相关ONS 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\_ONS\_USER\_GUIDE.pdf, 本文主要介绍在已开通ONS服务的基础上，如何使用ONS client4cpp 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ONS控制台进行查询。

### ONS 多语言SDK下载

目前ONS支持Linux C++，Windows C++, Windows .NET版本，下载地址分别为:

Linux C++版本下载地址：

<http://onsteam.oss-cn-hangzhou.aliyuncs.com/aliyun-ons-client-cpp.zip>

Linux PHP版本下载地址：

<http://onsteam.oss-cn-hangzhou.aliyuncs.com/aliyun-ons-client-php.zip>

Windows C++版本下载地址:

<http://onsteam.oss-cn-hangzhou.aliyuncs.com/aliyun-ons-client-cpp-windows.zip>

Windows .NET版本下载地址:

<http://onsteam.oss-cn-hangzhou.aliyuncs.com/aliyun-ons-client-net-windows.zip>

### ONS 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；

lib目录：

1.Linux C++版本:

包含client4cpp的动态库和静态库，链接任意一个都可以正常工作；

2.Linux PHP 版本

PHP拓展client4cpp版本，保证稳定性和高效性

ONSClient4PHP.so

libonsclient4cpp.so

使用方法：

1. 拷贝ONSClient4PHP.so和libonsclient4cpp.so到工作目录:/A
2. export LD\_LIBRARY\_PATH=/A :$libonsclient4cpp.so
3. php -d "extension=/A/ONSClient4PHP.so" Producer.php/pushConsumer.php

3.Windows C++版本:

VS2013编译的release 32位和64位版本，其他VS版本也可以使用，共包含： ONSClient4CPP.lib(32位和64位)

ONSClient4CPP.dll(32位和64位)

ONSClient4CPP.pdb(32位和64位)

msvcp120.dll,

msvcr120.dll

运行时，需要拷贝以上所有文件到运行目录

4.Windows .NET版本:

我们提供的.NET版本是基于ONS 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VS2013编译的release 32位和64位版本DLL，其他VS版本也可以使用，共包含:

ONSClient4CPP.lib(32位和64位)

ONSClient4CPP.dll(32位和64位)

ONSClient4CPP.pdb(32位和64位)

ManagedONS.dll(32位和64位)

ManagedONS.pdb(32位和64位)

msvcp120.dll,

msvcr120.dll

example目录： 包含producer和pushConsumer的用法实例

实际应用过程中，请与实例中调用各个接口的顺序保持一致

readme文档： 描述producer和pushConsumer正常工作必须调用的接口描述

### ONS .NET 工程配置

#### 3.1 其他VS版本使用.NET SDK

.NET SDK使用VS2013编译，其他VS版本使用SDK的注意事项：

1>. C/C++---代码生成—运行时检查：默认

2>. 清单工具—输入和输出—嵌入清单：否

3>. 拷贝msvcp120.dll和msvcr120.dll到运行目录或者C:\Windows\System32目录

#### 3.2 配置C#工程的注意事项

1. 根据需要，拷贝32位或者64位lib目录下的所有文件到运行目录
2. C#工程只需引用ManagedONS.dll,ManagedONS.dll自动加载ONSClient4CPP.dll.
3. C#工程属性设置(没有特殊说明，代表32位和64位的设置一致)

<1> 应用程序：

目标框架：.NET Framework 4 Client Profile或更高版本

<2> 生成

配置：设置为release，

平台：

如果使用32位DLL：设置为x86

如果使用64位DLL：设置为x64

<3> 生成-目标平台：

如果使用32位DLL：设置为x86

如果使用64位DLL：设置为x64

<4> 生成-输出-输出路径：即程序运行目录，设置为与所有dll所保存的目录保持一致

<5> 调试: 打开启用本机调试, 否则可能会找不到ONSClient4CPP.dll等程序集

#### 3.3 配置ASP .NET WEB工程的注意事项

##### 3.3.1 配置ASP .NET WEB应用程序工程

<1> 应用程序：

目标框架：.NET Framework 4 Client Profile或更高版本

<2> 生成

配置：设置为release，

平台：

如果使用32位DLL：设置为x86

如果使用64位DLL：设置为x64

<3> 生成-目标平台：

如果使用32位DLL：设置为x86

如果使用64位DLL：设置为x64

<4> 打包/发布 web：

按照下图所示，配置web发布：



##### 3.3.2 IIS加载32位和64位ASP .NET WEB的共同配置

1. 添加引用ManagedONS.dll;
2. 配置web.config:

