消息服务 MNS

SDK 参考

SDK 参考

Java SDK

下载

MNS Java SDK

建议下载最新发布的SDK版本以获得最佳性能和稳定性。

Version 1.1.8

更新日期

2016-12-15 sdk下载 sample下载

更新内容

1. Topic订阅增加batch短信发送接口；

使用帮助

1. 下载sample并解压aliyun-sdk-mns-samples-1.1.8.zip；
2. 用Eclipse导入Maven工程，选中aliyun-sdk-mns-samples文件夹；
3. 在用户目录(“/home/YOURNAME/“ in Linux or “C:\Users\YOURNAME” in Windows)中创建.aliyun-mns.properties文件，并填写服务地址、AccessKeyID和AccessKeySecret：

```
mns.accountendpoint=http://$accountid.mns.cn-hangzhou.aliyuncs.com
mns.accesskeyid=$your_accesskeyid
mns.accesskeysecret=$your_accesskeysecret
```

pom配置

```
<dependency>
<groupId>com.aliyun.mns</groupId>
<artifactId>aliyun-sdk-mns</artifactId>
<version>1.1.8</version>
<classifier>jar-with-dependencies</classifier>
</dependency>
```

Version 1.1.7

更新日期

2016-08-30 sdk下载 sample下载

更新内容

1. CloudAccount获取MNSClient单例化（ClientConfiguration相同则返回相同的MNSClient实例）；
2. bugfix；
3. Topic订阅增加JSON选项；

使用帮助

1. 下载sample并解压aliyun-sdk-mns-samples-1.1.7.zip；
2. 用Eclipse导入Maven工程，选中aliyun-sdk-mns-samples文件夹；
3. 在用户目录(“/home/YOURNAME/ ” in Linux or “C:\Users\YOURNAME” in Windows)中创建.aliyun-mns.properties文件，并填写服务地址、AccessKeyID和AccessKeySecret：

```
mns.accountendpoint=http://$accountid.mns.cn-hangzhou.aliyuncs.com
mns.accesskeyid=$your_accesskeyid
mns.accesskeysecret=$your_accesskeysecret
```

Version 1.1.5

更新日期

2016-05-26 sdk下载 sample下载

更新内容

1. 增加事务消息队列TransactionQueue；

2. 增加一对多广播消息功能；
3. 新增Java sdk性能测试示例代码；

使用帮助

1. 下载sample并解压aliyun-sdk-mns-samples-1.1.5.zip；
2. 用Eclipse导入Maven工程，选中aliyun-sdk-mns-samples文件夹；
3. 在用户目录(“/home/YOURNAME/“ in Linux or “C:\Users\YOURNAME” in Windows)中创建.aliyun-mns.properties文件，并填写服务地址、AccessKeyID和AccessKeySecret：

```
mns.accountendpoint=http://$accountid.mns.cn-hangzhou.aliyuncs.com
mns.accesskeyid=$your_accesskeyid
mns.accesskeysecret=$your_accesskeysecret
```

4. 运行QueueSample.java ， TopicSample.java ， CloudPullTopicDemo.java（广播消息示例代码），TransactionMessageDemo.java（事务队列完全封装版使用示例），TransactionMessageDemo2.java（事务队列用户自定义版示例，需要用户自定义本地事务，做failover处理）

Version 1.1.4

更新日期

2016-04-25 sdk下载 sample下载

更新内容

1. Subscription支持Queue/Mail Endpoint
2. Topic支持消息过滤
3. BugFix: 修复长轮询请求数超过单路由(maxConnectionsPerRoute)最大链接数导致请求超时

使用帮助

1. 下载sample并解压aliyun-sdk-mns-samples-1.1.4.zip；
2. 用Eclipse导入Maven工程，选中aliyun-sdk-mns-samples文件夹；
3. 在用户目录(“/home/YOURNAME/“ in Linux or “C:\Users\YOURNAME” in Windows)中创建.aliyun-mns.properties文件，并填写服务地址、AccessKeyID和AccessKeySecret：

```
mns.accountendpoint=http://$accountid.mns.cn-hangzhou.aliyuncs.com
mns.accesskeyid=$your_accesskeyid
mns.accesskeysecret=$your_accesskeysecret
```

4. 运行QueueSample.java 和 TopicSample.java 文件

Version 1.1.3

更新日期

2016-03-28 sdk下载 sample下载

更新内容

1. 支持HTTPS
2. 去除Message对象中priority，dequeueCount，delaySeconds的默认初始化值

使用帮助

1. 下载sample并解压aliyun-sdk-mns-samples-1.1.3.zip；
2. 用Eclipse导入Maven工程，选中aliyun-sdk-mns-samples文件夹；
3. 在用户目录(“/home/YOURNAME/“ in Linux or “C:\Users\YOURNAME” in Windows)中创建.aliyun-mns.properties文件，并填写服务地址、AccessKeyID和AccessKeySecret：

```
mns.accountendpoint=http://$accountid.mns.cn-hangzhou.aliyuncs.com
mns.accesskeyid=$your_accesskeyid
mns.accesskeysecret=$your_accesskeysecret
```

4. 运行QueueSample.java 和 TopicSample.java 文件

Version 1.1.2

更新日期

2016-01-30 sdk下载 sample下载

更新内容

1. fixbug: popMessage接口无参数情况下waitseconds取QueueMeta中设置的值，而非0

使用帮助

1. 下载sample并解压aliyun-sdk-mns-samples-1.1.2.zip；
2. 用Eclipse导入Maven工程，选中aliyun-sdk-mns-samples文件夹；
3. 在用户目录(“/home/YOURNAME/“ in Linux or “C:\Users\YOURNAME” in Windows)中创建.aliyun-mns.properties文件，并填写服务地址、AccessKeyID和AccessKeySecret：

```
mns.accountendpoint=http://$accountid.mns.cn-hangzhou.aliyuncs.com
mns.accesskeyid=$your_accesskeyid
mns.accesskeysecret=$your_accesskeysecret
```

4. 运行QueueSample.java 和 TopicSample.java 文件

Version 1.1.1

更新日期

2016-01-19 sdk下载 sample下载

更新内容

1. fixbug: 中文消息使用UTF - 8编码，而非平台默认字符集

使用帮助

1. 下载sample并解压aliyun-sdk-mns-samples-1.1.1.zip；
2. 用Eclipse导入Maven工程，选中aliyun-sdk-mns-samples文件夹；
3. 在用户目录(“/home/YOURNAME/“ in Linux or “C:\Users\YOURNAME” in Windows)中创建.aliyun-mns.properties文件，并填写服务地址、AccessKeyID和AccessKeySecret：

```
mns.accountendpoint=http://$accountid.mns.cn-hangzhou.aliyuncs.com
mns.accesskeyid=$your_accesskeyid
mns.accesskeysecret=$your_accesskeysecret
```

4. 运行QueueSample.java 和 TopicSample.java 文件

Version 1.1.0

更新日期

2016-01-06 sdk下载 sample下载

更新内容

1. 添加对于Topic功能的支持
2. 添加对于STS Token的支持
3. 消息Base64编码支持可选

使用帮助

1. 下载sample并解压aliyun-sdk-mns-samples-1.1.0.zip；
2. 用Eclipse导入Maven工程，选中aliyun-sdk-mns-samples文件夹；
3. 在用户目录(“/home/YOURNAME/“ in Linux or “C:\Users\YOURNAME” in Windows)中创建.aliyun-mns.properties文件，并填写服务地址、AccessKeyID和AccessKeySecret：

```
mns.accountendpoint=http://$accountid.mns.cn-hangzhou.aliyuncs.com
mns.accesskeyid=$your_accesskeyid
mns.accesskeysecret=$your_accesskeysecret
```

4. 运行QueueSample.java 和 TopicSample.java 文件

Version 1.0.5

更新日期

2015-12-02 sdk下载 sample下载

更新内容

1. 修复bug：多CloudAccount对象时导致内存泄漏
2. 依赖的httpasyncclient版本升至4.1

使用帮助

1. 下载sample并解压aliyun-sdk-mns-samples-1.0.5.zip；
2. 用Eclipse导入Maven工程，选中aliyun-sdk-mns-samples文件夹；
3. 在用户目录(“/home/YOURNAME/“ in Linux or “C:\Users\YOURNAME” in Windows)中创建.aliyun-mns.properties文件，并填写服务地址、AccessKeyID和AccessKeySecret：

```
mns.accountendpoint=http://$accountid.mns.cn-hangzhou.aliyuncs.com
mns.accesskeyid=$your_accesskeyid
mns.accesskeysecret=$your_accesskeysecret
```

4. 运行Sample.java文件

Version 1.0.4

更新日期

2015-11-05 sdk下载 sample下载

更新内容

1. 修复bug：网络异常时极端情况下线程挂起
2. 修复bug：关闭空闲连接回收常驻线程

使用帮助

1. 下载sample并解压aliyun-sdk-mns-samples-1.0.4.zip；

2. 用Eclipse导入Maven工程，选中aliyun-sdk-mns-samples文件夹；
3. 在用户目录(“/home/YOURNAME/“ in Linux or “C:\Users\YOURNAME” in Windows)中创建.aliyun-mns.properties文件，并填写服务地址、AccessKeyID和AccessKeySecret：

```
mns.accountendpoint=http://$accountid.mns.cn-hangzhou.aliyuncs.com
mns.accesskeyid=$your_accesskeyid
mns.accesskeysecret=$your_accesskeysecret
```

4. 运行Sample.java文件

Version 1.0.3

更新日期

2015-06-09 sdk下载 sample下载

更新内容

1. 修复bug：大量close wait的连接导致SDK挂起；
2. 增加sample code；
3. API协议升级：” x-mns-version” =” 2015-06-06” ；
4. 支持BatchSendMessage、BatchReceiveMessage、BatchPeekMessage、BatchDeleteMessage；

使用帮助

1. 下载sample并解压aliyun-sdk-mns-samples-1.0.3.zip；
2. 用Eclipse导入Maven工程，选中aliyun-sdk-mns-samples文件夹；
3. 在用户目录(“/home/YOURNAME/“ in Linux or “C:\Users\YOURNAME” in Windows)中创建.aliyun-mns.properties文件，并填写服务地址、AccessKeyID和AccessKeySecret：

```
mns.accountendpoint=http://$accountid.mns.cn-hangzhou.aliyuncs.com
mns.accesskeyid=$your_accesskeyid
mns.accesskeysecret=$your_accesskeysecret
```

4. 运行Sample.java文件

Version 1.0.2

更新日期

2015-03-03 下载

更新内容

1. 优化XML解析逻辑，提升性能

Version 1.0.1

更新日期

2014-12-19 下载

更新内容

1. 缺省线程池修正为50， 修复大规模并发同步时SDK端的性能瓶颈

Version 1.0.0

更新日期

2014-08-01 下载

队列使用手册

本文档介绍如何使用java sdk中的sample代码，完成创建队列、发送消息、接收删除消息和删除队列操作。

1. 准备

- 下载最新版java sdk，解压到aliyun-sdk-mns-samples文件夹；
- 用Eclipse导入Maven工程，选中aliyun-sdk-mns-samples文件夹；
- 在用户目录(Linux系统为”/home/YOURNAME/“目录或者Windows系统为“C:\Users\YOURNAME”目录)中创建.aliyun-mns.properties文件，并填写服务地址、AccessKeyID和AccessKeySecret：
    - AccessKeyId、AccessKeySecret
        - 访问阿里云API的密钥对；
        - 如果使用主账号访问，登陆阿里云 AccessKey 管理页面创建、查看；
        - 如果使用子账号访问，请登录阿里云访问控制控制台查看；
    - Endpoint
        - 访问MNS的接入地址，登陆MNS控制台 单击右上角 **获取Endpoint** 查看;
        - 不同地域的接入地址不同，分为公网以及内网域名；

2. 创建队列

下面给出了创建队列的代码段（队列详细信息请参考**详情**）；

```
public class CreateQueueDemo {
public static void main(String[] args) {
CloudAccount account = new CloudAccount("YourAccessId", "YourAccessKey", "MNSEndpoint");
MNSClient client = account.getMNSClient(); // 在程序中，CloudAccount以及MNSClient单例实现即可，多线程安全

String queueName = "TestQueue";
QueueMeta meta = new QueueMeta(); //生成本地QueueMeta属性，有关队列属性详细介绍见
https://help.aliyun.com/document_detail/27476.html
meta.setQueueName(queueName); // 设置队列名
meta.setPollingWaitSeconds(15);
meta.setMaxMessageSize(2048L);

try {
CloudQueue queue = client.createQueue(meta);
} catch (ClientException ce)
{
System.out.println("Something wrong with the network connection between client and MNS service."
+ "Please check your network and DNS availablity.");
ce.printStackTrace();
} catch (ServiceException se)
{
se.printStackTrace();
logger.error("MNS exception requestId:" + se.getRequestId(), se);
if (se.getErrorCode() != null) {
if (se.getErrorCode().equals("QueueNotExist"))
{
System.out.println("Queue is not exist.Please create before use");
} else if (se.getErrorCode().equals("TimeExpired"))
{
System.out.println("The request is time expired. Please check your local machine timeclock");
}
/*
you can get more MNS service error code from following link:
https://help.aliyun.com/document_detail/mns/api_reference/error_code/error_code.html
*/
}
} catch (Exception e)
{
System.out.println("Unknown exception happened!");
e.printStackTrace();
}

client.close(); // 程序退出时，需主动调用client的close方法进行资源释放
}
}
```

