ALIBABA CLOUD

Alibaba Cloud

消息服务MNS
SDK参考

文档版本：20220509

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.Java SDK

1.1. Java SDK版本说明

建议下载最新发布的SDK版本以获得较高的性能和稳定性。

Version 1.1.9.1

- 更新日期

  2021-12-24

  SDK下载

  sample下载

- 更新内容
  - 去除Log4j依赖，改为使用SLF4J依赖。
- 使用帮助
  i. 下载sample并解压*aliyun-sdk-mns-samples-1.1.9.1.zip*。
  ii. 用Eclipse导入Maven工程，选中*aliyun-sdk-mns-samples*文件夹。
  iii. 在用户目录中创建*.aliyun-mns.properties*文件，并填写服务地址、AccessKeyId和AccessKeySecret。

  > **说明** Linux系统用户目录为*/home/YOURNAME/*，Windows系统用户目录为*C:\Users\YOURNAME*。

  ```
  mns.accountendpoint=http://<yourAccountId>.mns.cn-hangzhou.aliyuncs.com
  mns.accesskeyid=<yourAccessKeyId>
  mns.accesskeysecret=<yourAccessKeySecret>
  ```

- pom配置

```
<dependency>
    <groupId>com.aliyun.mns</groupId>
    <artifactId>aliyun-sdk-mns</artifactId>
    <version>1.1.9.1</version>
</dependency>
```

Version 1.1.9

- 更新日期

  2021-03-18

  SDK下载

  sample下载

- 更新内容
  - 支持OpenService接口。
  - 修复缺陷。

- 使用帮助
  i. 下载sample并解压*aliyun-sdk-mns-samples-1.1.9.zip*。
  ii. 用Eclipse导入Maven工程，选中*aliyun-sdk-mns-samples*文件夹。
  iii. 在用户目录中创建*.aliyun-mns.properties*文件，并填写服务地址、AccessKeyId和AccessKeySecret。

  > **说明** Linux系统用户目录为*/home/YOURNAME/*，Windows系统用户目录为*C:\Users\YOURNAME*。

```
mns.accountendpoint=http://<yourAccountId>.mns.cn-hangzhou.aliyuncs.com
mns.accesskeyid=<yourAccessKeyId>
mns.accesskeysecret=<yourAccessKeySecret>
```

- pom配置

```
<dependency>
    <groupId>com.aliyun.mns</groupId>
    <artifactId>aliyun-sdk-mns</artifactId>
    <version>1.1.9</version>
</dependency>
```

Version 1.1.8

- 更新日期

  2016-12-15

  SDK下载

  sample下载

- 更新内容

  Topic订阅增加batch短信发送接口。

- 使用帮助
  i. 下载sample并解压*aliyun-sdk-mns-samples-1.1.8.zip*。
  ii. 用Eclipse导入Maven工程，选中*aliyun-sdk-mns-samples*文件夹。
  iii. 在用户目录中创建*.aliyun-mns.properties*文件，并填写服务地址、AccessKeyId和AccessKeySecret。

  > **说明** Linux系统用户目录为*/home/YOURNAME/*，Windows系统用户目录为*C:\Users\YOURNAME*。

```
mns.accountendpoint=http://<yourAccountId>.mns.cn-hangzhou.aliyuncs.com
mns.accesskeyid=<yourAccessKeyId>
mns.accesskeysecret=<yourAccessKeySecret>
```

- pom配置

```
<dependency>
    <groupId>com.aliyun.mns</groupId>
    <artifactId>aliyun-sdk-mns</artifactId>
    <version>1.1.8</version>
</dependency>
```

Version 1.1.7

- 更新日期

  2016-08-30

  SDK下载

  sample下载

- 更新内容
  - 多次调用getMNSClient时返回同一个Java对象。
  - 修复缺陷。
  - Topic订阅增加JSON选项。
- 使用帮助
  i. 下载sample并解压*aliyun-sdk-mns-samples-1.1.7.zip*。
  ii. 用Eclipse导入Maven工程，选中*aliyun-sdk-mns-samples*文件夹。
  iii. 在用户目录中创建*.aliyun-mns.properties*文件，并填写服务地址、AccessKeyId和AccessKeySecret。

  > **说明** Linux系统用户目录为*/home/YOURNAME/*，Windows系统用户目录为*C:\Users\YOURNAME*。

  ```
  mns.accountendpoint=http://<yourAccountId>.mns.cn-hangzhou.aliyuncs.com
  mns.accesskeyid=<yourAccessKeyId>
  mns.accesskeysecret=<yourAccessKeySecret>
  ```

- pom配置

```
<dependency>
    <groupId>com.aliyun.mns</groupId>
    <artifactId>aliyun-sdk-mns</artifactId>
    <version>1.1.7</version>
</dependency>
```

Version 1.1.5

- 更新日期

  2016-05-26

  SDK下载

  sample下载

- 更新内容
  - 增加事务消息队列TransactionQueue。

- 增加一对多广播消息功能。
- 新增JAVA SDK性能测试示例代码。

- 使用帮助
  i. 下载sample并解压*aliyun-sdk-mns-samples-1.1.5.zip*。
  ii. 用Eclipse导入Maven工程，选中*aliyun-sdk-mns-samples*文件夹。
  iii. 在用户目录中创建*.aliyun-mns.properties*文件，并填写服务地址、AccessKeyId和AccessKeySecret。

