

本文主要介绍在已开通ONS服务的基础上，如何使用ONSClnt4cpp 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：[ALIYUN_MQ_USER_GUIDE.pdf](#)。

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：[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三个库；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boost动态库和libonsclient4cpp.so。

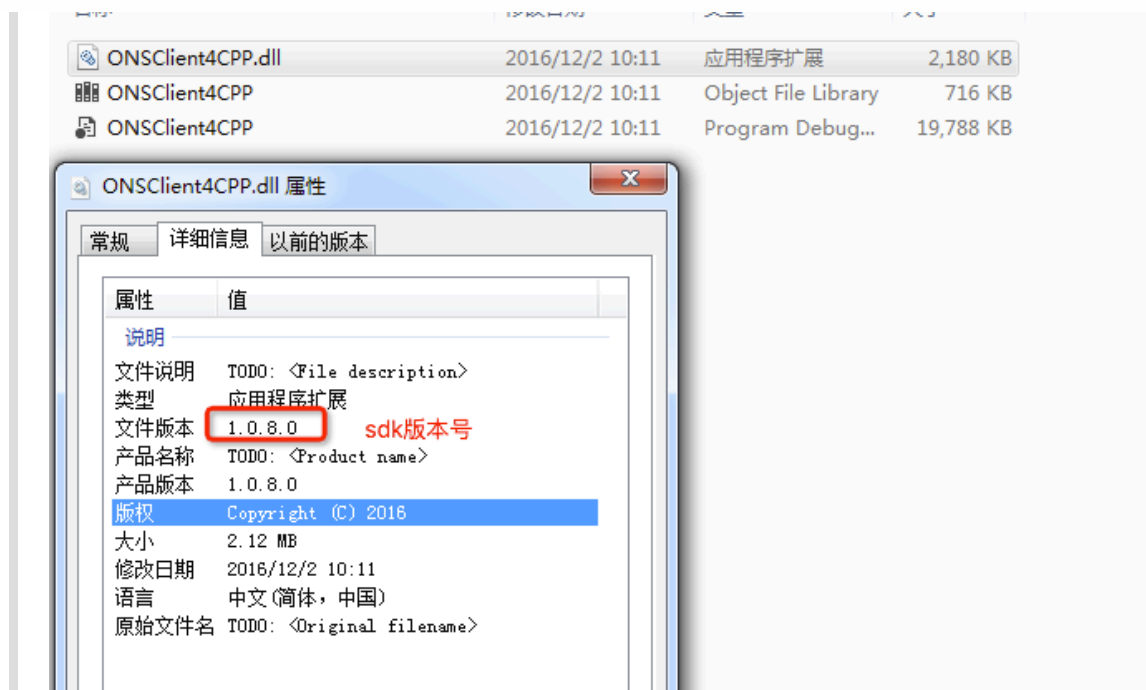
具体的部署方法见下文

- 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的运行环境)

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



- Windows .NET版本

我们提供的.NET版本是基于MQ CPP版本的托管封装, 这样能保证.NET完全不依赖于 windows .NET公共库, 内部才用C++多线程并发处理, 保证.NET版本的高效稳定; 在使用VS开发.NET的应用程序和类库时, 默认的目标平台为Any CPU, 即会在运行时可根据CPU类型自动选择X86或X64, 拥有这样的能力是因为.NET编译后的程序集是基于IL的, 在运行时, CLR才会将其JIT发射为X86或X64的机器码。而C或C++编译生成的DLL就是机器码, 所以, 其平台的决策是在编译时决定的。通过编译选项的设置, 我们将C/C++项目编译为X86的32位dll或者X64的64位dll, 因此我们提供了包含VS2015编译的release 64位版本DLL, 其他VS版本也可以使用, 共包含:

1. ONSClient4CPP.lib
2. ONSClient4CPP.dll
3. ONSClient4CPP.pdb
4. ManagedONS.dll (在引用dll的时候, 只要引用这一个就可以了, 它依赖 ONSClient4CPP.dll)
5. ManagedONS.pdb
6. 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。

release note

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 C++ 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。