3. 发送消息

创建完队列之后，就可以向队列发送消息

```
public class ProducerDemo {
public static void main(String[] args) {
CloudAccount account = new CloudAccount("YourAccessId", "YourAccessKey", "MNSEndpoint");
MNSClient client = account.getMNSClient(); // 在程序中，CloudAccount以及MNSClient单例实现即可，多线程安全

try {
CloudQueue queue = client.getQueueRef("TestQueue");
Message message = new Message();
message.setMessageBody("I am test message ");
Message putMsg = queue.putMessage(message);
System.out.println("Send message id is: " + putMsg.getMessageId());
} catch (ClientException ce)
{
System.out.println("Something wrong with the network connection between client and MNS service."
+ "Please check your network and DNS availablity.");
ce.printStackTrace();
} catch (ServiceException se)
{
se.printStackTrace();
logger.error("MNS exception requestId:" + se.getRequestId(), se);
if (se.getErrorCode() != null) {
if (se.getErrorCode().equals("QueueNotExist"))
{
System.out.println("Queue is not exist.Please create before use");
} else if (se.getErrorCode().equals("TimeExpired"))
{
System.out.println("The request is time expired. Please check your local machine timeclock");
}
/*
you can get more MNS service error code from following link:
https://help.aliyun.com/document_detail/mns/api_reference/error_code/error_code.html
*/
}
} catch (Exception e)
{
System.out.println("Unknown exception happened!");
e.printStackTrace();
}

client.close(); // 程序退出时，需主动调用client的close方法进行资源释放
}
}
```

4. 接收和删除消息

队列中已经发送了1条消息，下面是从队列中取出并删除该条消息。

```
public class ConsumerDemo {
public static void main(String[] args) {
CloudAccount account = new CloudAccount("YourAccessId", "YourAccessKey", "MNSEndpoint");
```

```
MNSClient client = account.getMNSClient(); // 在程序中，CloudAccount以及MNSClient单例实现即可，多线程安全

try{
CloudQueue queue = client.getQueueRef("TestQueue");
Message popMsg = queue.popMessage();
if (popMsg != null){
System.out.println("message handle: " + popMsg.getReceiptHandle());
System.out.println("message body: " + popMsg.getMessageBodyAsString());
System.out.println("message id: " + popMsg.getMessageId());
System.out.println("message dequeue count:" + popMsg.getDequeueCount());

//删除已经取出消费的消息
queue.deleteMessage(popMsg.getReceiptHandle());
System.out.println("delete message successfully.\n");
}
else{
System.out.println("message not exist in TestQueue.\n");
}
} catch (ClientException ce)
{
System.out.println("Something wrong with the network connection between client and MNS service."
+ "Please check your network and DNS availablity.");
ce.printStackTrace();
} catch (ServiceException se)
{
se.printStackTrace();
logger.error("MNS exception requestId:" + se.getRequestId(), se);
if (se.getErrorCode() != null) {
if (se.getErrorCode().equals("QueueNotExist"))
{
System.out.println("Queue is not exist.Please create before use");
} else if (se.getErrorCode().equals("TimeExpired"))
{
System.out.println("The request is time expired. Please check your local machine timeclock");
}
/*
you can get more MNS service error code from following link:
https://help.aliyun.com/document_detail/mns/api_reference/error_code/error_code.html
*/
}
} catch (Exception e)
{
System.out.println("Unknown exception happened!");
e.printStackTrace();
}

client.close();
}
}
```

5. 删除队列

```
public class DeleteQueueDemo {
public static void main(String[] args) {
```

```
CloudAccount account = new CloudAccount("YourAccessId", "YourAccessKey", "MNSEndpoint");

MNSClient client = account.getMNSClient(); // 在程序中，CloudAccount以及MNSClient单例实现即可，多线程安全

try{
CloudQueue queue = client.getQueueRef("TestQueue");
queue.delete();
} catch (ClientException ce)
{
System.out.println("Something wrong with the network connection between client and MNS service."
+ "Please check your network and DNS availablity.");
ce.printStackTrace();
} catch (ServiceException se)
{
se.printStackTrace();
} catch (Exception e)
{
System.out.println("Unknown exception happened!");
e.printStackTrace();
}

client.close();
}
}
```

主题使用手册

本文档介绍如何使用java sdk中的sample代码，完成创建主题、创建订阅，发布消息、接收消息以及删除主题等操作。

1. 准备

- 下载最新版java sdk，解压到aliyun-sdk-mns-samples文件夹；
- 用Eclipse导入Maven工程，选中aliyun-sdk-mns-samples文件夹；
- 在用户目录(Linux系统为”/home/YOURNAME/“目录或者Windows系统为“C:\Users\YOURNAME”目录)中创建.aliyun-mns.properties文件，并填写服务地址、AccessKeyID和AccessKeySecret：
    - AccessKeyId、AccessKeySecret
        - 访问阿里云API的密钥对；
        - 如果使用主账号访问，登陆阿里云 AccessKey 管理页面创建、查看；
        - 如果使用子账号访问，请登录阿里云访问控制控制台查看；
    - Endpoint
        - 访问MNS的接入地址，登陆MNS控制台 单击右上角 **获取Endpoint** 查看;
        - 不同地域的接入地址不同，分为公网以及内网域名；

2. 创建主题

下面给出了创建主题的代码示例，有关主题详细信息请参考详情;

```
public class CreateTopicDemo {
public static void main(String[] args) {
CloudAccount account = new CloudAccount("YourAccessId", "YourAccessKey", "MNSEndpoint");
MNSClient client = account.getMNSClient(); // 在程序中，CloudAccount以及MNSClient单例实现即可，多线程安全

String topicName = "TestTopic";
TopicMeta meta = new TopicMeta();
meta.setTopicName(topicName);

try {
CloudTopic topic = client.createTopic(meta);
} catch (Exception e)
e.printStackTrace();
System.out.println("create topic error, " + e.getMessage());
}

client.close();
}
}
```

3. 启动HttpEndpoint

aliyun-sdk-mns-samples中有一个HttpEndpoint.java类，简单实现了一个本地启动的Http消息接收端，主要功能包括：

```
对MNS推送消息请求做签名验证；
解析推送请求的消息body体；
返回相应的处理返回码：200；
```

HttpEndpoint的具体实现源码可参考sdk中源码；

4. 创建订阅

对已经创建好的主题Topic进行订阅，在订阅时需要设置对应的推送Endpoint地址（目前支持HTTP、邮件以及队列）、错误重试策略、推送消息格式等；

```
public class SubscribeDemo {
public static void main(String[] args) {
CloudAccount account = new CloudAccount("YourAccessId", "YourAccessKey", "MNSEndpoint");
MNSClient client = account.getMNSClient(); // 在程序中，CloudAccount以及MNSClient单例实现即可，多线程安全

CloudTopic topic = client.getTopicRef("TestTopic");
try {
SubscriptionMeta subMeta = new SubscriptionMeta();
subMeta.setSubscriptionName("TestSub");
```

```
subMeta.setEndpoint(HttpEndpoint.GenEndpointLocal());
subMeta.setNotifyContentFormat(SubscriptionMeta.NotifyContentFormat.XML);
//subMeta.setFilterTag("filterTag"); //设置订阅的filterTag
String subUrl = topic.subscribe(subMeta);
System.out.println("subscription url: " + subUrl);
} catch (Exception e) {
e.printStackTrace();
System.out.println("subscribe/unsubribe error");
}

client.close();
}
}
```

5.发布消息

在创建好主题以及订阅之后，并且已经启动了HttpEndpoint，我们可以向Topic发布消息。

```
public class PublishMessageDemo {
public static void main(String[] args) {
CloudAccount account = new CloudAccount("YourAccessId", "YourAccessKey", "MNSEndpoint");
MNSClient client = account.getMNSClient(); // 在程序中，CloudAccount以及MNSClient单例实现即可，多线程安全

CloudTopic topic = client.getTopicRef("TestTopic");
try {
TopicMessage msg = new Base64TopicMessage(); //可以使用TopicMessage结构，选择不进行Base64加密
msg.setMessageBody("hello world!");
//msg.setMessageTag("filterTag"); //设置该条发布消息的filterTag
msg = topic.publishMessage(msg);
System.out.println(msg.getMessageId());
System.out.println(msg.getMessageBodyMD5());
} catch (Exception e) {
e.printStackTrace();
System.out.println("subscribe error");
}

client.close();
}
}
```

6. 查看HttpEndpoint接收消息

第5步发布了一条消息到Topic中，MNS会将该消息推送到所有的订阅Endpoint，本例中的HttpEndpoint会将接收到的消息打印出来。

7. 取消订阅

如果不需要接收主题的消息，则可以选择取消订阅。

```
public class UnsubscribeDemo {
```

```
public static void main(String[] args) {
CloudAccount account = new CloudAccount("YourAccessId", "YourAccessKey", "MNSEndpoint");
MNSClient client = account.getMNSClient(); // 在程序中，CloudAccount以及MNSClient单例实现即可，多线程安全

CloudTopic topic = client.getTopicRef("TestTopic");
try {
topic.unsubscribe("TestSub");
} catch (Exception e) {
e.printStackTrace();
System.out.println("unsubribe error");
}

client.close();
}
}
```

8.删除主题

最后选择将Topic删除。

```
public class DeleteTopicDemo {
public static void main(String[] args) {
CloudAccount account = new CloudAccount("YourAccessId", "YourAccessKey", "MNSEndpoint");
MNSClient client = account.getMNSClient(); // 在程序中，CloudAccount以及MNSClient单例实现即可，多线程安全

CloudTopic topic = client.getTopicRef("TestTopic");
try {
topic.delete();
} catch (Exception e) {
e.printStackTrace();
System.out.println("delete topic error");
}

client.close();
}
}
```

9.FilterTag使用示例

```
package com.aliyun.mns.samples;


import com.aliyun.mns.client.CloudAccount;
import com.aliyun.mns.client.CloudQueue;
import com.aliyun.mns.client.CloudTopic;
import com.aliyun.mns.client.MNSClient;
import com.aliyun.mns.common.utils.ServiceSettings;
import com.aliyun.mns.model.*;

public class TopicSample {
```

```
public static void main(String[] args) {
CloudAccount account = new CloudAccount(
ServiceSettings.getMNSAccessKeyId(),
ServiceSettings.getMNSAccessKeySecret(),
ServiceSettings.getMNSAccountEndpoint());
MNSClient client = account.getMNSClient();

// step 1 : 创建队列
QueueMeta queueMeta = new QueueMeta();
queueMeta.setQueueName("TestSubForQueue");
CloudQueue queue = client.createQueue(queueMeta);
// step 2 : 创建主题
TopicMeta topicMeta = new TopicMeta();
topicMeta.setTopicName("TestTopic");
CloudTopic topic = client.createTopic(topicMeta);
// step 3 : 创建订阅
SubscriptionMeta subMeta = new SubscriptionMeta();
subMeta.setSubscriptionName("TestForQueueSub");
subMeta.setNotifyContentFormat(SubscriptionMeta.NotifyContentFormat.SIMPLIFIED);
subMeta.setEndpoint(topic.generateQueueEndpoint("TestSubForQueue"));
subMeta.setFilterTag("filterTag");
topic.subscribe(subMeta);
// step 4 : 发布消息
TopicMessage msg = new Base64TopicMessage();
msg.setMessageBody("hello world");
msg.setMessageTag("filterTag");
msg = topic.publishMessage(msg);
// step 5 : 从订阅的队列中获取消息
Message msgReceive = queue.popMessage(30);
System.out.println("ReceiveMessage From TestSubForQueue:");
System.out.println(msgReceive.getMessageBody());
System.exit(0);
}
}
```

并发测试示例代码

本文档介绍了基于Java SDK提供的队列消息发送以及消费的并发测试Case。

1. 用户可指定并发度、运行时间；
2. 使用发送总请求数除以运行时间得到QPS；

1. 初始化

在运行目录创建perf_test_config.properties文本文件，按照如下格式指定参数（其中队列请先行创建好）：

```
Endpoint=
AccessId=
AccessKey=
QueueName=JavaSDKPerfTestQueue
ThreadNum=200
TotalSeconds=180
```