  > **说明** Linux系统用户目录为*/home/YOURNAME/*，Windows系统用户目录为*C:\Users\YOURNAME*。

  ```
  mns.accountendpoint=http://<yourAccountId>.mns.cn-hangzhou.aliyuncs.com
  mns.accesskeyid=<yourAccessKeyId>
  mns.accesskeysecret=<yourAccessKeySecret>
  ```

  iv. 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）。

- pom配置

```
<dependency>
    <groupId>com.aliyun.mns</groupId>
    <artifactId>aliyun-sdk-mns</artifactId>
    <version>1.1.5</version>
</dependency>
```

Version 1.1.4

- 更新日期

  2016-04-25

  SDK下载

  sample下载

- 更新内容
  - 订阅支持设置队列和邮件为Endpoint。
  - 主题支持消息过滤。
  - 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
  i. 下载sample并解压*aliyun-sdk-mns-samples-1.1.4.zip*。
  ii. 用Eclipse导入Maven工程，选中*aliyun-sdk-mns-samples*文件夹。
  iii. 在用户目录中创建*.aliyun-mns.properties*文件，并填写服务地址、AccessKeyId和AccessKeySecret。

> **说明** Linux系统用户目录为*/home/YOURNAME/*，Windows系统用户目录为*C:\Users\YOURNAME*。

```
mns.accountendpoint=http://<yourAccountId>.mns.cn-hangzhou.aliyuncs.com
mns.accesskeyid=<yourAccessKeyId>
mns.accesskeysecret=<yourAccessKeySecret>
```

iv. 运行*QueueSample.java*和*TopicSample.java*文件。

- pom配置

```
<dependency>
    <groupId>com.aliyun.mns</groupId>
    <artifactId>aliyun-sdk-mns</artifactId>
    <version>1.1.4</version>
</dependency>
```

Version 1.1.3

- 更新日期

  2016-03-28

  SDK下载

  sample下载

- 更新内容
  - 支持HTTPS。
  - 去除Message对象中priority、dequeueCount、delaySeconds的默认初始化值。
- 使用帮助
  i. 下载sample并解压*aliyun-sdk-mns-samples-1.1.3.zip*。
  ii. 用Eclipse导入Maven工程，选中*aliyun-sdk-mns-samples*文件夹。
  iii. 在用户目录中创建*.aliyun-mns.properties*文件，并填写服务地址、AccessKeyId和AccessKeySecret。

  > **说明** Linux系统用户目录为*/home/YOURNAME/*，Windows系统用户目录为*C:\Users\YOURNAME*。

  ```
  mns.accountendpoint=http://<yourAccountId>.mns.cn-hangzhou.aliyuncs.com
  mns.accesskeyid=<yourAccessKeyId>
  mns.accesskeysecret=<yourAccessKeySecret>
  ```

  iv. 运行*QueueSample.java*和*TopicSample.java*文件。
- pom配置

```
<dependency>
    <groupId>com.aliyun.mns</groupId>
    <artifactId>aliyun-sdk-mns</artifactId>
    <version>1.1.3</version>
</dependency>
```

Version 1.1.2

- 更新日期

  2016-01-30

  SDK下载

  sample下载

- 更新内容

  修复popMessage接口无参数情况下waitseconds取QueueMeta中设置的值，而非0。

- 使用帮助

  i. 下载sample并解压*aliyun-sdk-mns-samples-1.1.2.zip*。

  ii. 用Eclipse导入Maven工程，选中*aliyun-sdk-mns-samples*文件夹。

  iii. 在用户目录中创建*.aliyun-mns.properties*文件，并填写服务地址、AccessKeyId和AccessKeySecret。

  > **说明** Linux系统用户目录为*/home/YOURNAME/*，Windows系统用户目录为*C:\Users\YOURNAME*。

  ```
  mns.accountendpoint=http://<yourAccountId>.mns.cn-hangzhou.aliyuncs.com
  mns.accesskeyid=<yourAccessKeyId>
  mns.accesskeysecret=<yourAccessKeySecret>
  ```

  iv. 运行*QueueSample.java*和*TopicSample.java*文件。

- pom配置

```
<dependency>
    <groupId>com.aliyun.mns</groupId>
    <artifactId>aliyun-sdk-mns</artifactId>
    <version>1.1.2</version>
</dependency>
```