按照下面的配置，添加NativePath:  
   <appSettings>  
    <add key="NativePath" value="C:\MyNativeDLLs"/>//value: ONSClient4CPP.dll放置的目录  
   </appSettings>

1. 配置global.asax:  
   添加下面的配置到Application\_Start(object sender, EventArgs e)函数中:  
   String \_path = String.Concat(System.Environment.GetEnvironmentVariable("PATH"), ";", ConfigurationSettings.AppSettings["NativePath"]);  
   System.Environment.SetEnvironmentVariable("PATH", \_path, EnvironmentVariableTarget.Process);

##### 3.3.3 IIS加载64位ASP .NET WEB应用的额外设置

1>. 启动Visual Studio自带的IISExpress设置

Visual Studio自带的IIS Express默认是32位的。如果需要Visual Studio使用IIS Express调试使用64位程序集的ASP.Net应用，应使用64位版本的IIS Express。设置方法如下：

（1）关闭正在运行的IIS Express实例；

（2）单击VS主菜单上的“工具”-“选项”命令，打开选项对话框；

（3）在选项对话框目录树中“项目和解决方案”-“Web项目”标签，勾选“对网站和项目使用IIS Express的64位版本”条目。

2>.Visual Studio发布Web项目时的设置

在Visual Studio发布ASP.Net项目时，如果设置了在发布期间预编译功能，Visual Studio将调用ASP.Net编译器对项目进行编译。

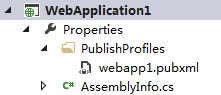


该编译器默认选择的是32位编译器，而Visual Studio未提供任何配置管理界面。更换64位编译器的方法为：

<1>.找到当前发布Web的配置文件名称，本例中的文件名称为webapp1；



<2>. 在解决方案资源管理器的Properies目录，找到对应的配置文件，进行编辑；



在该配置文件中添加下列语句：

<AspnetCompilerPath>

[Microsoft .Net Framework路径]\ Framework64\[版本号]

</AspnetCompilerPath>

例如：

<AspnetCompilerPath>

C:\Windows\Microsoft.NET\Framework64\v4.0.30319

</AspnetCompilerPath>

3> IIS的相应设置

IIS可以针对具体的ASP.Net应用设置是否以64位环境运行，具体的操作方法如下：

（1）部署Web站点后，打开Internet 信息服务(IIS)管理器；

（2）在目录中选择应用程序池，选择目标站点的应用配置，通过右键菜单打开高级设置；

（3）将启用32位应用程序选项设置为False，应用将以64位环境运行。

### ONS CPP SDK样例

Linux和windows C++版本接口完全一致, 以下两节描述的使用方法对两个平台都适用;

#### 4.1 发送消息样例

#include "ONSFactory.h" //必须引用的头文件

#include "ONSClientException.h"

using namespace ons; // ons client4cpp工作的命名空间

main()

{

// 创建producer和发送消息所必需的信息

ONSFactoryProperty factoryInfo;

factoryInfo.setFactoryProperty(ONSFactoryProperty::ProducerId, “producerId ”);

factoryInfo.setFactoryProperty(ONSFactoryProperty::PublishTopics, "msg topic" );

factoryInfo.setFactoryProperty(ONSFactoryProperty::MsgContent, "msg content");

factoryInfo.setFactoryProperty(ONSFactoryProperty::AccessKey, "input accesskey");

factoryInfo.setFactoryProperty(ONSFactoryProperty::SecretKey, "input secretkey" );

factoryInfo.setOnsChannel(CLOUD); 默认值为ALIYUN，聚石塔用户必须设置为CLOUD，阿里云用户不需要设置(如果设置，必须设置为ALIYUN)