注：

```
ThreadNum是发送/消费的线程数，MNS具备强大的高并发扩展性能；
TotalSeconds是测试Case运行的时间；
```

2. 代码

```
package com.aliyun.mns;

import com.aliyun.mns.client.CloudAccount;
import com.aliyun.mns.client.CloudQueue;
import com.aliyun.mns.client.MNSClient;
import com.aliyun.mns.common.http.ClientConfiguration;
import com.aliyun.mns.model.Message;
import com.aliyun.mns.model.QueueMeta;

import java.io.BufferedInputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Properties;
import java.util.concurrent.atomic.AtomicLong;

public class JavaSDKPerfTest {
private static MNSClient client = null;
private static AtomicLong totalCount = new AtomicLong(0);

private static String endpoint = null;
private static String accessId = null;
private static String accessKey = null;

private static String queueName = "JavaSDKPerfTestQueue";
private static int threadNum = 100;
private static int totalSeconds = 180;

protected static boolean parseConf() {
String confFilePath = System.getProperty("user.dir") + System.getProperty("file.separator") +
"perf_test_config.properties";

BufferedInputStream bis = null;
try {
bis = new BufferedInputStream(new FileInputStream(confFilePath));
if (bis == null) {
System.out.println("ConfFile not opened: " + confFilePath);
```

```
return false;
}
} catch (FileNotFoundException e) {
System.out.println("ConfFile not found: " + confFilePath);
return false;
}

// load file
Properties properties = new Properties();
try {
properties.load(bis);
} catch(IOException e) {
System.out.println("Load ConfFile Failed: " + e.getMessage());
return false;
} finally {
try {
bis.close();
} catch (Exception e) {
// do nothing
}
}

// init the member parameters
endpoint = properties.getProperty("Endpoint");
System.out.println("Endpoint: " + endpoint);
accessId = properties.getProperty("AccessId");
System.out.println("AccessId: " + accessId);
accessKey = properties.getProperty("AccessKey");

queueName = properties.getProperty("QueueName", queueName);
System.out.println("QueueName: " + queueName);
threadNum = Integer.parseInt(properties.getProperty("ThreadNum", String.valueOf(threadNum)));
System.out.println("ThreadNum: " + threadNum);
totalSeconds = Integer.parseInt(properties.getProperty("TotalSeconds", String.valueOf(totalSeconds)));
System.out.println("TotalSeconds: " + totalSeconds);

return true;
}

public static void main(String[] args) {
if (!parseConf()) {
return;
}

ClientConfiguration clientConfiguration = new ClientConfiguration();
clientConfiguration.setMaxConnections(threadNum);
clientConfiguration.setMaxConnectionsPerRoute(threadNum);

CloudAccount cloudAccount = new CloudAccount(accessId, accessKey, endpoint, clientConfiguration);
client = cloudAccount.getMNSClient();

CloudQueue queue = client.getQueueRef(queueName);
queue.delete();

QueueMeta meta = new QueueMeta();
meta.setQueueName(queueName);
```

```
client.createQueue(meta);

// 1. Now check the SendMessage
ArrayList<Thread> threads = new ArrayList<Thread>();
for (int i = 0; i < threadNum; ++i){
Thread thread = new Thread(new Runnable() {
public void run() {
try {
CloudQueue queue = client.getQueueRef(queueName);
Message message = new Message();
message.setMessageBody("Test");
long count = 0;
long startTime = System.currentTimeMillis();

System.out.println(startTime);
long endTime = startTime + totalSeconds * 1000;
while (true) {
for (int i = 0; i < 50; ++i) {
queue.putMessage(message);
}
count += 50;

if (System.currentTimeMillis() >= endTime) {
break;
}
}

System.out.println(System.currentTimeMillis());
System.out.println("Thread" + Thread.currentThread().getName() + ": " + String.valueOf(count));
totalCount.addAndGet(count);
} catch (Exception e) {
e.printStackTrace();
}
}
}, String.valueOf(i));
thread.start();
threads.add(thread);
}

for (int i = 0; i < threadNum; ++i) {
try {
threads.get(i).join();
} catch (InterruptedException e) {
e.printStackTrace();
}
}

System.out.println("SendMessage QPS: ");
System.out.println(totalCount.get() / totalSeconds);

// 2. Now is the ReceiveMessage
threads.clear();
totalCount.set(0);

totalSeconds = totalSeconds / 3; // To ensure that messages in queue are enough for receiving
for (int i = 0; i < threadNum; ++i){
```

```
Thread thread = new Thread(new Runnable() {
public void run() {
try {
CloudQueue queue = client.getQueueRef(queueName);
long count = 0;
long endTime = System.currentTimeMillis() + totalSeconds * 1000;

while (true) {
for (int i = 0; i < 50; ++i) {
queue.popMessage();
}
count += 50;

if (System.currentTimeMillis() >= endTime) {
break;
}
}

System.out.println("Thread" + Thread.currentThread().getName() + ": " + String.valueOf(count));
totalCount.addAndGet(count);
} catch (Exception e) {
e.printStackTrace();
}
}
}, String.valueOf(i));
thread.start();
threads.add(thread);
}

for (int i = 0; i < threadNum; ++i) {
try {
threads.get(i).join();
} catch (InterruptedException e) {
e.printStackTrace();
}
}

System.out.println("ReceiveMessage QPS: ");
System.out.println(totalCount.get() / totalSeconds);

return;
}
}
```

发送消息示例代码

发送消息

```
public class ProducerDemo {

public static void main(String[] args) {
CloudAccount account = new CloudAccount("YourAccessId", "YourAccessKey", "MNSEndpoint");

//这个client仅初始化一次
MNSClient client = account.getMNSClient();

//循环发送10条消息
try{
//TestQueue是你的测试队列，请提前创建
CloudQueue queue = client.getQueueRef("TestQueue");
for (int i = 0; i < 10; i++)
{
Message message = new Message();
message.setMessageBody("I am test message " + i);
message.setPriority(8);
Message putMsg = queue.putMessage(message);
System.out.println("Send message id is: " + putMsg.getMessageId());
}
} catch (ClientException ce)
{
System.out.println("Something wrong with the network connection between client and MNS service."
+ "Please check your network and DNS availablity.");
ce.printStackTrace();
} catch (ServiceException se)
{
se.printStackTrace();
logger.error("MNS exception requestId:" + se.getRequestId(), se);
if (se.getErrorCode() != null) {
if (se.getErrorCode().equals("QueueNotExist"))
{
System.out.println("Queue is not exist.Please create before use");
} else if (se.getErrorCode().equals("TimeExpired"))
{
System.out.println("The request is time expired. Please check your local machine timeclock");
}
/*
you can get more MNS service error code from following link:
https://help.aliyun.com/document_detail/mns/api_reference/error_code/error_code.html
*/
}
} catch (Exception e)
{
System.out.println("Unknown exception happened!");
e.printStackTrace();
}

client.close();
}

}
```

消费消息示例代码

消费消息

```
public class ConsumerDemo {

public static void main(String[] args) {
CloudAccount account = new CloudAccount("YourAccessId", "YourAccessKey", "MNSEndpoint");

//this client need only initialize once
MNSClient client = account.getMNSClient();

//循环消费10条消息
try{
CloudQueue queue = client.getQueueRef("TestQueue");
for (int i = 0; i < 10; i++)
{
Message popMsg = queue.popMessage();
if (popMsg != null){
System.out.println("message handle: " + popMsg.getReceiptHandle());
// 默认会做 base64 解码
System.out.println("message body: " + popMsg.getMessageBodyAsString());
// 消息 body 的原始数据，不做 base64 解码
// System.out.println("message body: " + popMsg.getMessageBodyAsRawString ());
System.out.println("message id: " + popMsg.getMessageId());
System.out.println("message dequeue count:" + popMsg.getDequeueCount());

//删除已经消费的消息
queue.deleteMessage(popMsg.getReceiptHandle());
System.out.println("delete message successfully.\n");
}
else{
System.out.println("message not exist in TestQueue.\n");
}
}
} catch (ClientException ce)
{
System.out.println("Something wrong with the network connection between client and MNS service."
+ "Please check your network and DNS availablity.");
ce.printStackTrace();
} catch (ServiceException se)
{
se.printStackTrace();
logger.error("MNS exception requestId:" + se.getRequestId(), se);
if (se.getErrorCode() != null) {
if (se.getErrorCode().equals("QueueNotExist"))
{
System.out.println("Queue is not exist.Please create before use");
} else if (se.getErrorCode().equals("TimeExpired"))
{
```

```
System.out.println("The request is time expired. Please check your local machine timeclock");
}
/*
you can get more MNS service error code from following link:
https://help.aliyun.com/document_detail/mns/api_reference/error_code/error_code.html
*/
}
} catch (Exception e)
{
System.out.println("Unknown exception happened!");
e.printStackTrace();
}

client.close();
}
}
```

Python SDK

下载

MNS Python SDK

建议下载最新发布的SDK版本以获得最佳性能和稳定性。

注意事项

- Python版本：Python 2.5 及以上版本；
- SDK中Account、Queue、Topic和Subscription结构非线程安全，多线程场景下请独立生成实例。

Version 1.1.5

更新日期

2019-04-26 SDK 下载

更新内容

1. 兼容 Python 3 版本。

Version 1.1.4

更新日期

2017-03-23 SDK 下载

更新内容

1. 主题模型支持短信推送;
2. 队列/主题支持消息包含中文;
3. mnscmd 支持参数指定mnsendpoint、accesskeyid和accesskeysecret;
4. mnscmd 支持指定是否对队列消息做base64编码和解码。

Version 1.1.3

更新日期

2016-09-13 SDK 下载

更新内容

1. 支持透传RequestID到MNS端；
2. Topic推送支持QueueEndpoint和MailEndpoint；
3. 主题消息推送支持json格式；
4. mnscmd 支持 --config_file 指定配置文件。

使用帮助

1. 下载sdk并解压；
2. 进入mns_python_sdk目录，根据README文档安装、使用SDK。

Version 1.1.2

更新日期

2016-05-23 SDK 下载

更新内容

1. Topic推送支持消息过滤；
2. 增加sample目录，包含更详细的示例代码。

使用帮助

1. 下载sdk并解压；
2. 进入mns_python_sdk目录，根据README文档安装、使用SDK；

Version 1.1.1

更新日期

2016-03-28 SDK 下载

更新内容

1. 支持HTTPS
2. 支持Logging

使用帮助

1. 下载sdk并解压；
2. 进入mns_python_sdk目录，根据README文档安装、使用SDK。

Version 1.1.0

更新日期

2016-01-05 SDK 下载

更新内容

1. 添加对于Topic功能的支持。
2. 添加对于STS Token的支持。

使用帮助

1. 下载sdk并解压；
2. 进入mns_python_sdk目录，根据README文档安装、使用SDK。

Version 1.0.2

更新日期

2015-06-09 下载

更新内容

1. 支持SDK安装；
2. 提供mnscmd命令；
3. API协议升级：”x-mns-version”=”2015-06-06”；
4. 支持BatchSendMessage、BatchReceiveMessage、BatchPeekMessage、BatchDeleteMessage。

使用帮助

1. 下载sdk并解压；
2. 进入mns_python_sdk目录，根据README文档安装、使用SDK。

Version 1.0.1

更新日期

2015-02-03 下载

更新内容

1. 统一队列非字符串属性为int类型；
2. 修正SetQueueAttribute的返回status为204。

Version 1.0.0

更新日期

2014-08-01 下载

队列使用手册

本文档介绍如何使用python sdk中的sample代码，完成创建队列、发送消息、接收删除消息和删除队列操作。

1. 准备

- 下载最新版python sdk，解压后进入mns_python_sdk子目录；
- 打开sample.cfg文件，配置AccessKeyID、AccessKeySecret和Endpoint；
    - AccessKeyId、AccessKeySecret
        - 访问阿里云API的密钥对；

- 如果使用主账号访问，登陆阿里云 AccessKey 管理页面创建、查看；
- 如果使用子账号访问，请登录阿里云访问控制控制台查看；

- Endpoint
  - 访问MNS的接入地址，登陆MNS控制台 单击右上角 **获取Endpoint** 查看;
  - 不同地域的接入地址不同；
- SecurityToken
  - 阿里云访问控制服务提供的短期访问权限凭证，直接使用阿里云账号或者子账号访问不需要配置该项，了解详情;

- 进入sample目录，后续使用的脚本都在这里；

2. 创建队列

运行 createqueue.py 创建队列；
默认创建的队列名称是 MySampleQueue，也可以通过参数指定队列名称；
队列详细信息请参考详情;

- 运行

```
$python createqueue.py MyQueue1
Create Queue Succeed! QueueName:MyQueue1
```

- 核心代码

endpoint，accid，acckey和token从第1步的配置文件中读取；

```
#init my_account, my_queue
my_account = Account(endpoint, accid, acckey, token)
queue_name = sys.argv[1] if len(sys.argv) > 1 else "MySampleQueue"
my_queue = my_account.get_queue(queue_name)

#you can get more information of QueueMeta from mns/queue.py
queue_meta = QueueMeta()
try:
queue_url = my_queue.create(queue_meta)
print "Create Queue Succeed! QueueName:%s\n" % queue_name
except MNSExceptionBase, e:
if e.type == "QueueAlreadyExist":
print "Queue already exist, please delete it before creating or use it directly."
sys.exit(0)
print "Create Queue Fail! Exception:%s\n" % e
```

3. 发送消息

运行sendmessage.py，发送多条消息到队列中；
如果第2步指定了队列名称，这里同样通过参数指定队列名称；
消息详细信息请参考详情;

- 运行

```
$python sendmessage.py MyQueue1
==========Send Message To Queue==========
QueueName:MyQueue1
MessageCount:3

Send Message Succeed! MessageBody:I am test message 0. MessageID:3EBE662B52BC99BC-1-154BD99CCA7-200000001
Send Message Succeed! MessageBody:I am test message 1. MessageID:64B92941FC57837F-2-154BD99CCCE-200000001
Send Message Succeed! MessageBody:I am test message 2. MessageID:3EBE662B52BC99BC-1-154BD99CCF0-200000002
```

- 核心代码

endpoint，accid，acckey和token从第1步的配置文件中读取；

```
#init my_account, my_queue
my_account = Account(endpoint, accid, acckey, token)
queue_name = sys.argv[1] if len(sys.argv) > 1 else "MySampleQueue"
my_queue = my_account.get_queue(queue_name)

#send some messages
msg_count = 3

print "%sSend Message To Queue%s\nQueueName:%s\nMessageCount:%s\n" % (10*"=", 10*"=", queue_name, msg_count)
for i in range(msg_count):
try:
msg_body = "I am test message %s." % i
msg = Message(msg_body)
re_msg = my_queue.send_message(msg)
print "Send Message Succeed! MessageBody:%s MessageID:%s" % (msg_body, re_msg.message_id)
except MNSExceptionBase, e:
if e.type == "QueueNotExist":
print "Queue not exist, please create queue before send message."
sys.exit(0)
print "Send Message Fail! Exception:%s\n" % e
```

4. 接收和删除消息

运行recvdelmessage.py，接收并删除队列中的消息，直到队列为空；
如果第2步指定了队列名称，这里同样通过参数指定队列名称；
程序中receive message使用long polling方式，指定 wait seconds为3秒，因此当队列为空时，程序会等待3秒；
消息详细信息请参考详情;