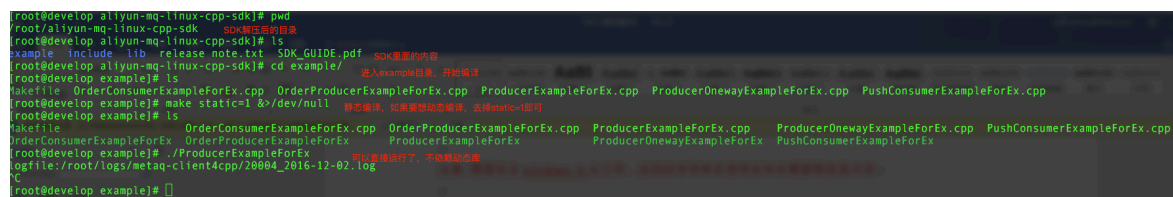
使用如上方式会出现一些Warning如下:

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

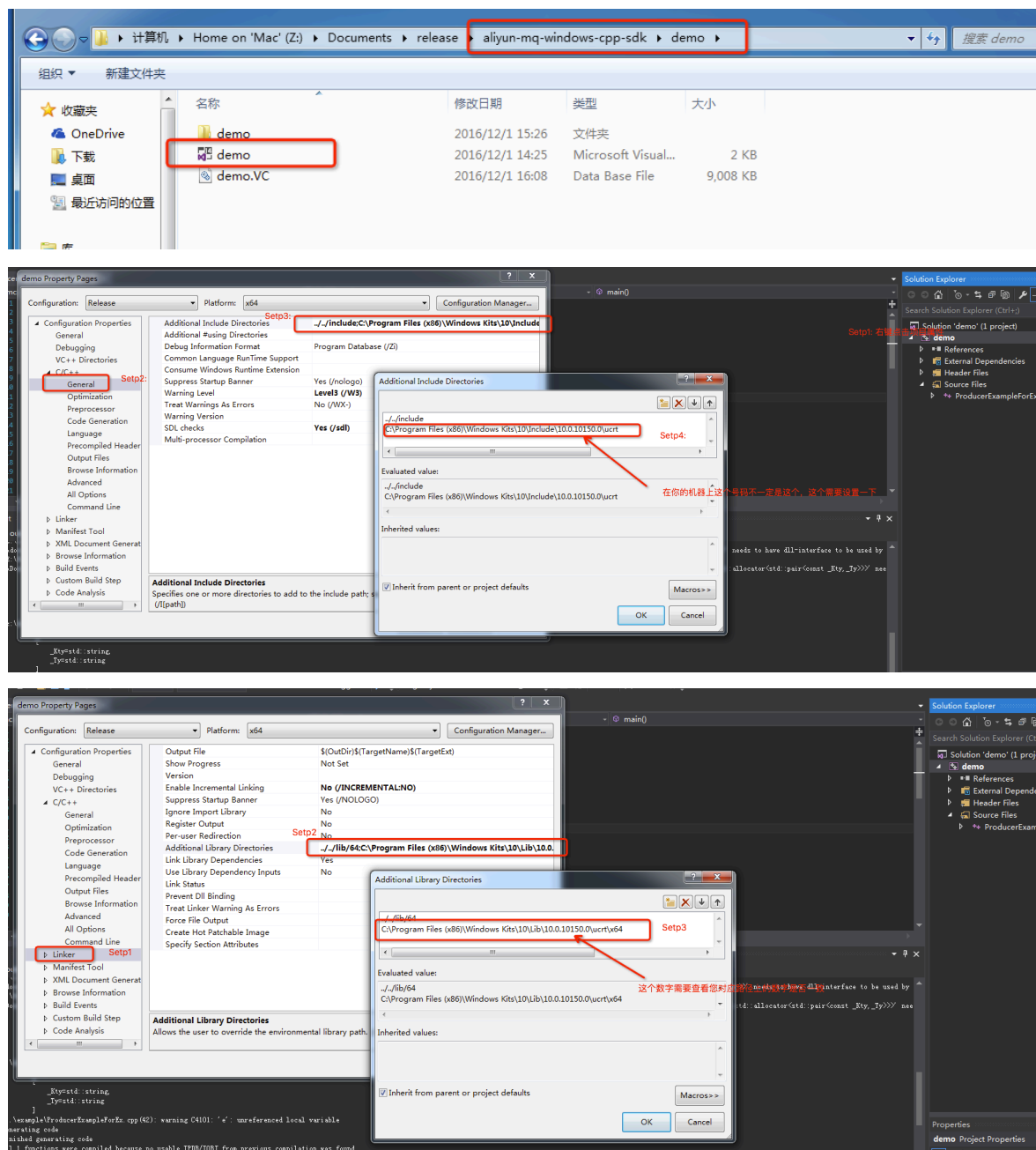
● Visual Studio 2015 环境下使用 C++ SDK

1. 使用 Visual Studio 2015 创建自己的项目。
2. 右键单击项目选择属性。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**。

● 非 Visual Studio 2013环境下使用 C++ SDK

1. 首先需要按照 Visual Studio 2015 的环境来配置，配置过程同上。
2. 右键单击项目选择属性>配置属性>C/C++-代码生成>运行时检查，选择默认。
3. 右键单击项目选择属性>配置属性>清单工具>输入和输出>嵌入清单，选择否。
4. 安装 vc_redist.x64 这是 visual C++ 2015 的运行环境。

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。



