后进行编译，最后将SDK下的dll放到和可执行文件同一级目录下，或者系统目录下即可运行

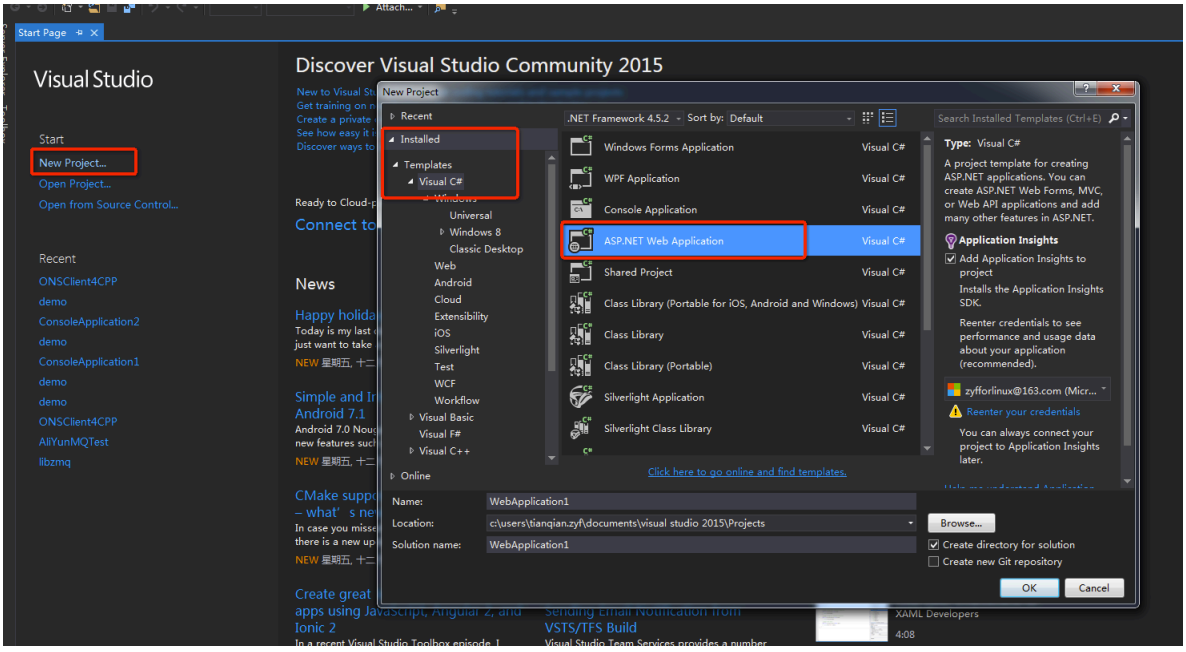


> 注意：提供了设置好的demo，操作如下

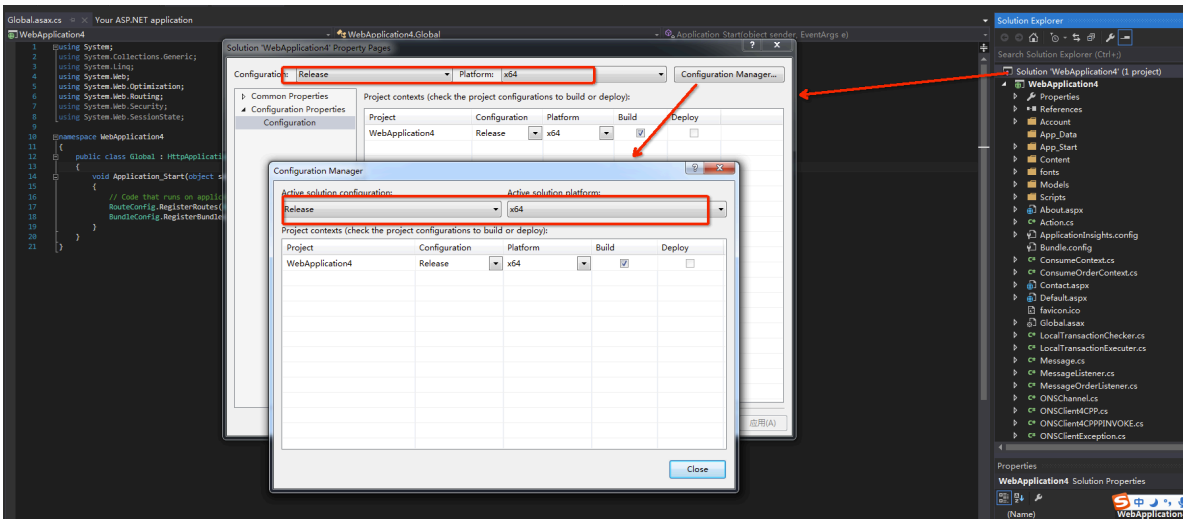


- **Visual Studio 2015配置ASP.NET使用MQ SDK**