- 运行

```
$python recvdelmessage.py MyQueue1
==========Receive And Delete Message From Queue==========
QueueName:MyQueue1
WaitSeconds:3

Receive Message Succeed! ReceiptHandle:1-ODU4OTkzNDU5My0xNDYzNDcwNDU4LTEtOA== MessageBody:I am
test message 0. MessageID:3EBE662B52BC99BC-1-154BD99CCA7-200000001
Delete Message Succeed! ReceiptHandle:1-ODU4OTkzNDU5My0xNDYzNDcwNDU4LTEtOA==
Receive Message Succeed! ReceiptHandle:1-ODU4OTkzNDU5NC0xNDYzNDcwNDU4LTEtOA== MessageBody:I am
test message 2. MessageID:3EBE662B52BC99BC-1-154BD99CCF0-200000002
Delete Message Succeed! ReceiptHandle:1-ODU4OTkzNDU5NC0xNDYzNDcwNDU4LTEtOA==
Receive Message Succeed! ReceiptHandle:1-ODU4OTkzNDU5My0xNDYzNDcwNDU4LTItOA== MessageBody:I am
test message 1. MessageID:64B92941FC57837F-2-154BD99CCCE-200000001
Delete Message Succeed! ReceiptHandle:1-ODU4OTkzNDU5My0xNDYzNDcwNDU4LTItOA==
Queue is empty!
```

- 核心代码

endpoint，accid，acckey和token从第1步的配置文件中读取；

```
#init my_account, my_queue
my_account = Account(endpoint, accid, acckey, token)
queue_name = sys.argv[1] if len(sys.argv) > 1 else "MySampleQueue"
my_queue = my_account.get_queue(queue_name)


#receive and delete message from queue util the queue is empty
#set the long polling wait time 3 seoncds by wait_seconds

wait_seconds = 3
print "%sReceive And Delete Message From Queue%s\nQueueName:%s\nWaitSeconds:%s\n" % (10*"=", 10*"=",
queue_name, wait_seconds)
while True:
#receive message
try:
recv_msg = my_queue.receive_message(wait_seconds)
print "Receive Message Succeed! ReceiptHandle:%s MessageBody:%s MessageID:%s" % (recv_msg.receipt_handle,
recv_msg.message_body, recv_msg.message_id)
except MNSExceptionBase,e:
if e.type == "QueueNotExist":
print "Queue not exist, please create queue before receive message."
sys.exit(0)
elif e.type == "MessageNotExist":
print "Queue is empty!"
sys.exit(0)
print "Receive Message Fail! Exception:%s\n" % e
continue

#delete message
try:
my_queue.delete_message(recv_msg.receipt_handle)
print "Delete Message Succeed! ReceiptHandle:%s" % recv_msg.receipt_handle
except MNSException,e:
print "Delete Message Fail! Exception:%s\n" % e
```

5. 删除队列

运行deletequeue.py 删除队列

- 运行

```
$python deletequeue.py MyQueue1
Delete Queue Succeed! QueueName:MyQueue1
```

- 核心代码

endpoint，accid，acckey和token从第1步的配置文件中读取；

```
#init my_account, my_queue
my_account = Account(endpoint, accid, acckey, token)
queue_name = sys.argv[1] if len(sys.argv) > 1 else "MySampleQueue"
my_queue = my_account.get_queue(queue_name)

#delete queue
try:
my_queue.delete()
print "Delete Queue Succeed! QueueName:%s\n" % queue_name
except MNSExceptionBase, e:
print "Delete Queue Fail! Exception:%s\n" % e
```

主题+HttpEndpoint使用手册

主题 + HttpEndpoint 使用手册

本文档介绍如何使用python sdk中的sample代码，完成创建主题、创建订阅、启动 HttpEndpoint、发布消息、查看HttpEndpoint接收消息和删除主题操作。

1. 准备

- 下载最新版python sdk，解压后进入mns_python_sdk子目录；
- 打开sample.cfg文件，配置AccessKeyID、AccessKeySecret和Endpoint；
    - AccessKeyId、AccessKeySecret
        - 访问阿里云API的密钥对；
        - 如果使用主账号访问，登陆阿里云 AccessKey 管理页面创建、查看；
        - 如果使用子账号访问，请登录阿里云访问控制控制台查看；

- Endpoint
- 访问MNS的接入地址，登陆MNS控制台 单击右上角 **获取Endpoint** 查看;
- 不同地域的接入地址不同；
- SecurityToken
- 阿里云访问控制服务提供的短期访问权限凭证，直接使用阿里云账号或者子账号访问不需要配置该项，**了解详情**;

- 进入sample目录，后续使用的脚本都在这里；

2. 创建主题

运行 createtopic.py 创建主题;
默认创建的主题名称是 MySampleTopic，也可以通过参数指定主题名称；
主题详细信息请参考**详情**;

- 运行

```
$python createtopic.py MyTopic1
Create Topic Succeed! TopicName:MyTopic1
```

- 核心代码

endpoint，accid，acckey和token从第1步的配置文件中读取；

```
#init my_account, my_topic
my_account = Account(endpoint, accid, acckey, token)
topic_name = sys.argv[1] if len(sys.argv) > 1 else "MySampleTopic"
my_topic = my_account.get_topic(topic_name)

#you can get more information of TopicMeta from mns/topic.py
topic_meta = TopicMeta()
try:
topic_url = my_topic.create(topic_meta)
print "Create Topic Succeed! TopicName:%s\n" % topic_name
except MNSExceptionBase, e:
if e.type == "TopicAlreadyExist":
print "Topic already exist, please delete it before creating or use it directly."
sys.exit(0)
print "Create Topic Fail! Exception:%s\n" % e
```

3. 启动 HttpEndpoint

运行 simple_http_notify_endpoint.py 启动 HttpEndpoint；
脚本启动后，输出该脚本的监听地址，这个地址后续作为创建订阅的Endpoint参数，用于接收 MNS 推送消息的请求；

服务器功能

- 对 MNS 推送消息请求做签名验证，如果错误，返回MNS 403；
- 解析推送请求的 body，打印到日志中，如果解析错误，返回MNS 400；
- 如果验权和解析 body 均正常，返回 MNS 201；

运行

```
$python simple_http_notify_endpoint.py
Start Endpoint! Address: http://10.101.161.**:8080
```

由于 simple_http_notify_endpoint.py 的代码较多，请直接查看sdk 中的源码。

4. 创建订阅

运行 subscribe.py 创建订阅；
第一个参数指定接收消息的 HttpEndpoint，使用第3步中脚本输出的Address；
第二个参数指定订阅的主题名称，如果第2步中指定了主题名称，这里同样指定主题名称；
第三个参数指定订阅的名称，默认是 MySampleTopic-Sub；
订阅详细信息请参考**详情**。

- 运行

```
$python subscribe.py http://10.101.161.**:8080 MyTopic1 MyTopic1-Sub1
Create Subscription Succeed! TopicName:MyTopic1 SubName:MyTopic1-Sub1 Endpoint:http://10.101.161.**:8080
```

- 核心代码

endpoint，accid，acckey和token从第1步的配置文件中读取；

```
sub_endpoint = sys.argv[1]

#init my_account, my_topic, my_sub
my_account = Account(endpoint, accid, acckey, token)

topic_name = sys.argv[2] if len(sys.argv) > 2 else "MySampleTopic"
my_topic = my_account.get_topic(topic_name)

sub_name = sys.argv[3] if len(sys.argv) > 3 else "MySampleTopic-Sub"
my_sub = my_topic.get_subscription(sub_name)

#you can get more information of SubscriptionMeta from mns/subscription.py
sub_meta = SubscriptionMeta(sub_endpoint)
try:
topic_url = my_sub.subscribe(sub_meta)
print "Create Subscription Succeed! TopicName:%s SubName:%s Endpoint:%s\n" % (topic_name, sub_name,
sub_endpoint)
except MNSExceptionBase, e:
if e.type == "TopicNotExist":
print "Topic not exist, please create topic."
```

```
sys.exit(0)
elif e.type == "SubscriptionAlreadyExist":
print "Subscription already exist, please unsubscribe or use it directly."
sys.exit(0)
print "Create Subscription Fail! Exception:%s\n" % e
```

5. 发布消息

运行publishmessage.py 发布多条消息到主题中；
如果第2步中指定了主题名称，这里同样通过第一个参数指定主题名称；
消息详细信息请参考**详情**;

- 运行

```
$python publishmessage.py MyTopic1
==========Publish Message To Topic==========
TopicName:MyTopic1
MessageCount:3

Publish Message Succeed. MessageBody:I am test message 0. MessageID:F6EA56633844DBFC-1-154BDFB8059-
20000****
Publish Message Succeed. MessageBody:I am test message 1. MessageID:F6EA56633844DBFC-1-154BDFB805F-
20000****
Publish Message Succeed. MessageBody:I am test message 2. MessageID:F6EA56633844DBFC-1-154BDFB8062-
20000****
```

- 核心代码

endpoint，accid，acckey和token从第1步的配置文件中读取；

```
#init my_account, my_topic
my_account = Account(endpoint, accid, acckey, token)
topic_name = sys.argv[1] if len(sys.argv) > 1 else "MySampleTopic"
my_topic = my_account.get_topic(topic_name)

#publish some messages
msg_count = 3
print "%sPublish Message To Topic%s\nTopicName:%s\nMessageCount:%s\n" % (10*"=", 10*"=", topic_name,
msg_count)

for i in range(msg_count):
try:
msg_body = "I am test message %s." % i
msg = TopicMessage(msg_body)
re_msg = my_topic.publish_message(msg)
print "Publish Message Succeed. MessageBody:%s MessageID:%s" % (msg_body, re_msg.message_id)
except MNSExceptionBase,e:
if e.type == "TopicNotExist":
print "Topic not exist, please create it."
sys.exit(1)
print "Publish Message Fail. Exception:%s" % e
```

6. 查看 HttpEndpoint 接收消息

第5步发布了多条消息到主题中，MNS 会将发布的消息推送给第3步启动的 HttpEndpoint；
HttpEndpoint 在接收到消息推送请求后，会记录两种日志：access_log 和 endpoint_log；

- access log
  - 启动 HttpEndpoint 的地方会打印access log，显示推送消息请求的基本信息，从左往右依次是：
    RequestTime RequestVersion ReturnCode URI HTTPVersion RequestLength Host Agent MNSRequestID MNSVersion
  - 除打印到屏幕上，access log会写到日志文件中，文件名格式是access_log.$port，本文档对应的日志文件是access_log.8080；
  - 第5步发布消息对应的推送请求access log如下：

```
$python simple_http_notify_endpoint.py
Start Endpoint! Address: http://10.101.161.**:8080
[17/May/2016 17:10:56]"POST" "201" "/notifications" "HTTP/1.1" "495" "10.101.161.**:8080" "Aliyun Notification Service Agent" "573AE020B2B71CFC6801A6EF" "2015-06-06"
[17/May/2016 17:10:56]"POST" "201" "/notifications" "HTTP/1.1" "495" "10.101.161.**:8080" "Aliyun Notification Service Agent" "573AE020B2B71CFC6801A712" "2015-06-06"
[17/May/2016 17:10:56]"POST" "201" "/notifications" "HTTP/1.1" "495" "10.101.161.**:8080" "Aliyun Notification Service Agent" "573AE020B2B71CFC6801A704" "2015-06-06"
```

- endpoint log
  - 记录每个请求的详细信息，包含完整的header、body以及解析后消息各属性的信息；
  - 日志的文件名格式是 endpoint_log.$port，本文档对应的日志文件是endpoint_log.8080
  - 第5步发布消息对应的推送请求的endpoint log如下：

```
$cat endpoint_log.8080
...
[2016-05-17 17:10:56] [root] [INFO] [simple_http_notify_endpoint.py:47] [1096657216] Notify Message Succeed!
MessageMD5 : 075C3D4AEB2D2F2D6A4C17C9D6DBBEBB
TopicOwner : 126912835662****
PublishTime : 1463476256857
Subscriber : 126912835662****
MessageTag :
SubscriptionName : MyTopic1-Sub1
MessageId : F6EA56633844DBFC-1-154BDFB8059-20000****
Message : I am test message 0.
TopicName : MyTopic1
[2016-05-17 17:10:56] [root] [INFO] [simple_http_notify_endpoint.py:47] [1123260736] Notify Message Succeed!
MessageMD5 : 3BCFB142A3CC597F5D409BFE9DB1B885
TopicOwner : 126912835662****
PublishTime : 1463476256866
Subscriber : 126912835662****
MessageTag :
SubscriptionName : MyTopic1-Sub1
MessageId : F6EA56633844DBFC-1-154BDFB8062-20000****
Message : I am test message 2.
```

```
TopicName : MyTopic1
[2016-05-17 17:10:56] [root] [INFO] [simple_http_notify_endpoint.py:47] [1112770880] Notify Message Succeed!
MessageMD5 : 8356BC7FFBD22CC971BE7FF7427202B6
TopicOwner : 126912835662****
PublishTime : 1463476256863
Subscriber : 126912835662****
MessageTag :
SubscriptionName : MyTopic1-Sub1
MessageId : F6EA56633844DBFC-1-154BDFB805F-20000****
Message : I am test message 1.
TopicName : MyTopic1
```

7. 删除主题

运行deletetopic.py 删除主题；
如果第2步中指定了主题名称，这里同样通过第一个参数指定主题名称；

- 运行

```
$python deletetopic.py MyTopic1
Delete Topic Succeed! TopicName:MyTopic1
```

- 核心代码

```
#init my_account, my_topic
my_account = Account(endpoint, accid, acckey, token)
topic_name = sys.argv[1] if len(sys.argv) > 1 else "MySampleTopic"
my_topic = my_account.get_topic(topic_name)

try:
my_topic.delete()
print "Delete Topic Succeed! TopicName:%s\n" % topic_name
except MNSExceptionBase, e:
print "Delete Topic Fail! Exception:%s\n" % e
```

主题+QueueEndpoint使用手册

本文档介绍如何使用python sdk中的sample代码，完成创建主题、创建QueueEndpoint订阅、创建队列、发布消息、从队列接收删除消息和删除主题操作。