到此为止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

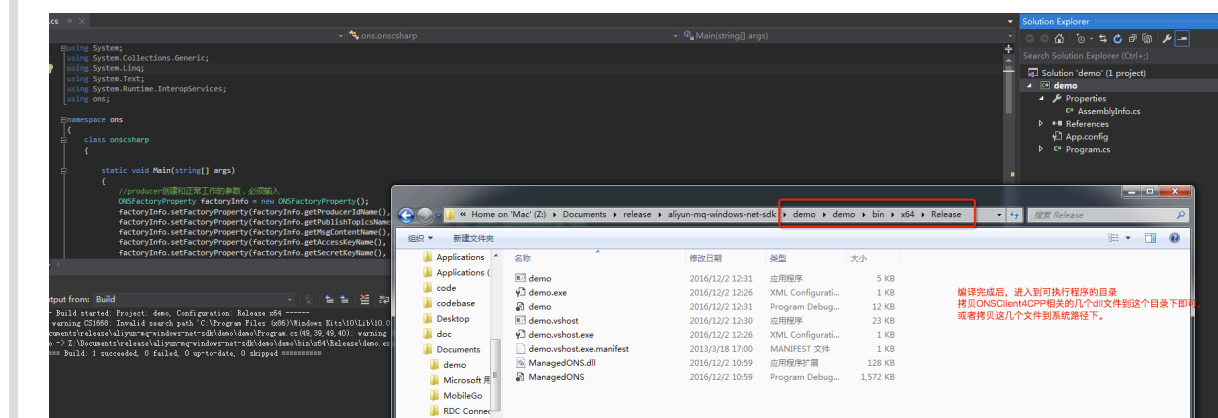
1. 使用 Visual Studio 2015 创建自己的项目。
2. 右键单击项目选择属性>配置管理器, 设置活动解决方案配置为 release, 活动解决方案平台为 x86 或 x64。
3. C# 工程只需引用 ManagedONS.dll, ManagedONS.dll 自动加载 ONSClnt4CPP.dll。
4. C# 工程属性设置:

- 1). 右键单击 C# 工程选择属性>应用程序，将目标框架设为 .NET Framework 4.5 或更高版本；
- 2). 右键单击 C# 工程选择属性>生成>配置，设置为 **release**；
- 3). 右键单击 C# 工程选择属性>生成>目标平台，设置为 **x64**；
- 4). 右键单击 C# 工程选择属性>生成>输出>输出路径，即程序运行目录 /A，设置为与所有 dll 所保存的目录保持一致，拷贝 **64** 位 **lib** 目录下的所有文件到运行目录 /A；
- 5). 右键单击 C# 工程选择属性>调试，打开启用本机调试，否则可能会找不到 **ONSCClient4CPP.dll** 等程序集。