// 创建producer

Producer \*producer = ONSFactory::getInstance()->createProducer(factoryInfo);

//在发送消息前，必须调用start方法来启动Producer，只需调用一次即可。

producer->start();

//定义所要发送的消息

Message msg(

//Message Topic

factoryInfo.getPublishTopics(),

//Message Tag,可理解为Gmail中的标签，对消息进行再归类，方便Consumer指定过滤条件在ONS服务器过滤

"TagA",

//Message Body，任何二进制形式的数据，ONS不做任何干预，需要Producer与Consumer协商好一致的序列化和反序列化方式

factoryInfo.getMessageContent()

);

// 设置代表消息的业务关键属性，请尽可能全局唯一。

// 以方便您在无法正常收到消息情况下，可通过ONS Console查询消息并补发。

// 注意：不设置也不会影响消息正常收发

msg.setKey("ORDERID\_100");

//发送消息，只要不抛出异常，就代表发送成功

Try

{

SendResultONS sendResult = pProducer->send(msg);

}

catch(ONSClientException & e)

{

//设置消息发送失败处理方法

}

// 在应用退出前，必须销毁Producer对象，否则会导致内存泄露等问题

producer->shutdown();

}

#### 4.2 订阅消息样例

#include "ONSFactory.h" //必须引用的头文件

using namespace ons; // ons client4cpp工作的命名空间

main()

{

//pushConsumer创建和工作需要的参数，必须输入

ONSFactoryProperty factoryInfo;

factoryInfo.setFactoryProperty(ONSFactoryProperty::ConsumerId, "inputConsumerId");//在ONS控制台申请的consumerId

factoryInfo.setFactoryProperty(ONSFactoryProperty::PublishTopics,"input msg topic" );// 在ONS 控制台申请的msg topic

factoryInfo.setFactoryProperty(ONSFactoryProperty::AccessKey, "inputAccessKey");// ONS AccessKey

factoryInfo.setFactoryProperty(ONSFactoryProperty::SecretKey, "inputSecretKey");//ONS SecretKey

// 创建pushConsumer

PushConsumer\* pushConsumer = ONSFactory::getInstance()->createPushConsumer(factoryInfo);

//指定pushConsumer 订阅的消息topic和tag, 注册消息回调函数

MyMsgListener msgListener;//msgListener继承MessageListener pushConsumer->subscribe(factoryInfo.getPublishTopics(),

"TagA||TagB||TagC||TagD",

&msgListener);//传入地址，ons不负责msgListener的内存释放

//启动pushConsumer

pushConsumer->start();

//销毁pushConsumer, 在应用退出前，必须销毁Consumer 对象，否则会导致内存泄露等问题

pushConsumer->shutdown();

}

// MyMsgListener：创建消费消息的实例

//pushConsumer拉取到消息后，会主动调用该实例的consume 函数

class MyMsgListener : public MessageListener

{

public:

MyMsgListener()

{

}

virtual ~MyMsgListener()

{

}

virtual Action consume(Message message, ConsumeContext context)

{

//根据业务需求，消费消息

return CommitMessage;//返回结果在Action.h定义

}

};

### ONS PHP SDK样例

#### 发送消息样例

<?php

// 创建producer和发送消息所必需的信息

$prop = new ONSFactoryProperty();

$prop->setFactoryProperty(ONSFactoryProperty::ProducerId, “producerId”);

$prop->setFactoryProperty(ONSFactoryProperty::PublishTopics, “msg topic”);

$prop->setFactoryProperty(ONSFactoryProperty::MsgContent, “vincentmsg”);

$prop->setFactoryProperty(ONSFactoryProperty::AccessKey, “xxxx”);// ONS accessKey