1. 准备

- 下载最新版python sdk，解压后进入mns_python_sdk子目录；

- 打开sample.cfg文件，配置AccessKeyID、AccessKeySecret和Endpoint；
  - AccessKeyId、AccessKeySecret
    - 访问阿里云API的密钥对；
    - 如果使用主账号访问，登陆阿里云 AccessKey 管理页面创建、查看；
    - 如果使用子账号访问，请登录阿里云访问控制控制台查看；
  - Endpoint
    - 访问MNS的接入地址，登陆MNS控制台 单击右上角 获取Endpoint 查看;
    - 不同地域的接入地址不同；
  - SecurityToken
    - 阿里云访问控制服务提供的短期访问权限凭证，直接使用阿里云账号或者子账号访问不需要配置该项，了解详情;
- 进入sample目录，后续使用的脚本都在这里；

2. 创建主题

运行 createtopic.py 创建主题;
默认创建的主题名称是 MySampleTopic，也可以通过参数指定主题名称；
主题详细信息请参考详情;

- 运行

```
$python createtopic.py MyTopic1
Create Topic Succeed! TopicName:MyTopic1
```

- 核心代码

endpoint，accid，acckey和token从第1步的配置文件中读取；

```
#init my_account, my_topic
my_account = Account(endpoint, accid, acckey, token)
topic_name = sys.argv[1] if len(sys.argv) > 1 else "MySampleTopic"
my_topic = my_account.get_topic(topic_name)

#you can get more information of TopicMeta from mns/topic.py
topic_meta = TopicMeta()
try:
topic_url = my_topic.create(topic_meta)
print "Create Topic Succeed! TopicName:%s\n" % topic_name
except MNSExceptionBase, e:
if e.type == "TopicAlreadyExist":
print "Topic already exist, please delete it before creating or use it directly."
sys.exit(0)
print "Create Topic Fail! Exception:%s\n" % e
```

3. 创建队列

运行 createqueue.py 创建队列；

默认创建的队列名称是 MySampleQueue，也可以通过参数指定队列名称；
队列详细信息请参考详情;

- 运行

```
$python createqueue.py MyQueue1
Create Queue Succeed! QueueName:MyQueue1
```

- 核心代码

endpoint，accid，acckey和token从第1步的配置文件中读取；

```
#init my_account, my_queue
my_account = Account(endpoint, accid, acckey, token)
queue_name = sys.argv[1] if len(sys.argv) > 1 else "MySampleQueue"
my_queue = my_account.get_queue(queue_name)

#you can get more information of QueueMeta from mns/queue.py
queue_meta = QueueMeta()
try:
queue_url = my_queue.create(queue_meta)
print "Create Queue Succeed! QueueName:%s\n" % queue_name
except MNSExceptionBase, e:
if e.type == "QueueAlreadyExist":
print "Queue already exist, please delete it before creating or use it directly."
sys.exit(0)
print "Create Queue Fail! Exception:%s\n" % e
```

4. 创建订阅

运行 subscribe.py 创建订阅；
第一个参数指定队列的地域，必须与主题在同一个地域，此处以杭州为例；
第二个参数指定队列的名称，使用第3步创建的队列名称；
第三个参数指定订阅的主题名称，如果第2步中指定了主题名称，这里同样指定主题名称；
第四个参数指定订阅的名称，默认是 MySampleTopic-Sub；
订阅详细信息请参考详情;

- 运行

```
$python subscribe_queueendpoint.py cn-hangzhou MyQueue1 MyTopic1 MyTopic1-Sub1
Create Subscription Succeed! TopicName:MyTopic1 SubName:MyTopic1-Sub1 Endpoint:acs:mns:cn-
hangzhou:127797386164059:queues/MyQueue1
```

- 核心代码

endpoint，accid，acckey和token从第1步的配置文件中读取；

```
region = sys.argv[1]
queue_name = sys.argv[2]
queue_endpoint = TopicHelper.generate_queue_endpoint(region, account_id, queue_name)

#init my_account, my_topic, my_sub
my_account = Account(endpoint, accid, acckey, token)
topic_name = sys.argv[3] if len(sys.argv) > 3 else "MySampleTopic"
my_topic = my_account.get_topic(topic_name)
sub_name = sys.argv[4] if len(sys.argv) > 4 else "MySampleTopic-Sub"
my_sub = my_topic.get_subscription(sub_name)

#you can get more information of SubscriptionMeta from mns/subscription.py
sub_meta = SubscriptionMeta(queue_endpoint, notify_content_format =
SubscriptionNotifyContentFormat.SIMPLIFIED)
try:
topic_url = my_sub.subscribe(sub_meta)
print "Create Subscription Succeed! TopicName:%s SubName:%s Endpoint:%s\n" % (topic_name, sub_name,
queue_endpoint)
except MNSExceptionBase, e:
if e.type == "TopicNotExist":
print "Topic not exist, please create topic."
sys.exit(0)
elif e.type == "SubscriptionAlreadyExist":
print "Subscription already exist, please unsubscribe or use it directly."
sys.exit(0)
print "Create Subscription Fail! Exception:%s\n" % e
```

5. 发布消息

运行publishmessage.py 发布多条消息到主题中；
如果第2步中指定了主题名称，这里同样通过第一个参数指定主题名称；
消息详细信息请参考**详情**;

- 运行

```
$python publishmessage.py MyTopic1
==========Publish Message To Topic==========
TopicName:MyTopic1
MessageCount:3

Publish Message Succeed. MessageBody:I am test message 0. MessageID:F6EA56633844DBFC-1-154BDFB8059-
200000004
Publish Message Succeed. MessageBody:I am test message 1. MessageID:F6EA56633844DBFC-1-154BDFB805F-
200000005
Publish Message Succeed. MessageBody:I am test message 2. MessageID:F6EA56633844DBFC-1-154BDFB8062-
200000006
```

- 核心代码

endpoint，accid，acckey和token从第1步的配置文件中读取；

```
#init my_account, my_topic
```

```
my_account = Account(endpoint, accid, acckey, token)
topic_name = sys.argv[1] if len(sys.argv) > 1 else "MySampleTopic"
my_topic = my_account.get_topic(topic_name)

#publish some messages
msg_count = 3
print "%sPublish Message To Topic%s\nTopicName:%s\nMessageCount:%s\n" % (10*"=", 10*"=", topic_name,
msg_count)

for i in range(msg_count):
try:
msg_body = "I am test message %s." % i
msg = TopicMessage(msg_body)
re_msg = my_topic.publish_message(msg)
print "Publish Message Succeed. MessageBody:%s MessageID:%s" % (msg_body, re_msg.message_id)
except MNSExceptionBase,e:
if e.type == "TopicNotExist":
print "Topic not exist, please create it."
sys.exit(1)
print "Publish Message Fail. Exception:%s" % e
```

6. 从队列获取和删除消息

第5步发布了多条消息到主题中，MNS 会将发布的消息推送给第4步指定的队列中；
运行recvdelmessage.py，接收并删除队列中的消息，直到队列为空；
第一个参数指定队列的名称，使用第4步指定的队列名；
第二个参数指定消息体不做base64解码，因为publish message时未编码；
程序中receive message使用long polling方式，指定 wait seconds为3秒，因此当队列为空时，程序会等待3秒；
消息详细信息请参考**详情**;

```
$python recvdelmessage.py MyQueue1 false
==========Receive And Delete Message From Queue==========
QueueName:MyQueue1
WaitSeconds:3

Receive Message Succeed! ReceiptHandle:1-ODU4OTkzNDU5My0xNDczMzkwMjkyLTEtOA== MessageBody:I am
test message 0. MessageID:E56AE055BAA638AC-1-1570CE43AD3-200000001
Delete Message Succeed! ReceiptHandle:1-ODU4OTkzNDU5My0xNDczMzkwMjkyLTEtOA==
Receive Message Succeed! ReceiptHandle:1-ODU4OTkzNDU5NC0xNDczMzkwMjkyLTEtOA== MessageBody:I am
test message 1. MessageID:E56AE055BAA638AC-1-1570CE43AE0-200000002
Delete Message Succeed! ReceiptHandle:1-ODU4OTkzNDU5NC0xNDczMzkwMjkyLTEtOA==
Receive Message Succeed! ReceiptHandle:1-ODU4OTkzNDU5My0xNDczMzkwMjkyLTItOA== MessageBody:I am
test message 2. MessageID:CDAC88D223C0F9E3-2-1570CE43B2E-200000001
Delete Message Succeed! ReceiptHandle:1-ODU4OTkzNDU5My0xNDczMzkwMjkyLTItOA==
Queue is empty!
```

7. 删除主题

运行deletetopic.py 删除主题；

如果第2步中指定了主题名称，这里同样通过第一个参数指定主题名称；

- 运行

```
$python deletetopic.py MyTopic1
Delete Topic Succeed! TopicName:MyTopic1
```

- 核心代码

```
#init my_account, my_topic
my_account = Account(endpoint, accid, acckey, token)
topic_name = sys.argv[1] if len(sys.argv) > 1 else "MySampleTopic"
my_topic = my_account.get_topic(topic_name)

try:
my_topic.delete()
print "Delete Topic Succeed! TopicName:%s\n" % topic_name
except MNSExceptionBase, e:
print "Delete Topic Fail! Exception:%s\n" % e
```

8. 删除队列

运行deletequeue.py 删除队列

- 运行

```
$python deletequeue.py MyQueue1
Delete Queue Succeed! QueueName:MyQueue1
```

- 核心代码

endpoint，accid，acckey和token从第1步的配置文件中读取；

```
#init my_account, my_queue
my_account = Account(endpoint, accid, acckey, token)
queue_name = sys.argv[1] if len(sys.argv) > 1 else "MySampleQueue"
my_queue = my_account.get_queue(queue_name)

#delete queue
try:
my_queue.delete()
print "Delete Queue Succeed! QueueName:%s\n" % queue_name
except MNSExceptionBase, e:
print "Delete Queue Fail! Exception:%s\n" % e
```

HttpEndpoint 示例代码

这里仅展示HttpEndpoint 部分核心代码，完整的代码请参考 python sdk中 simple_http_notify_endpoint.py。

```
class SimpleHttpNotifyEndpoint(BaseHTTPServer.BaseHTTPRequestHandler):
server_version = "SimpleHttpNotifyEndpoint/" + __version__
access_log_file = "access_log"
msg_type = "XML"

def do_POST(self):
content_length = int(self.headers.getheader('content-length', 0))
self.req_body = self.rfile.read(content_length)
self.msg = NotifyMessage()
logger.info("Headers:%s\nBody:%s" % (self.headers, self.req_body))
if not self.authenticate():
res_code = 403
res_content = "Access Forbidden"
logger.warning("Access Forbidden!\nHeaders:%s\nReqBody:%s\n" % (self.headers, self.req_body))
elif not self.validateBody(self.req_body, self.msg, self.msg_type):
res_code = 400
res_content = "Invalid Notify Message"
logger.warning("Invalid Notify Message!\nHeaders:%s\nReqBody:%s\n" % (self.headers, self.req_body))
else:
res_code = 201
res_content = ""
logger.info("Notify Message Succeed!\n%s" % self.msg)
self.access_log(res_code)
self.response(res_code, res_content)

def authenticate(self):
#get string to signature
service_str = "\n".join(sorted(["%s:%s" % (k,v) for k,v in self.headers.items() if k.startswith("x-mns-")]))
sign_header_list = []
for key in ["content-md5", "content-type", "date"]:
if key in self.headers.keys():
sign_header_list.append(self.headers.getheader(key))
else:
sign_header_list.append("")
str2sign = "%s\n%s\n%s\n%s" % (self.command, "\n".join(sign_header_list), service_str, self.path)

#verify
authorization = self.headers.getheader('Authorization')
signature = base64.b64decode(authorization)
cert_str = urllib2.urlopen(base64.b64decode(self.headers.getheader('x-mns-signing-cert-url'))).read()
pubkey = M2Crypto.X509.load_cert_string(cert_str).get_pubkey()
pubkey.reset_context(md='sha1')
pubkey.verify_init()
pubkey.verify_update(str2sign)
return pubkey.verify_final(signature)
```

```
def validateBody(self, data, msg, type):
if type == "XML":
return self.xml_decode(data, msg)
else:
msg.message = data
return True

def xml_decode(self, data, msg):
if data == "":
logger.error("Data is \"\".")
return False
try:
dom = xml.dom.minidom.parseString(data)
except Exception, e:
logger.error("Parse string fail, exception:%s" % e)
return False

node_list = dom.getElementsByTagName("Notification")
if not node_list:
logger.error("Get node of \"Notification\" fail:%s" % e)
return False

data_dic = {}
for node in node_list[0].childNodes:
if node.nodeName != "#text" and node.childNodes != []:
data_dic[node.nodeName] = str(node.childNodes[0].nodeValue.strip())

key_list = ["TopicOwner", "TopicName", "Subscriber", "SubscriptionName", "MessageId", "MessageMD5",
"Message", "PublishTime"]
for key in key_list:
if key not in data_dic.keys():
logger.error("Check item fail. Need \"%s\"." % key)
return False

msg.topic_owner = data_dic["TopicOwner"]
msg.topic_name = data_dic["TopicName"]
msg.subscriber = data_dic["Subscriber"]
msg.subscription_name = data_dic["SubscriptionName"]
msg.message_id = data_dic["MessageId"]
msg.message_md5 = data_dic["MessageMD5"]
msg.message_tag = data_dic["MessageTag"] if data_dic.has_key("MessageTag") else ""
msg.message = data_dic["Message"]
msg.publish_time = data_dic["PublishTime"]
return True
```