● 非 VS2015 使用 .NET SDK 特殊配置

右键单击 C# 工程选择属性>调试，勾选启用非托管调试，否则可能会找不到 **ONSCClient4CPP.dll** 等程序集。

注意：提供了设置好的demo，操作如下



FAQ

● .NET 端如何消费消息打印消息内容

```
public override Action consume(ref Message value)
{
    byte[] bbody = value.GetBody();
    string body = Encoding.UTF8.GetString(bbody);
    Console.WriteLine(body);
    return ons.Action.CommitMessage;
}
```

● Windows CPP版本的字符编码问题

C++中的string就是普通的多字节ASCII码，在Windows上输入中文，会被转换为gb2312的编码存入到ASCII的string中。在输出的时候会编码为gb2312显示。如果你想发送UTF8编码，就需要将ASCII的string转换为UTF8编码的String，然后发送。然而UTF8是不能直接转换为ASCII，需要使用Unicode编码来转换，首先需要转换为Unicode，然后再转换为UTF8，代码如下：

```

char *multichar_2_utf8(const char *m_string)
{
    int len = 0;
    wchar_t *w_string;      //存储Unicode编码
    char *utf8_string;
    //计算由ansi转换为unicode后, unicode编码的长度
    len = MultiByteToWideChar(CP_ACP, 0, m_string, -1, NULL, 0); //cp_acp指示了
    转换为unicode编码的编码类型
    w_string = (wchar_t *)malloc(2 * len + 2);
    memset(w_string, 0, 2 * len + 2);
    //ansi到unicode转换
    MultiByteToWideChar(CP_ACP, 0, m_string, -1, w_string, len); //cp_acp指示了
    转换为unicode编码的编码类型
    //计算unicode转换为utf8后, utf8编码的长度
    len = WideCharToMultiByte(CP_UTF8, 0, w_string, -1, NULL, 0, NULL,
    NULL); //cp_utf8指示了unicode转换为的类型
    utf8_string = (char *)malloc(len + 1);
    memset(utf8_string, 0, len + 1);
    //unicode到utf8转换
    WideCharToMultiByte(CP_UTF8, 0, w_string, -1, utf8_string, len, NULL,
    NULL); //cp_utf8指示了unicode转换为的类型
    free(w_string);
    return utf8_string;
}

```

接收的时候, 因为服务器是二进制传输, 因此接收到的也是utf8表示的string, 为此需要先将utf8的string, 转换为unicode, 然后将unicode转换为ASCII的string, 输出的时候系统会将unicode转换为gb2312显示, utf8到ascii的转换如下:

```

char *utf8_2_multichar(const char *m_string)
{
    int len = 0;
    wchar_t *w_string;      //存储Unicode编码
    char *ascii_string;

    //计算由utf8转换为unicode后, unicode编码的长度
    len = MultiByteToWideChar(CP_UTF8, 0, m_string, -1, NULL, 0); //CP_UTF8指示
    //了转换为unicode编码的编码类型
    w_string = (wchar_t *)malloc(2 * len + 2);
    memset(w_string, 0, 2 * len + 2);

    //utf8到unicode转换
    MultiByteToWideChar(CP_UTF8, 0, m_string, -1, w_string, len); //CP_UTF8指示
    //了转换为unicode编码的编码类型

    //计算unicode转换为ascii后, ascii编码的长度
    len = WideCharToMultiByte(CP_ACP, 0, w_string, -1, NULL, 0, NULL,
    NULL); //cp_utf8指示了unicode转换为的类型
    ascii_string = (char *)malloc(len + 1);
    memset(ascii_string, 0, len + 1);
    //unicode到ascii转换
    WideCharToMultiByte(CP_ACP, 0, w_string, -1, ascii_string, len, NULL,
    NULL); //cp_utf8指示了unicode转换为的类型
    free(w_string);
    return ascii_string;
}

```