$prop->setFactoryProperty(ONSFactoryProperty::SecretKey, “xxxx”);//ONS secretKey

$prop->setOnsChannel(ONSFactoryProperty::CLOUD);//默认值为ONSFactoryProperty::ALIYUN，聚石塔用户必须设置为ONSFactoryProperty::CLOUD，阿里云用户不需要设置(如果设置，必须设置为ONSFactoryProperty::ALIYUN)

$msg = new Message($prop->getPublishTopics(), $msg\_tag, $prop->getMessageContent());

//msg\_tag:可理解为Gmail中的标签，对消息进行再归类，方便Consumer指定过滤条件在ONS服务器过滤

//msg\_body:任何二进制形式的数据，ONS不做任何干预，需要Producer与Consumer协商好一致的序列化和反序列化方式

$producer = ONSFactory::createProducer($prop);//创建producer，为了更适应PHP的用法，producer创建完成后会自动Producer::start，无须像C++/.NET调用start接口

try

{

$result = $producer->send($msg);//发送消息，如果不报异常，就代表发送成功

}

catch(Exception $e)

{

echo $e->getMessage();

}

//为了更适应PHP的使用发放，进程退出时系统会自动调用Producer::shutdown接口，用户无须像C++/.NET那样调用shutdown接口；

#### 订阅消息样例

<?php

// MyListener：创建消费消息的实例

//pushConsumer拉取到消息后，会主动调用该实例的consume 函数

class MyListener implements MessageListener{

public function consume($msg ){

//通过Message.h查看返回消息的具体细节

return MessageListener::CommitMessage;//如果消费失败，则返回：MessageListener::ReconsumeLater

}

}//pushConsumer创建和工作需要的参数，必须输入

$prop = new ONSFactoryProperty();

$prop->setFactoryProperty(ONSFactoryProperty::ConsumerId, “consumerId”);

$prop->setFactoryProperty(ONSFactoryProperty::PublishTopics, “msg topic”);

$prop->setFactoryProperty(ONSFactoryProperty::AccessKey, “xxx”);// ONS AccessKey

$prop->setFactoryProperty(ONSFactoryProperty::SecretKey, “xxx”);//ONS secretkey

$prop->setOnsChannel(ONSFactoryProperty::CLOUD);//默认值为ONSFactoryProperty::ALIYUN，聚石塔用户必须设置为ONSFactoryProperty::CLOUD，阿里云用户不需要设置(如果设置，必须设置为ONSFactoryProperty::ALIYUN)

// 创建pushConsumer

$consumer = ONSFactory::createPushConsumer($prop);

//指定pushConsumer 订阅的消息topic和tag, 注册消息回调函数

$listener = new MyListener();

$consumer->subscribe($prop->getPublishTopics(),"TagA||TagB||TagC||TagD",$listener);

//启动pushConsumer

$consumer->start();

//注：因为PHP在多线程环境下运行不够稳定，所以consumer启动之后，主线程最好sleep，交给msgListener线程进行消息接收；

//销毁pushConsumer, 在应用退出前，必须销毁Consumer 对象，否则会导致内存泄露等问题

$consumer->shutdown();

### ONS Windows .NET SDK样例

#### 发送消息样例

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Runtime.InteropServices;

using ons;

namespace ons

{

class onscsharp

{

static void Main(string[] args)

{

//producer创建和正常工作的参数，必须输入

ONSFactoryProperty factoryInfo = new ONSFactoryProperty();

factoryInfo.setFactoryProperty(factoryInfo.getProducerIdName(), "PID\_xxxx ");

factoryInfo.setFactoryProperty(factoryInfo.getPublishTopicsName(), "xxxxxxxx");

factoryInfo.setFactoryProperty(factoryInfo.getMsgContentName(), "input msg content"); factoryInfo.setFactoryProperty(factoryInfo.getAccessKeyName(), "xxxxx");//ONS AccessKey

factoryInfo.setFactoryProperty(factoryInfo.getSecretKeyName(),"xxxxxx");//ONS SecretKey

factoryInfo.setOnsChannel(ONSChannel.CLOUD);//默认值为ONSChannel.ALIYUN，聚石塔用户必须设置为CLOUD，阿里云用户不需要设置(如果设置，必须设置为ALIYUN)