Version 1.1.1

- 更新日期

  2016-01-19

  SDK下载

  sample下载

- 更新内容

  修复中文消息使用UTF-8编码，而非平台默认字符集。

- 使用帮助

  i. 下载sample并解压*aliyun-sdk-mns-samples-1.1.1.zip*。

  ii. 用Eclipse导入Maven工程，选中*aliyun-sdk-mns-samples*文件夹。

  iii. 在用户目录中创建*.aliyun-mns.properties*文件，并填写服务地址、AccessKeyId和AccessKeySecret。

> **说明** Linux系统用户目录为*/home/YOURNAME/*，Windows系统用户目录为*C:\Users\YOURNAME*。

```
mns.accountendpoint=http://<yourAccountId>.mns.cn-hangzhou.aliyuncs.com
mns.accesskeyid=<yourAccessKeyId>
mns.accesskeysecret=<yourAccessKeySecret>
```

iv. 运行*QueueSample.java*和*TopicSample.java*文件。

- pom配置

```
<dependency>
    <groupId>com.aliyun.mns</groupId>
    <artifactId>aliyun-sdk-mns</artifactId>
    <version>1.1.1</version>
</dependency>
```

Version 1.1.0

- 更新日期

  2016-01-06

  SDK参考

  sample下载

- 更新内容
  - 添加对于Topic功能的支持。
  - 添加对于STS Token的支持。
  - 消息Base64编码支持可选。
- 使用帮助
  i. 下载sample并解压*aliyun-sdk-mns-samples-1.1.0.zip*。
  ii. 用Eclipse导入Maven工程，选中*aliyun-sdk-mns-samples*文件夹。
  iii. 在用户目录中创建*.aliyun-mns.properties*文件，并填写服务地址、AccessKeyId和AccessKeySecret。

> **说明** Linux系统用户目录为*/home/YOURNAME/*，Windows系统用户目录为*C:\Users\YOURNAME*。

```
mns.accountendpoint=http://<yourAccountId>.mns.cn-hangzhou.aliyuncs.com
mns.accesskeyid=<yourAccessKeyId>
mns.accesskeysecret=<yourAccessKeySecret>
```

iv. 运行*QueueSample.java*和*TopicSample.java*文件。

- pom配置

```
<dependency>
    <groupId>com.aliyun.mns</groupId>
    <artifactId>aliyun-sdk-mns</artifactId>
    <version>1.1.0</version>
</dependency>
```

Version 1.0.5

- 更新日期

  2015-12-02

  SDK下载

  sample下载

- 更新内容
  - 修复问题多CloudAccount对象时导致内存泄漏。
  - 依赖的httpasyncclient版本升至4.1。
- 使用帮助
  i. 下载sample并解压*aliyun-sdk-mns-samples-1.0.5.zip*。
  ii. 用Eclipse导入Maven工程，选中*aliyun-sdk-mns-samples*文件夹。
  iii. 在用户目录中创建*.aliyun-mns.properties*文件，并填写服务地址、AccessKeyId和AccessKeySecret。

  > **说明** Linux系统用户目录为*/home/YOURNAME/*，Windows系统用户目录为*C:\Users\YOURNAME*。

  ```
  mns.accountendpoint=http://<yourAccountId>.mns.cn-hangzhou.aliyuncs.com
  mns.accesskeyid=<yourAccessKeyId>
  mns.accesskeysecret=<yourAccessKeySecret>
  ```

  iv. 运行*Sample.java*文件。
- pom配置

```
<dependency>
    <groupId>com.aliyun.mns</groupId>
    <artifactId>aliyun-sdk-mns</artifactId>
    <version>1.0.5</version>
</dependency>
```

Version 1.0.4

- 更新日期

  2015-11-05

  SDK下载

  sample下载

- 更新内容
  - 修复网络异常时极端情况下线程中止。

- 修复关闭空闲连接回收常驻线程。

- 使用帮助
  i. 下载sample并解压*aliyun-sdk-mns-samples-1.0.4.zip*。
  ii. 用Eclipse导入Maven工程，选中*aliyun-sdk-mns-samples*文件夹。
  iii. 在用户目录中创建*.aliyun-mns.properties*文件，并填写服务地址、AccessKeyId和AccessKeySecret。

  > **说明** Linux系统用户目录为*/home/YOURNAME/*，Windows系统用户目录为*C:\Users\YOURNAME*。

  ```
  mns.accountendpoint=http://<yourAccountId>.mns.cn-hangzhou.aliyuncs.com
  mns.accesskeyid=<yourAccessKeyId>
  mns.accesskeysecret=<yourAccessKeySecret>
  ```

  iv. 运行*Sample.java*文件。

- pom配置

```
<dependency>
    <groupId>com.aliyun.mns</groupId>
    <artifactId>aliyun-sdk-mns</artifactId>
    <version>1.0.4</version>
</dependency>
```

Version 1.0.3

- 更新日期

  2015-06-09

  SDK参考

  sample下载

- 更新内容
  - 修复大量close wait的连接导致SDK挂起。
  - 增加sample code。
  - API协议升级："x-mns-version"="2015-06-06"。
  - 支持BatchSendMessage、BatchReceiveMessage、BatchPeekMessage、BatchDeleteMessage。

- 使用帮助
  i. 下载sample并解压*aliyun-sdk-mns-samples-1.0.3.zip*。
  ii. 用Eclipse导入Maven工程，选中*aliyun-sdk-mns-samples*文件夹。
  iii. 在用户目录中创建*.aliyun-mns.properties*文件，并填写服务地址、AccessKeyId和AccessKeySecret。