1. 使用vs2015创建一个ASP.NET的**WEB Forms**项目



2. 右键单击项目选择**属性**。选择**配置管理器**，设置**活动解决方案配置**为 **release**；设置**活动解决方案平台**为 **x64**。



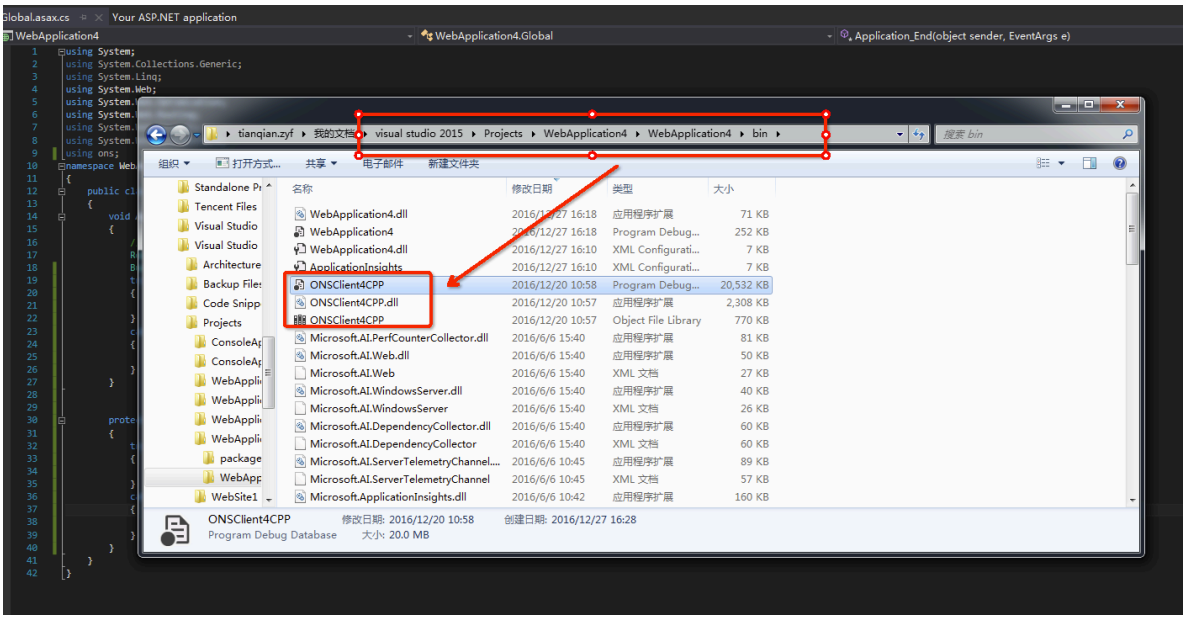
3. 右键单击项目选择**添加>现存在项** 将下载的SDK中的interface目录下的所有文件都添加进去。