主题——发布消息

本文档介绍使用PythonSDK发布主题消息的示例代码，当需要将消息推送到邮箱或者以短信的方式推送到指定的手机，需要在发布消息设置额外的属性，具体设置方式参考代码。

```
#you can get $accountid from https://account.console.aliyun.com/#/secure
#you can get $accid and $acckey from https://ak-console.aliyun.com/#/accesskey
#you can generate $endpoint: http://$accountid.mns.cn-hangzhou.aliyuncs.com, eg.
http://1234567890123456.mns.cn-hangzhou.aliyuncs.com
my_account = Account("$endpoint", "$accid", "$acckey")
topic_name = "TestTopic"
my_topic = my_account.get_topic(topic_name)

#attributes for Mail
direct_mail = DirectMailInfo(account_name="direct_mail_account_name@aliyun-inc.com",
subject="TestMailSubject", address_type=0, is_html=0, reply_to_address=0)

#attributes for SMS
direct_sms = DirectSMSInfo(free_sign_name="SignName", template_code="TemplateCode", single=False)
direct_sms.add_receiver(receiver="$phone1", params={"name": "Tom"})
direct_sms.add_receiver(receiver="$phone2", params={"name": "David"})

#init TopicMessage
msg_body = "I am test message."
msg = TopicMessage(msg_body, "msg_tag", direct_mail, direct_sms)
try:
re_msg = my_topic.publish_message(msg)
print "Publish Message Succeed. MessageBody:%s MessageID:%s" % (msg_body, re_msg.message_id)
except MNSExceptionBase,e:
if e.type == "TopicNotExist":
print "Topic not exist, please create it."
sys.exit(1)
print "Publish Message Fail. Exception:%s" % e
```

C# SDK

下载

MNS Csharp SDK

建议下载最新发布的 SDK 版本，以达到最佳性能和稳定性。

前置需求

1. 必须是阿里云开发者账户 (参见阿里云官网);

2. 必须开通 MNS 服务(立刻开通)

运行帮助

1. 用 VisualStudio 导入工程，选中 AliyunSDK_MNS_Sample；
2. 在 Sample 工程里可以看到几个 Sample 文件，分别对应不同的操作示例;
3. 填充 Aliyun_MNS_Sample 工程中对应 Sample 文件的 AccessKeyId, SecretAccessKey 和 EndPoint；

```
private const string _accessKeyId = "your access key id";
private const string _secretAccessKey = "your secret access key";
private const string _endpoint = "valid endpoint, 比如http://$AccountId.mns.cn-hangzhou.aliyuncs.com";
```

4. 将想要执行的 Sample 文件设置为 VisualStudio 工程的启动项;
5. 运行;

Release Note

Version 1.3.8

更新日期

2017-08-25 SDK下载

更新内容

1. 添加 RAM STS 功能。

Version 1.3.7

更新日期

2017-03-10 SDK下载

更新内容

1. 修正批量推送的json序列化问题

Version 1.3.6

更新日期

2016-12-19 SDK下载

更新内容

1. 支持短信推送

Version 1.3.5

更新日期

2016-11-1 SDK下载

更新内容

1. 为PublishMessage和Subscribe增加MessageTag的支持

Version 1.3.4

更新日期

2016-10-10 SDK下载

更新内容

1. 取消SDK中无意义的MD5检查

Version 1.3.3

更新日期

2016-06-07 SDK下载

更新内容

1. 发送带有DelaySeconds属性的消息时，response中添加ReceiptHandle

Version 1.3.2

更新日期

2016-05-31 SDK下载

更新内容

1. 修复Subscribe时ContentFormat设置的问题

Version 1.3.1

更新日期

2016-05-18 SDK下载

更新内容

1. 修复SendMessage时Priority设置的问题

Version 1.3.0

更新日期

2016-05-17 SDK下载

更新内容

1. 支持LoggingEnabled属性
2. Topic支持Queue和邮件推送
3. 支持.net framework 3.5

Version 1.1.3

更新日期

2016-03-17 SDK下载

更新内容

1. 为MNSClient添加GetNativeTopic

Version 1.1.2

更新日期

2016-02-19 SDK下载

更新内容

1. 为Topic添加SampleHttpServer, 提供了如何处理通知消息的示例

Version 1.1.1

更新日期

2016-02-18 SDK下载

更新内容

1. 为Topic添加PublishMessage的接口

Version 1.1.0

更新日期

2016-01-05 SDK下载

更新内容

1. 添加对于Topic功能的支持

Version 1.0.0

更新日期

2015-09-10 SDK下载

队列使用手册

本文档介绍如何使用csharp sdk，完成创建队列、发送消息、接收删除消息和删除队列操作。

1. 准备

1. 下载最新版csharp sdk，解压后将工程导入到VisualStudio；

   工程里有4个项目，其中一个是AliyunSDK_MNS，这个就是SDK所在的项目。右击这个项目名，选择重新生成，可以在项目的bin目录下找到生成的Aliyun.MNS.dll

   2.1 其他几个项目里都需要引用这个生成的dll，请配置好其他几个项目的“引用”

在AliyunSDK_MNS_Sample这个项目里，有我们接下来会介绍的队列操作的Sample：SyncOperationSample.cs

3.1 将AliyunSDK_MNS_Sample这个项目设置为启动项，并将SyncOperationSample设置为启动对象

3.2 打开SyncOperationSample.cs文件，在文件的最上几行，配置AccessKeyID、AccessKeySecret和Endpoint；

- AccessKeyId、AccessKeySecret
  - 访问阿里云API的密钥对；
  - 如果使用主账号访问，登陆阿里云 AccessKey 管理页面创建、查看；
  - 如果使用子账号访问，请登录阿里云访问控制控制台查看；
- Endpoint
  - 访问MNS的接入地址，登陆MNS控制台 单击右上角 **获取Endpoint** 查看;
  - 不同地域的接入地址不同；

2. 创建队列

- 如果之前未创建过队列，那么首先需要创建队列。默认创建的队列名称是 myqueue，也可以修改代码指定队列名称；

```
// 1. 这里我们指定了Queue的一些属性，对于Queue的属性，请参考
http://help.aliyun.com/document_detail/27476.html
var createQueueRequest = new CreateQueueRequest
{
QueueName = _queueName,
Attributes =
{
// VisibilityTimeout是最重要的属性，默认为30秒，请务必参考Queue的属性页面里的详细介绍
VisibilityTimeout = 30,
MaximumMessageSize = 40960,
MessageRetentionPeriod = 345600,
// PollingWaitSeconds是非常重要的长轮询超时时间
PollingWaitSeconds = 30
}
};

try
{
// 2. 创建队列
// 2.1 如果不需要特别指定Queue的属性，那么这里也可以直接使用client.CreateQueue(_queueName);
var queue = client.CreateQueue(createQueueRequest);
Console.WriteLine("Create queue successfully, queue name: {0}", queue.QueueName);
}
catch (MNSException me)
{
// 3. 如果创建队列出错，这里可以根据具体的错误Code做处理
// 3.1 也可以分别Catch QueueAlreadyExistException等做单独处理
Console.WriteLine("CreateQueue Failed! ErrorCode: " + me.ErrorCode);
```

```
}
catch (Exception ex)
{
Console.WriteLine("Create queue failed, exception info: " + ex.Message);
}
```

3. 发送消息

- 创建完队列之后，就可以开始发送消息到队列中了；

```
try
{
// 1. 获取Queue的实例
var nativeQueue = client.GetNativeQueue(_queueName);
// 2. 生成SendMessageRequest，可以对Message有一些额外的属性设置
var sendMessageRequest = new SendMessageRequest("阿里云<MessageBody>计算");
// 3. 发送消息
// 3.1 也可以直接使用nativeQueue.SendMessage("MessageBody");
var sendMessageResponse = nativeQueue.SendMessage(sendMessageRequest);
Console.WriteLine("Send message succeed  ");
}
catch (MNSException me)
{
// 3. 如果创建队列出错，这里可以根据具体的错误Code做处理
// 3.1 也可以分别Catch QueueNotExistException等做单独处理
Console.WriteLine("SendMessage Failed! ErrorCode: " + me.ErrorCode);
}
catch (Exception ex)
{
Console.WriteLine("Send message failed, exception info: " + ex.Message);
}
```

4. 接收和删除消息

- 现在队列里已经有了一条消息，我们可以来尝试接收消息。
- 消息服务MNS的Message有一项属性叫做NextVisibleTime。这是一项非常有用和重要的属性，具体描述请参照http://help.aliyun.com/document_detail/27477.html 页面里对应的解释。

```
try
{
// 1. 直接调用receiveMessage函数
// 1.1 receiveMessage函数接受waitSeconds参数，无特殊情况这里都是建议设置为30
// 1.2 waitSeconds非0表示这次receiveMessage是一次http long polling，如果queue内刚好没有message，那么这次request会在server端等到queue内有消息才返回。最长等待时间为waitSeconds的值，最大为30。
var receiveMessageResponse = nativeQueue.ReceiveMessage(30);
// 2. 获取ReceiptHandle，这是一个有时效性的Handle，可以用来设置Message的各种属性和删除Message。具体的解释请参考：help.aliyun.com/document_detail/27477.html 页面里的ReceiptHandle
_receiptHandle = message.ReceiptHandle;
}
catch (MNSException me)
{
```

```
// 3. 像前面的CreateQueue和SendMessage一样，我们认为ReceiveMessage也是有可能出错的，所以这里加上
CatchException并做对应的处理。
Console.WriteLine("ReceiveMessage Failed! ErrorCode: " + me.ErrorCode);
}
catch (Exception ex)
{
Console.WriteLine("ReceiveMessage failed, exception info: " + ex.Message);
}

// 这里是用户自己的处理消息的逻辑。Sample里就直接略过这一步了。
// 如果这里发生了程序崩溃或卡住等异常情况，对应的Message会在VisibilityTimeout之后重新可见，从而可以被其他进程处
理，避免消息丢失。

// 4. 现在消息已经处理完了。我们可以从队列里删除这条消息了。
try
{
// 5. 直接调用deleteMessage即可。
var deleteMessageResponse = nativeQueue.DeleteMessage(_receiptHandle);
}
catch (MNSException me)
{
// 6. 这里CatchException并做异常处理
// 6.1 如果是receiptHandle已经过期，那么ErrorCode是MessageNotExist，表示通过这个receiptHandle已经找不到对应的
消息。
// 6.2 为了保证receiptHandle不过期，VisibilityTimeout的设置需要保证足够消息处理完成。并且在消息处理过程中，也可
以调用changeMessageVisibility这个函数来延长消息的VisibilityTimeout时间。
Console.WriteLine("DeleteMessage Failed! ErrorCode: " + me.ErrorCode);
}
catch (Exception ex)
{
Console.WriteLine("Delete message failed, exception info: " + ex.Message);
}
```

5. 删除队列

- Sample里最后会删除这个测试队列

```
try
{
var deleteQueueResponse = client.DeleteQueue(deleteQueueRequest);
}
catch (MNSException me)
{
Console.WriteLine("DeleteQueue Failed! ErrorCode: " + me.ErrorCode);
}
catch (Exception ex)
{
Console.WriteLine("Delete queue failed, exception info: " + ex.Message);
}
```

主题使用手册

本文档介绍如何使用csharp sdk中的sample代码，完成创建主题、创建订阅、启动 HttpEndpoint、发布消息、查看HttpEndpoint接收消息和删除主题操作。

1. 准备

1. 下载最新版csharp sdk，解压后将工程导入到VisualStudio；

   工程里有4个项目，其中一个是AliyunSDK_MNS，这个就是SDK所在的项目。右击这个项目名，选择重新生成，可以在项目的bin目录下找到生成的Aliyun.MNS.dll

   2.1 其他几个项目里都需要引用这个生成的dll，请配置好其他几个项目的“引用”

   在AliyunSDK_MNS_Sample这个项目里，有我们接下来会介绍的队列操作的
   Sample：SyncTopicOperations.cs

   3.1 将AliyunSDK_MNS_Sample这个项目设置为启动项，并将SyncTopicOperations设置为启动对象

   3.2 打开SyncTopicOperations.cs文件，在文件的最上几行，配置AccessKeyID、AccessKeySecret和Endpoint；

   - AccessKeyId、AccessKeySecret
     - 访问阿里云API的密钥对；
     - 如果使用主账号访问，登陆阿里云 AccessKey 管理页面创建、查看；
     - 如果使用子账号访问，请登录阿里云访问控制控制台查看；
   - Endpoint
     - 访问MNS的接入地址，登陆MNS控制台 单击右上角 **获取Endpoint** 查看;
     - 不同地域的接入地址不同；

2. 创建主题

- 如果之前未创建过主题(Topic)，那么首先需要创建Topic。默认创建的Topic名称是TestCSharpTopic，也可以修改代码指定Topic名称；

```
// 1. 生成一个CreateTopicRequest实例，参数传入topicName。这里可以同时传入TopicAttributes，以便在CreateTopic时同时设置自定义的Topic属性。
var createTopicRequest = new CreateTopicRequest
{
TopicName = _topicName
};
```

```
Topic topic = null;
try
{
topic = client.CreateTopic(createTopicRequest);
Console.WriteLine("Create topic successfully, topic name: {0}", topic.TopicName);
}
catch (MNSException me)
{
// 2. 可能因为网络错误，或者Topic已经存在等原因导致CreateTopic失败，这里CatchException并做对应的处理
// 2.1 也可以分别Catch TopicAlreadyExistException等做单独处理
Console.WriteLine("CreateTopic Failed! ErrorCode: " + me.ErrorCode);
}
catch (Exception ex)
{
Console.WriteLine("Create topic failed, exception info: " + ex.Message);
return;
}
```

3. 启动 HttpEndpoint

- MNS_CSharp_SDK_Test项目下有一个SampleHttpServer.cs；将它设为启动项目并运行即可，这个SampleHttpServer依赖于.net framework 4.5