  > **说明** Linux系统用户目录为*/home/YOURNAME/*，Windows系统用户目录为*C:\Users\YOURNAME*。

```
mns.accountendpoint=http://<yourAccountId>.mns.cn-hangzhou.aliyuncs.com
mns.accesskeyid=<yourAccessKeyId>
mns.accesskeysecret=<yourAccessKeySecret>
```

iv. 运行*Sample.java*文件。

- pom配置

```
<dependency>
    <groupId>com.aliyun.mns</groupId>
    <artifactId>aliyun-sdk-mns</artifactId>
    <version>1.0.3</version>
</dependency>
```

Version 1.0.2

- 更新日期

  2015-03-03

  SDK下载

- 更新内容

  优化XML解析逻辑，提升性能。

- pom配置

```
<dependency>
    <groupId>com.aliyun.mns</groupId>
    <artifactId>aliyun-sdk-mns</artifactId>
    <version>1.0.2</version>
</dependency>
```

Version 1.0.1

- 更新日期

  2014-12-19

  SDK下载

- 更新内容

  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。

- pom配置

```
<dependency>
    <groupId>com.aliyun.mns</groupId>
    <artifactId>aliyun-sdk-mns</artifactId>
    <version>1.0.1</version>
</dependency>
```

Version 1.0.0

- 更新日期

  2014-08-01

  SDK下载

- pom配置

```
<dependency>
    <groupId>com.aliyun.mns</groupId>
    <artifactId>aliyun-sdk-mns</artifactId>
    <version>1.0.0</version>
</dependency>
```

1.2. 队列使用手册

本文介绍如何使用Java SDK中的Sample，完成创建队列、发送消息、接收和删除消息、以及删除队列。

步骤一：准备工作

1. 下载最新版Java SDK，解压到*aliyun-sdk-mns-samples*文件夹。
2. 用Eclipse导入Maven工程，选择*aliyun-sdk-mns-samples*文件夹。
3. 在用户目录中创建*.aliyun-mns.properties*文件，并填写服务地址、AccessKeyId和AccessKeySecret。

   > **说明** Linux系统的用户目录为*/home/YOURNAME/*，Windows系统的用户目录为*C:\Users\YOURNAME*。

   - AccessKeyId、AccessKeySecret
     - 访问阿里云API的密钥对。
     - 如果使用阿里云账号访问，请登录阿里云 AccessKey 管理页面创建和查看。
     - 如果使用RAM用户访问，请登录阿里云访问控制控制台查看。
   - Endpoint
     - 访问的接入地址，登录MNS控制台查看。具体操作，请参见获取接入点。
     - 不同地域的接入地址不同，分为公网和私网域名。

步骤二：创建队列

创建队列的代码段如下。

```
public class CreateQueueDemo {
    public static void main(String[] args) {
        CloudAccount account = new CloudAccount(
                ServiceSettings.getMNSAccessKeyId(),
                ServiceSettings.getMNSAccessKeySecret(),
                ServiceSettings.getMNSAccountEndpoint());
        MNSClient client = account.getMNSClient();
        try
        {
            QueueMeta qMeta = new QueueMeta();
            qMeta.setQueueName("queue-demo");
            qMeta.setPollingWaitSeconds(30);
            CloudQueue cQueue = client.createQueue(qMeta);
            System.out.println("Create queue successfully. URL: " + cQueue.getQueueURL());
        } catch (ClientException ce)
        {
            System.out.println("Something wrong with the network connection between client
and MNS service."
                    + "Please check your network and DNS availability.");
            ce.printStackTrace();
        } catch (ServiceException se)
        {
            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 ma
chine timeclock");
            }
            se.printStackTrace();
        } catch (Exception e)
        {
            System.out.println("Unknown exception happened!");
            e.printStackTrace();
        }
        client.close();
    }
}
```

步骤三：发送消息

创建队列后，即可向队列发送消息。

```
public class ProducerDemo {
    public static void main(String[] args) {
        CloudAccount account = new CloudAccount(
                ServiceSettings.getMNSAccessKeyId(),
                ServiceSettings.getMNSAccessKeySecret(),
                ServiceSettings.getMNSAccountEndpoint());
        MNSClient client = account.getMNSClient();
        try{
            CloudQueue queue = client.getQueueRef("queue-demo");
            for (int i = 0; i < 10; i++)
            {
                Message message = new Message();
                message.setMessageBody("demo_message_body" + i);
                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 availability.");
            ce.printStackTrace();
        } catch (ServiceException se)
        {
            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 ma
chine timeclock");
            }
            se.printStackTrace();
        } catch (Exception e)
        {
            System.out.println("Unknown exception happened!");
            e.printStackTrace();
        }
        client.close();
    }
}
```