> 参考上文中配置普通的.net项目的步骤2

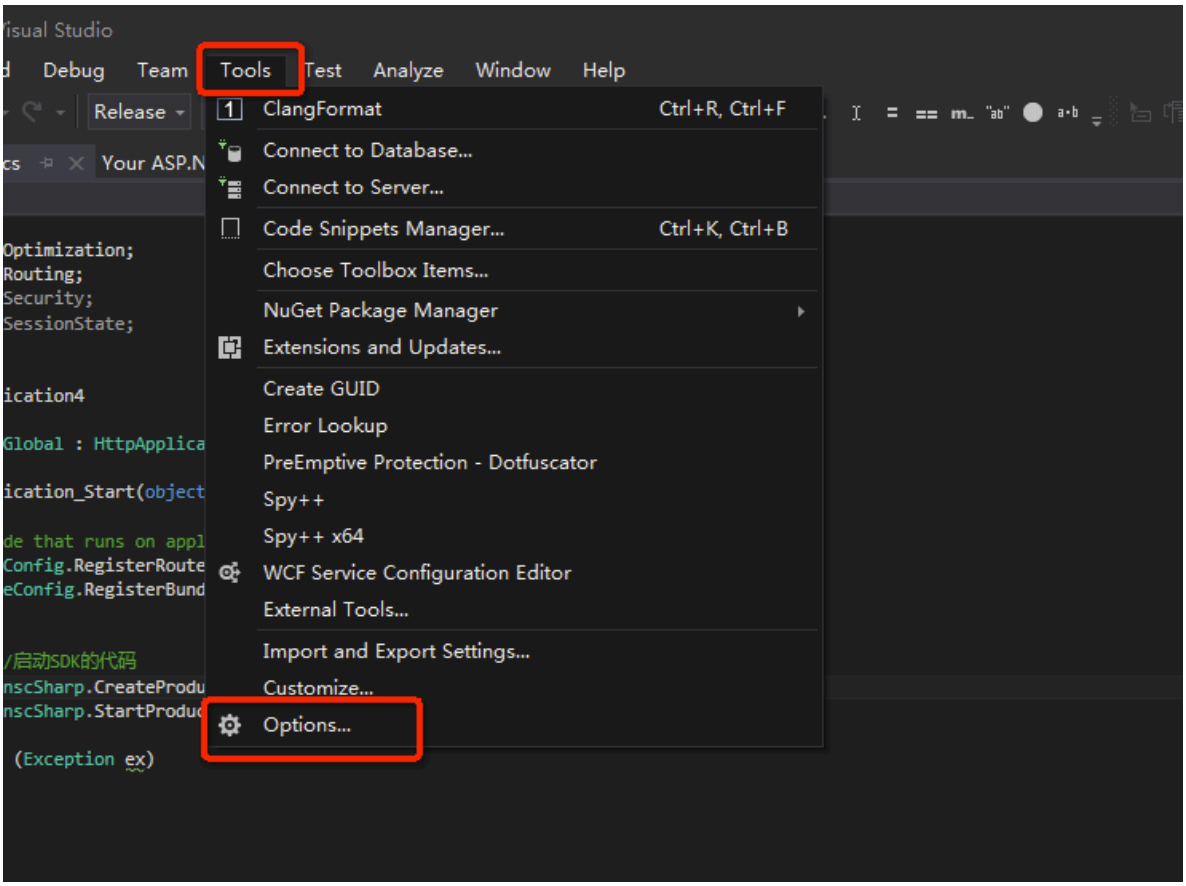
4. 在**Global.asax.cs**文件中添加启动和关闭SDK的代码，应该试图将SDK的代码封装成一个单例类，这样避免因为作用域的问题被垃圾回收器回收掉。