//创建producer

ONSFactory onsfactory = new ONSFactory();

Producer pProducer = onsfactory. getInstance ().createProducer(factoryInfo);

//在发送消息前，必须调用start方法来启动Producer，只需调用一次即可

pProducer.start();

Message msg = new Message(

//Message Topic

factoryInfo.getPublishTopics(),

//Message Tag

"TagA",

//Message Body

factoryInfo.getMessageContent()

);

// 设置代表消息的业务关键属性，请尽可能全局唯一。

// 以方便您在无法正常收到消息情况下，可通过ONS Console查询消息并补发。

// 注意：不设置也不会影响消息正常收发

msg.setKey("ORDERID\_100");

//发送消息，只要不抛出异常，就代表发送成功

try

{

SendResultONS sendResult = pProducer.send(msg);

}

catch(ONSClientException e)

{

//发送失败处理

}

// 在应用退出前，必须销毁Producer对象，否则会导致内存泄露等问题

pProducer.shutdown();

}

}

#### 订阅消息样例

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Runtime.InteropServices;

using ons;

using std;

namespace ons

{

//pushConsumer拉取到消息后，会主动调用该实例的consume 函数

public class MyMsgListener : MessageListener

{

public MyMsgListener()

{

}

~MyMsgListener()

{

}

public override Action consume(ref Message value)

{//形参value是返回的消息实例，可以根据业务逻辑提取Message的各个字段

Console.WriteLine("\nCallback topic: {0}, tag:{1}, key:{2}, msgId:{3},msgbody:{4}",

value.getTopic(),value.getTag(),value.getKey(),value.getMsgID(),value.getBody());

return ons.Action.CommitMessage;

}

}

class onscsharp

{

static void Main(string[] args)

{

//pushConsumer创建和工作需要的参数，必须输入

ONSFactoryProperty factoryInfo = new ONSFactoryProperty();

factoryInfo.setFactoryProperty(factoryInfo.getConsumerIdName(), "CID\_xxx");

factoryInfo.setFactoryProperty(factoryInfo.getPublishTopicsName(), "xxxxx");

factoryInfo.setFactoryProperty(factoryInfo.getAccessKeyName(),"xx");//ONS AccessKey

factoryInfo.setFactoryProperty(factoryInfo.getSecretKeyName(), "xxxx");// ONS SecretKey

factoryInfo.setOnsChannel(ONSChannel.CLOUD);//默认值为ONSChannel.ALIYUN，聚石塔用户必须设置为CLOUD，阿里云用户不需要设置(如果设置，必须设置为ALIYUN)

//create consumer

ONSFactory onsfactory = new ONSFactory();

PushConsumer pConsumer = onsfactory. getInstance ().createPushConsumer(factoryInfo);

//register msg listener and subscribe msg topic

MessageListener msgListener = new MyMsgListener ();

pConsumer.subscribe(factoryInfo.getPublishTopics(), "\*", ref msgListener);

//start consumer

pConsumer.start();

//consumer启动后，会自动拉取消息，拉取到消息后，会自动调用MyMsgListener实例的consume函数；

//确定消费完成后，调用shutdown函数；在应用退出前，必须销毁Consumer 对象，否则会导致内存泄露等问题

pConsumer.shutdown();

}

}

}

### 用VS编译出可独立运行的程序

如何在未安装VS的ECS机器上使用C++和.NET ONS SDK在VS2013编译出的可执行文件？

正常运行可执行文件的前提只有一个：安装运行使用 **VisualStudio2013** 生成的 **C++** 应用程序时所需的运行时组件

VS2013运行组件下载地址：

<http://www.microsoft.com/zh-cn/download/details.aspx?id=40784>

如果使用其他版本的VS编译，请搜索并安装其他版本VS的运行组件