步骤四：接收和删除消息

向队列发送消息后，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(
                ServiceSettings.getMNSAccessKeyId(),
                ServiceSettings.getMNSAccessKeySecret(),
                ServiceSettings.getMNSAccountEndpoint());
        MNSClient client = account.getMNSClient();
        try{
            CloudQueue queue = client.getQueueRef("queue-demo");
            for (int i = 0; i < 10; i++)
            {
                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())
;
                    queue.deleteMessage(popMsg.getReceiptHandle());
                    System.out.println("delete message successfully.\n");
                }
            }
        } catch (ClientException ce)
        {
            System.out.println("Something wrong with the network connection between client
and MNS service."
                    + "Please check your network and DNS availability.");
            ce.printStackTrace();
        } catch (ServiceException se)
        {
            if (se.getErrorCode().equals("QueueNotExist"))
            {
                System.out.println("Queue is not exist.Please create queue before use");
            } else if (se.getErrorCode().equals("TimeExpired"))
            {
                System.out.println("The request is time expired. Please check your local ma
chine timeclock");
            }
            se.printStackTrace();
        } catch (Exception e)
        {
            System.out.println("Unknown exception happened!");
            e.printStackTrace();
        }
        client.close();
    }
}
```

步骤五：删除队列

删除测试用的队列。

```
public class DeleteQueueDemo {
    public static void main(String[] args) {
        CloudAccount account = new CloudAccount(
                ServiceSettings.getMNSAccessKeyId(),
                ServiceSettings.getMNSAccessKeySecret(),
                ServiceSettings.getMNSAccountEndpoint());
        MNSClient client = account.getMNSClient();
        try
        {
            CloudQueue queue = client.getQueueRef("queue-demo");
            queue.delete();
            System.out.println("Delete cloud-queue-demo successfully!");
        } catch (ClientException ce)
        {
            System.out.println("Something wrong with the network connection between client
and MNS service."
                    + "Please check your network and DNS availability.");
            ce.printStackTrace();
        } catch (ServiceException se)
        {
            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 ma
chine timeclock");
            }
            se.printStackTrace();
        } catch (Exception e)
        {
            System.out.println("Unknown exception happened!");
            e.printStackTrace();
        }
        client.close();
    }
}
```

1.3. 主题使用手册

本文介绍如何使用Java SDK中的sample代码，完成创建主题、创建订阅、发布消息和接收消息等操作。

步骤一：准备工作

1. 下载最新版Java SDK，解压到*aliyun-sdk-mns-samples*文件夹。
2. 用Eclipse导入Maven工程，选中*aliyun-sdk-mns-samples*文件夹。
3. 在用户目录中创建*.aliyun-mns.properties*文件，并填写服务地址、AccessKey ID和AccessKey Secret。

> **说明** Linux系统的用户目录为*/home/YOURNAME/*，Windows系统的用户目录为*C:\Users\YOURNAME*。

- AccessKeyId、AccessKeySecret
  - 访问阿里云API的密钥对。
  - 如果使用阿里云账号访问，登录阿里云 AccessKey 管理页面创建、查看。
  - 如果使用RAM用户访问，登录阿里云访问控制控制台查看。
- Endpoint
  - 访问的接入地址，登录MNS控制台查看。具体操作，请参见获取接入点。
  - 不同地域的接入地址不同，分为公网以及私网域名。

步骤二：创建主题

创建主题的代码示例如下。详细说明，请参见Topic。

```
public class CreateTopicDemo {
    public static void main(String[] args) {
        CloudAccount account = new CloudAccount("YourAccessId", "YourAccessKey", "MNSEndpoi
nt");
        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();
    }
}
```

步骤三：创建队列

创建队列的代码段如下。

```
public class CreateQueueDemo {
    public static void main(String[] args) {
        CloudAccount account = new CloudAccount(
                ServiceSettings.getMNSAccessKeyId(),
                ServiceSettings.getMNSAccessKeySecret(),
                ServiceSettings.getMNSAccountEndpoint());
        MNSClient client = account.getMNSClient();
        try
        {
            QueueMeta qMeta = new QueueMeta();
            qMeta.setQueueName("queue-demo");
            qMeta.setPollingWaitSeconds(30);
            CloudQueue cQueue = client.createQueue(qMeta);
            System.out.println("Create queue successfully. URL: " + cQueue.getQueueURL());
        } catch (ClientException ce)
        {
            System.out.println("Something wrong with the network connection between client
and MNS service."
                    + "Please check your network and DNS availability.");
            ce.printStackTrace();
        } catch (ServiceException se)
        {
            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 ma
chine timeclock");
            }
            se.printStackTrace();
        } catch (Exception e)
        {
            System.out.println("Unknown exception happened!");
            e.printStackTrace();
        }
        client.close();
    }
}
```

步骤四：创建订阅

对已创建的主题进行订阅，在订阅时需要设置对应的推送Endpoint地址（目前仅支持队列）、错误重试策略、推送消息格式等。

```
public class SubscribeDemo {
    public static void main(String[] args) {
        String region = "";
        String accountId = "";
        String queueName = "TestQueue";
        CloudAccount account = new CloudAccount("YourAccessId", "YourAccessKey", "MNSEndpoi
nt");
        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("QueueEndpoint2");
            subMeta.setEndpoint(String.format("acs:mns:%s:%s:queues/%s", region, accountId,
queueName));
            subMeta.setNotifyContentFormat(SubscriptionMeta.NotifyContentFormat.XML);
            String subUrl = topic.subscribe(subMeta);
            System.out.println("subscription url: " + subUrl);
        } catch (Exception e) {
            e.printStackTrace();
            System.out.println("subscribe/unsubribe error");
        }
        client.close();
    }
}
```

步骤五：发布消息

完成创建主题和订阅，即可向主题发布消息。