  请确认本机有公网IP，否则MNS Server无法把消息通过HttpEndpoint推送到你的机器

  功能

  - 对 MNS 推送消息请求做签名验证；
  - 解析推送请求的 body；
  - 返回StatusCode: 200；

  由于 SampleHttpServer 的代码较多，请直接查看sdk 中的源码。

4. 创建订阅

- 创建订阅以告诉MNS Server，Topic里面的消息应该推送给谁
- Sample里使用的是Http的Endpoint

```
try
{
// 1. 生成SubscriptionAttributes，这里第二个参数是Subscription的Endpoint。请将"XXXX"改成实际的IP和Port
// 1.1 这里设置的是刚才启动的http server的地址
// 1.2 更多支持的Endpoint类型可以参考：help.aliyun.com/document_detail/27479.html
SubscribeResponse res = topic.Subscribe(_subscriptionName, "http://XXXX");
// 2. 订阅成功
Console.WriteLine("Subscribe succeed");
}
catch (MNSException me)
{
```

```
// 3. 可能因为网络错误，或者同名的Subscription已存在等原因导致订阅出错，这里CatchException并做对应的处理
Console.WriteLine("Subscribe Failed! ErrorCode: " + me.ErrorCode);
}
catch (Exception ex)
{
Console.WriteLine("Subscribe failed, exception info: " + ex.Message);
}
```

5. 发布消息

- 现在我们可以发布消息到Topic中，并且期待在HttpServer上收到对应的消息。

```
try
{
// 1.1 如果是推送到邮箱，还需要生成PublishMessageRequest并设置MessageAttributes
var response = topic.PublishMessage("message here </asdas\">");
Console.WriteLine("PublishMessage succeed! " + response.MessageId);
}
catch (MNSException me)
{
// 3. 可能因为网络错误等原因导致PublishMessage失败，这里CatchException并做对应处理
Console.WriteLine("PublishMessage Failed! ErrorCode: " + me.ErrorCode);
}
catch (Exception ex)
{
Console.WriteLine("PublishMessage failed, exception info: " + ex.Message);
}
```

6. 查看 HttpEndpoint 接收消息

- 第5步发布了一条消息到Topic中，MNS 会将发布的消息推送给第3步启动的 HttpEndpoint；
- HttpEndpoint 在接收到消息推送请求后，会打印到console；

7. 取消订阅

- 现在我们不需要再接收消息啦，可以告诉MNS Server取消订阅

```
try
{
topic.Unsubscribe(_subscriptionName);
Console.WriteLine("Unsubscribe succeed!");
}
catch (Exception ex)
{
Console.WriteLine("Subscribe failed, exception info: " + ex.Message);
}
```

8. 删除主题

- Sample中，我们最后删除了这个测试用的Topic

```
try
{
client.DeleteTopic(_topicName);
Console.WriteLine("Delete topic succeed");
}
catch (Exception ex)
{
Console.WriteLine("Delete topic failed, exception info: " + ex.Message);
}
```

PHP SDK

下载

MNS PHP SDK

建议下载最新发布的SDK版本以获得最佳性能和稳定性。

Version 1.3.6

更新日期

2019-04-26 SDK下载

更新内容

1. 支持 Composer 安装

前置需求

1. 阿里云开发者账户 (参见www.aliyun.com)
2. 开通了 MNS 服务(http://www.aliyun.com/product/mns)
3. PHP 5.5 及以上版本

Version 1.3.5

更新日期

2017-06-06 SDK下载

更新内容

1. 在 SendMessage 的时候对于 Priority 增加判断

前置需求

1. 阿里云开发者账户 (参见www.aliyun.com)
2. 开通了 MNS 服务(http://www.aliyun.com/product/mns)
3. PHP 5.5 及以上版本

Version 1.3.4

更新日期

2017-04-13 SDK下载

更新内容

1. 支持空变量模板的短信推送

前置需求

1. 阿里云开发者账户 (参见www.aliyun.com)
2. 开通了 MNS 服务(http://www.aliyun.com/product/mns)
3. PHP 5.5 及以上版本

Version 1.3.3

更新日期

2016-12-15 SDK下载

更新内容

1. 支持短信推送，可以查看 TopicTest.php 里面的对应代码

前置需求

1. 阿里云开发者账户 (参见www.aliyun.com);
2. 开通了 MNS 服务(http://www.aliyun.com/product/mns)
3. PHP 5.5 及以上版本

Version 1.3.2

更新日期

2016-11-03 SDK下载

更新内容

1. 增加超时时间设置
2. GetSubscriptionAttr 的 response 里修复对于 Endpoint 的解析

前置需求

1. 阿里云开发者账户 (参见www.aliyun.com);
2. 开通了 MNS 服务(http://www.aliyun.com/product/mns)
3. PHP 5.5 及以上版本

Version 1.3.1

更新日期

2016-05-19 SDK下载

更新内容

1. Samples改进，主要是修改了读取Topic消息的HttpServerSample

前置需求

1. 阿里云开发者账户 (参见www.aliyun.com);
2. 开通了MNS服务(http://www.aliyun.com/product/mns)
3. PHP 5.5 及以上版本

Version 1.3.0

更新日期

2016-02-25 sdk下载

更新内容

1. Subscription增加Queue/Mail推送的支持

前置需求

1. 阿里云开发者账户 (参见www.aliyun.com)
2. 开通了 MNS 服务(http://www.aliyun.com/product/mns)
3. PHP 5.5 及以上版本

运行帮助

1. 下载并解压 SDK 到任意目录
2. 在使用 SDK 的文件里加上一行：require_once(“/path_to_sdk/mns-autoloader.php”)

Version 1.2.2

更新日期

2016-02-25 sdk下载

更新内容

1. 修复了 PHP SDK 里 BatchSendResponse 未能正确返回结果的缺陷。

前置需求

1. 阿里云开发者账户 (参见www.aliyun.com)
2. 开通了 MNS 服务(http://www.aliyun.com/product/mns)
3. PHP 5.5 及以上版本

运行帮助

1. 下载并解压 SDK 到任意目录
2. 在使用 SDK 的文件里加上一行：require_once(“/path_to_sdk/mns-autoloader.php”)

Version 1.2.1

更新日期

2016-02-22 sdk下载

更新内容

1. 在 PHP SDK 里 Queue 的所有 MessageBody 都是被默认做了 Base64 处理。这个版本的 Queue

提供了禁用 Base64 的选项，需要在getQueueRef 的时候传入参数 $base64 = FALSE。

前置需求

1. 阿里云开发者账户 (参见www.aliyun.com);
2. 开通了 MNS 服务(http://www.aliyun.com/product/mns)
3. PHP 5.5 及以上版本

运行帮助

1. 下载并解压 SDK 到任意目录
2. 在使用 SDK 的文件里加上一行：require_once(“/path_to_sdk/mns-autoloader.php”)

Version 1.2.0

更新日期

2016-02-14 sdk下载

更新内容

1. (重要)PublishMessage接口不再对MessageBody做Base64编码

前置需求

1. 阿里云开发者账户 (参见www.aliyun.com);
2. 开通了 MNS 服务(http://www.aliyun.com/product/mns)
3. PHP 5.5 及以上版本

运行帮助

1. 下载并解压 SDK 到任意目录
2. 在使用 SDK 的文件里加上一行：require_once(“/path_to_sdk/mns-autoloader.php”);

Version 1.1.0

更新日期

2016-01-05 sdk下载

更新内容

1. 添加对于 Topic 功能的支持
2. 添加对于 STS Token 的支持

前置需求

1. 阿里云开发者账户 (参见www.aliyun.com)
2. 开通了 MNS 服务(http://www.aliyun.com/product/mns)
3. PHP 5.5 及以上版本

运行帮助

1. 下载并解压 SDK 到任意目录
2. 在使用 SDK 的文件里加上一行：require_once(“/path_to_sdk/mns-autoloader.php”)

Version 1.0.0

更新日期

2015-11-07 sdk下载

前置需求

1. 阿里云开发者账户 (参见www.aliyun.com)
2. 开通了 MNS 服务(http://www.aliyun.com/product/mns)
3. PHP 5.5 及以上版本

运行帮助

1. 下载并解压 SDK 到任意目录
2. 在使用 SDK 的文件里加上一行：require_once(“/path_to_sdk/mns-autoloader.php”)

队列使用手册

本文档介绍如何使用php sdk，完成创建队列、发送消息、接收删除消息和删除队列操作。

1. 准备

- 下载最新版php sdk，解压后进入php_sdk/Samples/Queue子目录；
- 打开CreateQueueAndSendMessage.php文件，在文件的最下几行，配置AccessKeyID、AccessKeySecret和Endpoint；
    - AccessKeyId、AccessKeySecret
        - 访问阿里云API的密钥对；
        - 如果使用主账号访问，登陆阿里云 AccessKey 管理页面创建、查看；

    - 如果使用子账号访问，请登录阿里云访问控制控制台查看；
  - Endpoint
    - 访问MNS的接入地址，登陆MNS控制台 单击右上角 **获取Endpoint** 查看;
    - 不同地域的接入地址不同；
- CreateQueueAndSendMessage的代码顶部有一些设置，在使用SDK的时候需要做同样的设置

```
// require sdk里自带的一个autoload文件即可
require_once(dirname(dirname(dirname(__FILE__))).'/mns-autoloader.php');

// 代码里需要用的一些php class
use AliyunMNS\Client;
use AliyunMNS\Requests\SendMessageRequest;
use AliyunMNS\Requests\CreateQueueRequest;
use AliyunMNS\Exception\MnsException;
```

2. 创建队列

- 如果之前未创建过队列，那么首先需要创建队列。默认创建的队列名称是 CreateQueueAndSendMessageExample，也可以修改代码指定队列名称；

```
// 1. 首先初始化一个client
$this->client = new Client($this->endPoint, $this->accessId, $this->accessKey);

// 2. 生成一个CreateQueueRequest对象。CreateQueueRequest还可以接受一个QueueAttributes参数，用来初始化生成的queue的属性。
// 2.1 对于queue的属性，请参考help.aliyun.com/document_detail/27476.html
$request = new CreateQueueRequest($queueName);
try
{
$res = $this->client->createQueue($request);
// 2.2 CreateQueue成功
echo "QueueCreated! \n";
}
catch (MnsException $e)
{
// 2.3 可能因为网络错误，或者Queue已经存在等原因导致CreateQueue失败，这里CatchException并做对应的处理
echo "CreateQueueFailed: " . $e . "\n";
echo "MNSErrorCode: " . $e->getMnsErrorCode() . "\n";
return;
}
```

3. 发送消息

- 创建完队列之后，就可以开始发送消息到队列中了；

```
// 1. 首先获取Queue的实例
// 1.1 PHP SDK默认会对发送的消息做Base64 Encode，对接收到的消息做Base64 Decode。
// 1.2 如果不希望SDK做这样的Base64操作，可以在getQueueRef的时候，传入参数$base64=FALSE。即$queue = $this-
```

```
>client->getQueueRef($queueName, FALSE);
$queue = $this->client->getQueueRef($queueName);

$messageBody = "test";
// 2. 生成一个SendMessageRequest对象
// 2.1 SendMessageRequest对象本身也包含了DelaySeconds和Priority属性可以设置。
// 2.2 对于Message的属性，请参考help.aliyun.com/document_detail/27477.html
$request = new SendMessageRequest($messageBody);
try
{
$res = $queue->sendMessage($request);
// 3. 消息发送成功
echo "MessageSent! \n";
}
catch (MnsException $e)
{
// 4. 可能因为网络错误，或MessageBody过大等原因造成发送消息失败，这里CatchException并做对应的处理。
echo "SendMessage Failed: " . $e . "\n";
echo "MNSErrorCode: " . $e->getMnsErrorCode() . "\n";
return;
}
```

4. 接收和删除消息

- 现在队列里已经有了一条消息，我们可以来尝试接收消息。
- 消息服务MNS的Message有一项属性叫做NextVisibleTime。这是一项非常有用和重要的属性，具体描述请参照http://help.aliyun.com/document_detail/27477.html 页面里对应的解释。

```
$receiptHandle = NULL;
try
{
// 1. 直接调用receiveMessage函数
// 1.1 receiveMessage函数接受waitSeconds参数，无特殊情况这里都是建议设置为30
// 1.2 waitSeconds非0表示这次receiveMessage是一次http long polling，如果queue内刚好没有message，那么这次request会在server端等到queue内有消息才返回。最长等待时间为waitSeconds的值，最大为30。
$res = $queue->receiveMessage(30);
echo "ReceiveMessage Succeed! \n";
// 2. 获取ReceiptHandle，这是一个有时效性的Handle，可以用来设置Message的各种属性和删除Message。具体的解释请参考：help.aliyun.com/document_detail/27477.html 页面里的ReceiptHandle
$receiptHandle = $res->getReceiptHandle();
}
catch (MnsException $e)
{
// 3. 像前面的CreateQueue和SendMessage一样，我们认为ReceiveMessage也是有可能出错的，所以这里加上CatchException并做对应的处理。
echo "ReceiveMessage Failed: " . $e . "\n";
echo "MNSErrorCode: " . $e->getMnsErrorCode() . "\n";
return;
}

// 这里是用户自己的处理消息的逻辑。Sample里就直接略过这一步了。
// 如果这里发生了程序崩溃或卡住等异常情况，对应的Message会在VisibilityTimeout之后重新可见，从而可以被其他进程处理，避免消息丢失。
```

```
// 4. 现在消息已经处理完了。我们可以从队列里删除这条消息了。
try
{
// 5. 直接调用deleteMessage即可。
$res = $queue->deleteMessage($receiptHandle);
echo "DeleteMessage Succeed! \n";
}
catch (MnsException $e)
{
// 6. 这里CatchException并做异常处理
// 6.1 如果是receiptHandle已经过期，那么ErrorCode是MessageNotExist，表示通过这个receiptHandle已经找不到对应的消息。
// 6.2 为了保证receiptHandle不过期，VisibilityTimeout的设置需要保证足够消息处理完成。并且在消息处理过程中，也可以调用changeMessageVisibility这个函数来延长消息的VisibilityTimeout时间。
echo "DeleteMessage Failed: " . $e . "\n";
echo "MNSErrorCode: " . $e->getMnsErrorCode() . "\n";
return;
}
```