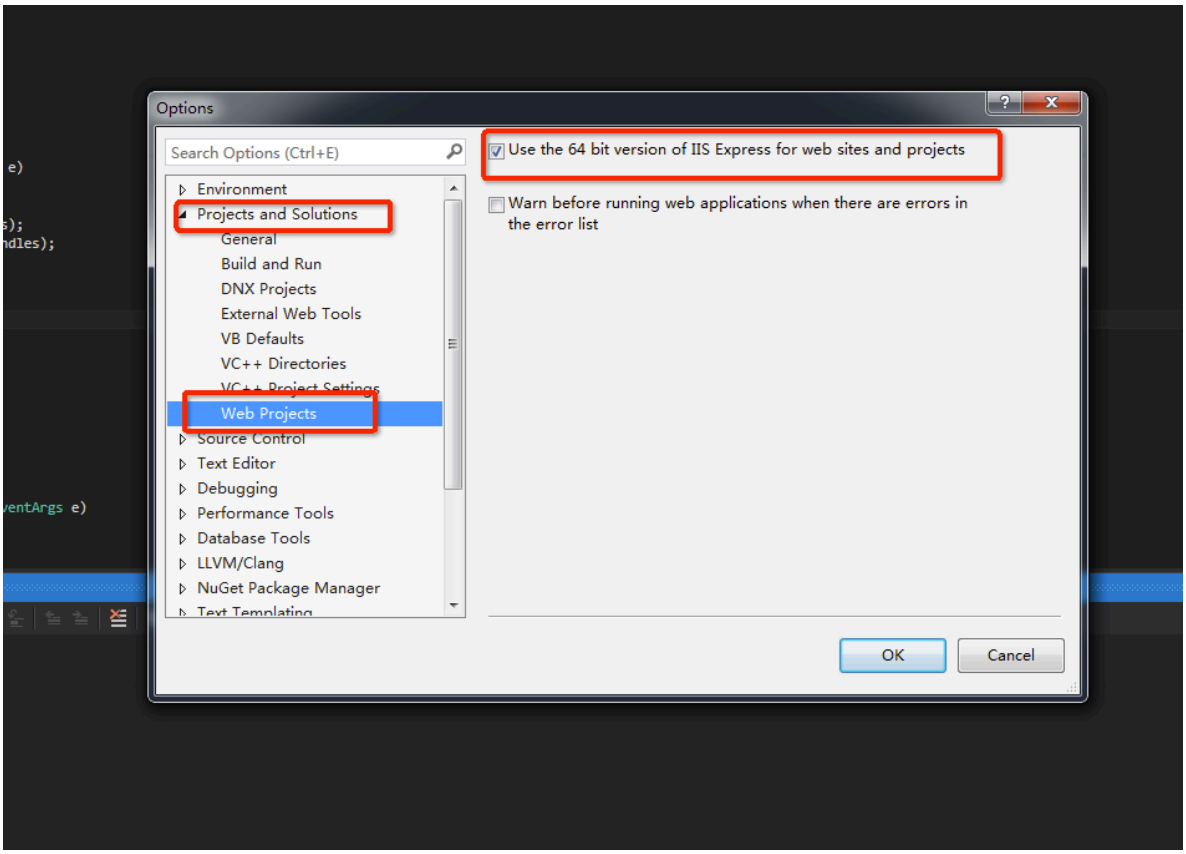
```
sing System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Optimization;
using System.Web.Routing;
using System.Web.Security;
using System.Web.SessionState;
using ons;  // 这个命名空间是SDK所在的命名空间
using test; // 这是一个简单封装sdk的类所在命名空间，见SDK的example目录下的
Example.cs
namespace WebApplication4
{
    public class Global : HttpApplication
    {
        void Application_Start(object sender, EventArgs e)
        {
            // Code that runs on application startup
            RouteConfig.RegisterRoutes(RouteTable.Routes);
            BundleConfig.RegisterBundles(BundleTable.Bundles);
            try
            {
                //启动SDK的代码,下面是简单封装sdk后的代码
                OnscSharp.CreateProducer();
                OnscSharp.StartProducer();
            }
            catch (Exception ex)
            {
            }
        }
        protected void Application_End(object sender, EventArgs e)
        {
            try
            {
               // 关闭SDK的代码
              OnscSharp.ShutdownProducer();
            }
            catch (Exception ex)
            {
            }
        }
    }
}
```

5. 编写测试程序，然后进行编译，最后将SDK下的dll放到和可执行文件同一级目录下，或者系统目录下即可运行



6. 点击**工具>选项>项目和解决方案>Web项目**，然后勾选对网站和项目使用**IIS Express**的64位版





> 关于.net sdk的使用示例查看sdk的example目录下的Example.cs，这是一个简单封装了MQ SDK的例子，SDK的demo目录相应的耶提供了ASP.NET的demo，可以直接打开运行。

FAQ

- .net程序在shutdown后程序无法退出?

  使用**Process.GetCurrentProcess().Kill()**进行退出，因为c#的程序在当前还有线程没有退出的时候是不会结束程序的，我们的SDK内部启了很多线程，但是shutdown的时候，并没有优雅的关闭所有的线程，因此程序没有退出，使用上面的代码可以让程序退出，后续会进行改进。

- 环境配置出问题，无法配置成功?