```
public class PublishMessageDemo {
    public static void main(String[] args) {
        CloudAccount account = new CloudAccount("YourAccessId", "YourAccessKey", "MNSEndpoi
nt");
        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();
    }
}
```

步骤六：从队列接收和删除消息

消息从主题推送到队列后，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(
                ServiceSettings.getMNSAccessKeyId(),
                ServiceSettings.getMNSAccessKeySecret(),
                ServiceSettings.getMNSAccountEndpoint());
        MNSClient client = account.getMNSClient();
        try{
            CloudQueue queue = client.getQueueRef("queue-demo");
            for (int i = 0; i < 10; i++)
            {
                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())
;
                    queue.deleteMessage(popMsg.getReceiptHandle());
                    System.out.println("delete message successfully.\n");
                }
            }
        } catch (ClientException ce)
        {
            System.out.println("Something wrong with the network connection between client
and MNS service."
                    + "Please check your network and DNS availability.");
            ce.printStackTrace();
        } catch (ServiceException se)
        {
            if (se.getErrorCode().equals("QueueNotExist"))
            {
                System.out.println("Queue is not exist.Please create queue before use");
            } else if (se.getErrorCode().equals("TimeExpired"))
            {
                System.out.println("The request is time expired. Please check your local ma
chine timeclock");
            }
            se.printStackTrace();
        } catch (Exception e)
        {
            System.out.println("Unknown exception happened!");
            e.printStackTrace();
        }
        client.close();
    }
}
```

步骤七：删除主题

删除测试用的主题。

```
public class DeleteTopicDemo {
    public static void main(String[] args) {
        CloudAccount account = new CloudAccount("YourAccessId", "YourAccessKey", "MNSEndpoi
nt");
        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();
    }
}
```

步骤八：删除队列

删除测试用的队列。

```
public class DeleteQueueDemo {
    public static void main(String[] args) {
        CloudAccount account = new CloudAccount(
                ServiceSettings.getMNSAccessKeyId(),
                ServiceSettings.getMNSAccessKeySecret(),
                ServiceSettings.getMNSAccountEndpoint());
        MNSClient client = account.getMNSClient();
        try
        {
            CloudQueue queue = client.getQueueRef("queue-demo");
            queue.delete();
            System.out.println("Delete cloud-queue-demo successfully!");
        } catch (ClientException ce)
        {
            System.out.println("Something wrong with the network connection between client
and MNS service."
                    + "Please check your network and DNS availability.");
            ce.printStackTrace();
        } catch (ServiceException se)
        {
            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 ma
chine timeclock");
            }
            se.printStackTrace();
        } catch (Exception e)
        {
            System.out.println("Unknown exception happened!");
            e.printStackTrace();
        }
        client.close();
    }
}
```

FilterTag使用示例

FilterTag的使用示例如下。

```
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();
        // 1. 创建队列。
        QueueMeta queueMeta = new QueueMeta();
        queueMeta.setQueueName("TestSubForQueue");
        CloudQueue queue = client.createQueue(queueMeta);
        // 2. 创建主题。
        TopicMeta topicMeta = new TopicMeta();
        topicMeta.setTopicName("TestTopic");
        CloudTopic topic = client.createTopic(topicMeta);
        // 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);
        // 4. 发布消息。
        TopicMessage msg = new Base64TopicMessage();
        msg.setMessageBody("hello world");
        msg.setMessageTag("filterTag");
        msg = topic.publishMessage(msg);
        // 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);
    }
}
```

1.4. 并发测试示例代码

本文介绍基于Java SDK提供的队列消息发送以及消费的并发测试用例。

在并发测试中，您可以：

- 指定并发度、运行时间。
- 通过发送总请求数除以运行时间计算得到QPS。

步骤一：初始化

在运行目录创建*perf_test_config.properties*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
```

参数说明如下：

- ThreadNum：发送或消费的线程数，具备强大的高并发扩展性能。
- TotalSeconds：测试用例运行的时间。

步骤二：运行代码

```
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.sep
arator") + "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;
        }
        // 加载文件。
```