5. 删除队列

- Sample里最后会删除这个测试队列

```
try {
$this->client->deleteQueue($queueName);
echo "DeleteQueue Succeed! \n";
} catch (MnsException $e) {
echo "DeleteQueue Failed: " . $e;
return;
}
```

主题使用手册

本文档介绍如何使用php sdk中的sample代码，完成创建主题、创建订阅、启动 HttpEndpoint、发布消息、查看HttpEndpoint接收消息和删除主题操作。

1. 准备

- 下载最新版php sdk，解压后进入php_sdk/Samples/Topic子目录；
- 打开CreateTopicAndSendMessage.php文件，在文件的最下几行，配置AccessKeyID、AccessKeySecret，Endpoint，以及要推送到的HttpServer的IP和Port；
    - AccessKeyId、AccessKeySecret
        - 访问阿里云API的密钥对；

        - 如果使用主账号访问，登陆阿里云 AccessKey 管理页面创建、查看；
        - 如果使用子账号访问，请登录阿里云访问控制控制台查看；
    - Endpoint
        - 访问MNS的接入地址，登陆MNS控制台 单击右上角 **获取Endpoint** 查看;
        - 不同地域的接入地址不同；
    - ip
        - 需要是公网能访问的IP

- CreateTopicAndSendMessage的代码顶部有一些设置，在使用SDK的时候需要做同样的设置

```
// require sdk里自带的一个autoload文件即可
require_once(dirname(dirname(dirname(__FILE__))).'/mns-autoloader.php');

// 代码里需要用的一些php class
use AliyunMNS\Client;
use AliyunMNS\Model\SubscriptionAttributes;
use AliyunMNS\Requests\PublishMessageRequest;
use AliyunMNS\Requests\CreateTopicRequest;
use AliyunMNS\Exception\MnsException;
```

2. 创建主题

- 如果之前未创建过主题(Topic)，那么首先需要创建Topic。默认创建的Topic名称是 CreateTopicAndPublishMessageExample，也可以修改代码指定Topic名称；

```
// 1. 生成一个CreateTopicRequest实例，参数传入topicName。这里可以同时传入TopicAttributes，以便在CreateTopic时同时设置自定义的Topic属性。
$request = new CreateTopicRequest($topicName);
try
{
$res = $this->client->createTopic($request);
echo "TopicCreated! \n";
}
catch (MnsException $e)
{
// 2. 可能因为网络错误，或者Topic已经存在等原因导致CreateTopic失败，这里CatchException并做对应的处理
echo "CreateTopicFailed: " . $e . "\n";
echo "MNSErrorCode: " . $e->getMnsErrorCode() . "\n";
return;
}
```

3. 启动 HttpEndpoint

- 运行 http_server_sample.php 启动 PHP的内置HttpServer，用来接收MNS Server发送过来的http request；

  具体命令：php -S $ip:$port http_server_sample.php 这里的IP必须是公网能访问的IP

功能

- 对 MNS 推送消息请求做签名验证；
- 对消息内容做MD5验证；
- 解析推送请求的 body；
- 返回StatusCode: 200；

由于 http_server_sample.php 的代码较多，请直接查看sdk 中的源码。

4. 创建订阅

- 创建订阅以告诉MNS Server，Topic里面的消息应该推送给谁
- Sample里使用的是Http的Endpoint

```
$subscriptionName = "SubscriptionExample";
// 1. 生成SubscriptionAttributes，这里第二个参数是Subscription的Endpoint。
// 1.1 这里设置的是刚才启动的http server的地址
// 1.2 更多支持的Endpoint类型可以参考：help.aliyun.com/document_detail/27479.html
$attributes = new SubscriptionAttributes($subscriptionName, 'http://' . $this->ip . ':' . $this->port);

try
{
$topic->subscribe($attributes);
// 2. 订阅成功
echo "Subscribed! \n";
}
catch (MnsException $e)
{
// 3. 可能因为网络错误，或者同名的Subscription已存在等原因导致订阅出错，这里CatchException并做对应的处理
echo "SubscribeFailed: " . $e . "\n";
echo "MNSErrorCode: " . $e->getMnsErrorCode() . "\n";
return;
}
```

5. 发布消息

- 现在我们可以发布消息到Topic中，并且期待在HttpServer上收到对应的消息。

```
$messageBody = "test";
// 1. 生成PublishMessageRequest
// 1.1 如果是推送到邮箱，还需要设置MessageAttributes，可以参照Tests/TopicTest.php里面的testPublishMailMessage
$request = new PublishMessageRequest($messageBody);
try
{
$res = $topic->publishMessage($request);
// 2. PublishMessage成功
echo "MessagePublished! \n";
}
catch (MnsException $e)
```

```
{
// 3. 可能因为网络错误等原因导致PublishMessage失败，这里CatchException并做对应处理
echo "PublishMessage Failed: " . $e . "\n";
echo "MNSErrorCode: " . $e->getMnsErrorCode() . "\n";
return;
}
```

6. 查看 HttpEndpoint 接收消息

- 第5步发布了一条消息到Topic中，MNS 会将发布的消息推送给第3步启动的 HttpEndpoint；
- HttpEndpoint 在接收到消息推送请求后，会通过error_log打印到console；

7. 取消订阅

- 现在我们不需要再接收消息啦，可以告诉MNS Server取消订阅

```
try
{
$topic->unsubscribe($subscriptionName);
echo "Unsubscribe Succeed! \n";
}
catch (MnsException $e)
{
echo "Unsubscribe Failed: " . $e;
return;
}
```

8. 删除主题

- Sample中，我们最后删除了这个测试用的Topic

```
try
{
$this->client->deleteTopic($topicName);
echo "DeleteTopic Succeed! \n";
}
catch (MnsException $e)
{
echo "DeleteTopic Failed: " . $e;
return;
}
```

C++ SDK

MNS C++ SDK

建议下载最新发布的SDK版本以获得最佳性能和稳定性。

Version 1.3.5

更新日期

2017-04-11 SDK下载

更新内容

1. 支持MessageTag功能
2. 支持短信推送功能

前置需求

1. 如果endpoint是https类型，curl版本建议 >= 7.26.0
2. Linux系统g++版本建议 >= 4.1.2，Windows系统需要安装VS2015
3. 安装scons

运行帮助

1. 下载并解压SDK
2. 在SDK目录执行 scons 命令
3. lib会被自动编译到SDK的lib目录

运行Sample

1. 在sample目录修改aliyun-mns.properties，填上正确的Endpoint/AccessId/AccessKey
2. 在sample目录下执行mns_sample

Version 1.3.4

更新日期

2016-12-29 sdk下载

更新内容

1. 更新了签名时使用的Base64Encode方法，避免极端情况下的出错

前置需求

1. 如果endpoint是https类型，curl版本建议 >= 7.26.0
2. Linux系统g++版本建议 >= 4.1.2，Windows系统需要安装VS2015
3. 安装scons

运行帮助

1. 下载并解压SDK
2. 在SDK目录执行 scons 命令
3. lib会被自动编译到SDK的lib目录

运行Sample

1. 在sample目录修改aliyun-mns.properties，填上正确的Endpoint/AccessId/AccessKey
2. 在sample目录下执行mns_sample

Version 1.3.3

更新日期

2016-12-10 sdk下载

更新内容

1. 对于InvalidEndpoint做了容错处理

前置需求

1. 如果endpoint是https类型，curl版本建议 >= 7.26.0
2. Linux系统g++版本建议 >= 4.1.2，Windows系统需要安装VS2015
3. 安装scons

运行帮助

1. 下载并解压SDK
2. 在SDK目录执行 scons 命令
3. lib会被自动编译到SDK的lib目录

运行Sample

1. 在sample目录修改aliyun-mns.properties，填上正确的Endpoint/AccessId/AccessKey
2. 在sample目录下执行mns_sample

Version 1.3.2

更新日期

2016-11-01 sdk下载

更新内容

1. 在ServiceException增加GetMessage函数

前置需求

1. 如果endpoint是https类型，curl版本建议 >= 7.26.0
2. Linux系统g++版本建议 >= 4.1.2，Windows系统需要安装VS2015
3. 安装scons

运行帮助

1. 下载并解压SDK
2. 在SDK目录执行 scons 命令
3. lib会被自动编译到SDK的lib目录

运行Sample

1. 在sample目录修改aliyun-mns.properties，填上正确的Endpoint/AccessId/AccessKey
2. 在sample目录下执行mns_sample

Version 1.3.1

更新日期

2016-07-28 sdk下载

更新内容

1. 为MnsClient增加超时设置

前置需求

1. 如果endpoint是https类型，curl版本建议 >= 7.26.0
2. Linux系统g++版本建议 >= 4.1.2，Windows系统需要安装VS2015
3. 安装scons

运行帮助

1. 下载并解压SDK
2. 在SDK目录执行 scons 命令
3. lib会被自动编译到SDK的lib目录

运行Sample

1. 在sample目录修改aliyun-mns.properties，填上正确的Endpoint/AccessId/AccessKey
2. 在sample目录下执行mns_sample

Version 1.3.0

更新日期

2016-01-05 sdk下载

更新内容

1. Subscription增加Queue/Mail推送的支持
2. 编译方式修改为scons，兼容Windows
3. 部分兼容性改动

前置需求

1. 如果endpoint是https类型，curl版本建议 >= 7.26.0
2. Linux系统g++版本建议 >= 4.1.2，Windows系统需要安装VS2015
3. 安装scons

运行帮助

1. 下载并解压SDK
2. 在SDK目录执行 scons 命令
3. lib会被自动编译到SDK的lib目录

运行Sample

1. 在sample目录修改aliyun-mns.properties，填上正确的Endpoint/AccessId/AccessKey
2. 在sample目录下执行mns_sample

Version 1.1.0

更新日期

2016-01-05 sdk下载

更新内容

1. 添加对于Topic功能的支持
2. 添加对于STS Token的支持

前置需求

1. 阿里云开发者账户 (参见www.aliyun.com);
2. 开通了MNS服务(http://www.aliyun.com/product/mns)
3. 如果endpoint是https类型，curl版本建议 >= 7.26.0
4. g++版本建议 >= 4.1.2

运行帮助

1. 下载并解压SDK
2. 在SDK目录执行 ./configure && make && sudo make install
3. library现在已经默认安装到/usr/local/lib，在不同的系统上可能有不同的路径。请参照自己系统的具体设置，确定是否需要修改ldconfig。

运行Sample

1. 在sample目录修改aliyun-mns.properties，填上正确的Endpoint/AccessId/AccessKey
2. 在sample目录下执行make
3. 运行./mns_sample

Version 1.0.0

更新日期

2015-12-14 sdk下载

前置需求

1. 阿里云开发者账户 (参见www.aliyun.com);
2. 开通了MNS服务(http://www.aliyun.com/product/mns)
3. g++版本建议 >= 4.1.2

运行帮助

1. 下载并解压SDK
2. 在SDK目录执行 ./configure && make && sudo make install
3. library现在已经默认安装到/usr/local/lib，在不同的系统上可能有不同的路径。请参照自己系统的具体设置，确定是否需要修改ldconfig。

执行Sample

1. 在sample目录修改aliyun-mns.properties，填上正确的Endpoint/AccessId/AccessKey
2. 在sample目录下执行make
3. 运行./mns_sample

Go SDK

MNS Go SDK

建议下载最新发布的 SDK 版本以获得最佳性能和稳定性。

Version 1.0.0

更新日期

2019-04-30 SDK 下载

更新内容

- 支持队列模型的相关操作：
    - 创建、修改、获取和删除队列;
    - 发送、查看、消费和删除队列消息，以及修改队列消息的下次可消费时间。
- 支持主题模型的相关操作:
    - 创建、修改和删除主题；
    - 创建和删除订阅；
    - 发送主题消息。

Android SDK

通告

通告：阿里云消息队列（MQ）推出【消息队列 for IoT（LMQ）】，适用于移动互联网以及物联网应用，连接端（浏览器、Android、iOS、智能设备、互动直播、车联网）与云，实现双向通信，万物互联。

- Demo 工程下载
- 如有其它疑问，您可以通过提交工单 进行反馈。

Objective-C (iOS) SDK

通告

通告：阿里云消息队列（MQ）推出【消息队列 for IoT（LMQ）】，适用于移动互联网以及物联网应用，连接端（浏览器、Android、iOS、智能设备、互动直播、车联网）与云，实现双向通信，万物互联。

- Demo 工程下载
- 如有其它疑问，您可以通过提交工单 进行反馈。

其它 SDK

非官方 SDK

阿里云有计划提供 MNS 的 node SDK, 但是暂时没有时间表，在官方版本出来前，您可以直接调用 MNS 提供的 RESTful API。以下是论坛热心 MNS 用户自行封装的非官方 SDK, 非官方 SD K无法得到阿里云的官方支持，仅用于参考学习：

非官方 Golang SDK（已有对应官方 SDK）

- https://github.com/weikaishio/ali_mns

非官方 Node.js SDK

- https://www.npmjs.com/package/ali-mns

非官方 iOS SDK

- https://github.com/JuneQinEasy/AliCloudAPI

非官方 .NET SDK（已有对应官方 SDK）

- https://github.com/saberwang/aliyun-mqs-cshap-library

非官方 PHP SDK（已有对应官方 SDK）

- http://git.oschina.net/xybingbing/aliyunMQS