  在配置运行环境的时候，请一定要先确认提供的demo是否可以正常工作，如果连demo都无法正常工作的话，就更别提自己的配置环境了。原因可能有以下几个。

  1. 是否认真读了文档，demo执行的时候需要拷贝相关的dll。
  2. asp.net的IIS调试器是否配置成64位。
  3. linux C++场景下执行make是动态编译，用户的机器上需要安装boost库，如果要静态编译直接执行make static=1，除此之外请确认下gcc是否是4.8.x系列，5.x系列的ABI和4.x系列是不兼容的。还有boost的版本是否大于等于1.6.2，因为boost向下兼容的。
  4. 更新SDK的时候是否除了dll外其它的没有更新，比如SDK中的include目录下的头文件，.net的interface目录下的文件，因为每次发布新版本的时候可能会轻微改动这些文件，所以这些都需要全部更新。
  5. windows C++ demo无法运行成功，是否详细看了文档，vs2015在win7需要进行一些额外的配置，ucrt库的路径问题，这个需要单独设置一下，因为不同的win7系统和vs版本这个路径不是一定相同的，这个问题可以很简单的通过google来解决。这个路径不配置的话你是无法使用vs2015创建一个c++的项目运行起来的。
  6. 是否安装了vc2015的运行时环境，也就是下载的SDK包里面的vc_redist.x64。

- 如何定位SDK在ASP.NET无法正常运行的问题?

  1. 理解**Global.asax.cs** 中的**Application_Start**和**Application_End**的调用时机
  2. SDK是作为一个全局的服务来用，启动会需要初始化，仅能初始化一次，结束需要关闭只关闭一次，而网页是无状态的，每个页面有自己的生命周期，因此为了使用SDK，最简单的方法是封装SDK成为一个单例，这样就可以全局的去使用
  3. vs里面的iis调试器默认是32位的，但是我们的dll是64位的，所以会导致无法解析dll，需要设置iis调试器为64位
  4. 调试的时候要先确认SDK的日志生成了吗？只有日志生成了才能说明SDK初始化了。
  5. 确认SDK启动了，但是消息发不出去，原因有很多，归为三大类，第一类就是参数配置错误，比如key，topic之类的配置错误，第二类就是网络问题，第三类就是sdk被shutdown了(这类问题在频繁的使用vs的iis调试器调试的时候会出现)可以通过查看日志是否有shutdown等关键字来确认sdk是否被关闭了
  6. 消费消息的时候，需要设置监听类，这个类会被dll中的C++代码回调，C++不负责管理这个类的生命周期，C#有自己的gc管理，为了避免C#的gc机制将监听类意外析构，应该尽可能用C#的方式延长监听类的生命周期，比如可以设置成全局的，或者单例，或者static等等。

- 日志太大，日志路径的设置问题?

  新版本的SDK可以进行日志路径的设置了，设置方式如下。

  ```
  C#
  ONSFactoryProperty factoryInfo = new ONSFactoryProperty();
  factoryInfo.setFactoryProperty(ONSFactoryProperty.LogPath, "E://log");

  C++
  ONSFactoryProperty factoryInfo;
  factoryInfo.setFactoryProperty(ONSFactoryProperty::LogPath, "/tmp/log");

  注意：windows路径和linux路径的不同，启动的时候会打印日志文件，如果日志路径不对会在启动程序的时候输出错误信息。
  ```

  很多人反感启动程序的时候输出的日志路径名这样控制台打印信息，这个下个版本去掉。

- 新版本的.net SDK和老版本的SDK的区别是是什么?

  老版本的SDK分成三层，第一层是用户的C#代码，第二层是基于C++和.NET混合编写的中间接口层，会生成ManagedONS.dll，其具体实现依赖底层的C++ DLL 第三层就是底层的C++ DLL，也就是ONSClient4CPP.dll。这个版本的SDK维护工作量大，容易出错，不稳定，不支持ASP.NET。新版的SDK的基于C# PINVOKE直接来调用底层的C++ DLL，通过开源软件SWIG生成了封装PINVOKE的C#代码，用户只需要包含这些C# 代码就可以使用SDK了。分成两层，第一层是用户的C#代码和封装PINVOKE调用的C#代码，第二层是C++ DLL。新版本的SDK支持ASP.NET，部署配置很方便。因为给你的是一堆C#文件，你直接包含就可以进行开发了，只是在运行的时候需要把dll和可执行文件放到同级目录即可。