```
        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) {
            }
        }
        // 初始化成员参数。
        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(thr
eadNum)));
        System.out.println("ThreadNum: " + threadNum);
        totalSeconds = Integer.parseInt(properties.getProperty("TotalSeconds", String.value
Of(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, clientC
onfiguration);
        client = cloudAccount.getMNSClient();
        CloudQueue queue = client.getQueueRef(queueName);
        queue.delete();
        QueueMeta meta = new QueueMeta();
        meta.setQueueName(queueName);
        client.createQueue(meta);
        // 1. 检查发送消息。
        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. 接收消息。
        threads.clear();
        totalCount.set(0);
        totalSeconds = totalSeconds;
        // 3. 确保队列中的消息能被接收。
        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;
    }
}
```

2.Python SDK

2.1. Python SDK版本说明

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 1.1.6

- 更新日期

  2021-05-13

  SDK下载

- 更新内容
  - 增加发送消息返回值的ReceiptHandle解析。
  - 修复Python 3兼容性问题。

Version 1.1.5

- 更新日期

  2019-04-26

  SDK 下载

- 更新内容

  兼容Python 3版本。

Version 1.1.4

- 更新日期

  2017-03-23

  SDK 下载

- 更新内容
  - 主题模型支持短信推送。
  - 队列和主题支持消息包含中文。
  - mnscmd支持参数指定mnsendpoint、accesskeyid和accesskeysecret。
  - 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

- 支持透传RequestID到MNS端。
- Topic推送支持QueueEndpoint和MailEndpoint。
- 主题消息推送支持JSON格式。
- mnscmd支持config_file指定配置文件。

- 使用帮助
  i. 下载SDK并解压。
  ii. 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
  - Topic推送支持消息过滤。
  - 增加*sample*目录，包含更详细的示例代码。

- 使用帮助
  i. 下载SDK并解压。
  ii. 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
  - 支持HTTPS。
  - 支持Logging。

- 使用帮助
  i. 下载SDK并解压。
  ii. 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
  - 添加对于Topic功能的支持。
  - 添加对于STS Token的支持。

- 使用帮助

i. 下载SDK并解压。

ii. 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

  SDK 下载

- 更新内容
  - 支持SDK安装。
  - 提供**mnscmd**命令。
  - API协议升级："x-mns-version"="2015-06-06"。
  - 支持BatchSendMessage、BatchReceiveMessage、BatchPeekMessage、BatchDeleteMessage。
- 使用帮助

  i. 下载SDK并解压。

  ii. 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

  SDK 下载

- 更新内容
  - 统一队列非字符串属性为int类型。
  - 修正SetQueueAttribute的返回status为204。

Version 1.0.0

- 更新日期

  2014-08-01

  SDK 下载

- 更新内容

  首次发布。

2.2. 队列使用手册

本文介绍如何使用Python SDK中的sample代码，完成创建队列、发送消息、接收和删除消息、以及删除队列操作。

步骤一：准备工作

1. 下载最新版Python SDK，解压后进入*mns_python_sdk*子目录。
2. 打开*sample.cfg*文件，配置AccessKeyId、AccessKeySecret、Endpoint和SecurityToken。
   - AccessKeyId、AccessKeySecret

- 访问阿里云API的密钥对。
- 如果使用阿里云账号访问，请登录阿里云 AccessKey 管理页面创建、查看。
- 如果使用RAM用户访问，请登录阿里云访问控制控制台查看。

- Endpoint
  - 访问的接入地址，登录MNS控制台查看。具体操作，请参见获取接入点。
  - 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，如果使用阿里云账号或者RAM用户访问，不需要配置该项。更多信息，请参见什么是STS。

3. 进入*sample*目录，后续使用的脚本都在该目录。

步骤二：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。更多信息，请参见Queue。

- 运行以下命令：

```
python createqueue.py MyQueue1
```

返回结果如下：

```
Create Queue Succeed! QueueName:MyQueue1
```

- 核心代码：

endpoint、accid、acckey和token从步骤一配置的*sample.cfg*文件中读取。

```
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)
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
```

步骤三：发送消息

运行*sendmessage.py*，发送多条消息到队列中。如果步骤二指定了队列名称，这里同样通过参数指定队列名称。更多信息，请参见QueueMessage。

- 运行以下命令：

```
python sendmessage.py MyQueue1
```

返回结果如下：

```
==========Send Message To Queue==========
QueueName:MyQueue1
MessageCount:3
Send Message Succeed! MessageBody:I am test message 0. MessageID:3EBE662B52BC99BC-1-154BD
99CCA7-200000001
Send Message Succeed! MessageBody:I am test message 1. MessageID:64B92941FC57837F-2-154BD
99CCCE-200000001
Send Message Succeed! MessageBody:I am test message 2. MessageID:3EBE662B52BC99BC-1-154BD
99CCF0-200000002
```

- 核心代码：

  endpoint、accid、acckey和token从步骤一配置的*sample.cfg*文件中读取。

```
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)
msg_count = 3
print "%sSend Message To Queue%s\nQueueName:%s\nMessageCount:%s\n" % (10*"=", 10*"=", que
ue_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.mes
sage_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
```

步骤四：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。如果步骤二指定了队列名称，这里同样通过参数指定队列名称。程序中receive message使用LongPolling方式，指定WaitSeconds为3秒，因此当队列为空时，程序会等待3秒。更多信息，请参见QueueMessage。

- 运行以下命令：

```
python recvdelmessage.py MyQueue1
```

  返回结果如下：

```
==========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从步骤一配置的*sample.cfg*文件中读取。

```
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)
wait_seconds = 3
print "%sReceive And Delete Message From Queue%s\nQueueName:%s\nWaitSeconds:%s\n" % (10*"
=", 10*"=", queue_name, wait_seconds)
while True:
    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
    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
```

步骤五：删除队列

运行*deletequeue.py*删除队列。

- 运行以下命令：

```
python deletequeue.py MyQueue1
```

返回结果如下：

```
Delete Queue Succeed! QueueName:MyQueue1
```

- 核心代码：

  endpoint、accid、acckey和token从步骤一配置的*sample.cfg*文件中读取。

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

2.3. 主题+QueueEndpoint使用手册

本文介绍如何使用Python SDK中的sample代码，完成创建主题、创建队列、创建订阅、发布消息和删除主题等操作。

步骤一：准备工作

1. 下载最新版Python SDK，解压后进入*mns_python_sdk*子目录。
2. 打开*sample.cfg*文件，配置AccessKeyId、AccessKeySecret、Endpoint和SecurityToken。
   - AccessKeyId、AccessKeySecret
     - 访问阿里云API的密钥对。
     - 如果使用阿里云账号访问，请登录阿里云AccessKey管理页面创建、查看。
     - 如果使用RAM用户访问，请登录阿里云访问控制控制台查看。
   - Endpoint
     - 访问的接入地址，登录MNS控制台查看。具体操作，请参见获取接入点
     - 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，使用阿里云账号或者RAM用户访问不需要配置该项。更多信息，请参见什么是STS。
3. 进入*sample*目录，后续使用的脚本都在该目录。

步骤二：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。更多信息，请参见Topic。

- 运行以下命令：

```
python createtopic.py MyTopic1
```

  返回结果如下：

```
Create Topic Succeed! TopicName:MyTopic1
```

- 核心代码：

  endpoint、accid、acckey和token从步骤一配置的文件*sample.cfg*中读取。

```
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)
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
```

步骤三：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。更多信息，请参见Queue。

- 运行以下命令：

```
python createqueue.py MyQueue1
```

返回结果如下：

```
Create Queue Succeed! QueueName:MyQueue1
```

- 核心代码：

endpoint、accid、acckey和token从步骤一配置的文件*sample.cfg*中读取。

```
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)
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
```

步骤四：创建订阅

运行*subscribe.py*创建订阅。文件中四个参数的含义如下所示。更多信息，请参见Subscription。

- 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，使用步骤三创建的队列名称；
- 第三个参数指定订阅的主题名称，如果步骤二中指定了主题名称，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。
- 运行以下命令：

```
python subscribe_queueendpoint.py cn-hangzhou MyQueue1 MyTopic1 MyTopic1-Sub1
```

返回结果如下：

```
Create Subscription Succeed! TopicName:MyTopic1 SubName:MyTopic1-Sub1 Endpoint:acs:mns:cn
-hangzhou:123456789098****:queues/MyQueue1
```

- 核心代码：

  endpoint、accid、acckey和token从步骤一配置的文件*sample.cfg*中读取。

```
region = sys.argv[1]
queue_name = sys.argv[2]
queue_endpoint = TopicHelper.generate_queue_endpoint(region, account_id, queue_name)
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)
sub_meta = SubscriptionMeta(queue_endpoint, notify_content_format = SubscriptionNotifyCon
tentFormat.SIMPLIFIED)
try:
    topic_url = my_sub.subscribe(sub_meta)
    print "Create Subscription Succeed! TopicName:%s SubName:%s Endpoint:%s\n" % (topic_n
ame, 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
```

步骤五：发布消息

运行*publishmessage.py*发布多条消息到主题中。如果步骤二中指定了主题名称，这里同样通过第一个参数指定主题名称。更多信息，请参见TopicMessage。

- 运行以下命令：

```
python publishmessage.py MyTopic1
```

返回结果如下：

```
==========Publish Message To Topic==========
TopicName:MyTopic1
MessageCount:3
Publish Message Succeed. MessageBody:I am test message 0. MessageID:F6EA56633844DBFC-1-15
4BDFB****-200000004
Publish Message Succeed. MessageBody:I am test message 1. MessageID:F6EA56633844DBFC-1-15
4BDFB****-200000005
Publish Message Succeed. MessageBody:I am test message 2. MessageID:F6EA56633844DBFC-1-15
4BDFB****-200000006
```

- 核心代码：

  endpoint、accid、acckey和token从步骤一配置的文件*sample.cfg*中读取。

```
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)
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
```

步骤六：从队列获取和删除消息

步骤五中，发布了多条消息到主题，会将发布的消息推送给步骤四指定的队列中。

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，使用步骤四指定的队列名；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。更多信息，请参见QueueMessage。

```
python recvdelmessage.py MyQueue1 false
```

返回结果如下：

```
==========Receive And Delete Message From Queue==========
QueueName:MyQueue1
WaitSeconds:3
Receive Message Succeed! ReceiptHandle:1-ODU4OTkzNDU5My0xNDczMzkwMjkyLTEtOA== MessageBody:I
am test message 0. MessageID:E56AE055BAA638AC-1-1570CE4****-200000001
Delete Message Succeed!  ReceiptHandle:1-ODU4OTkzNDU5My0xNDczMzkwMjkyLTEtOA==
Receive Message Succeed! ReceiptHandle:1-ODU4OTkzNDU5NC0xNDczMzkwMjkyLTEtOA== MessageBody:I
am test message 1. MessageID:E56AE055BAA638AC-1-1570CE4****-200000002
Delete Message Succeed!  ReceiptHandle:1-ODU4OTkzNDU5NC0xNDczMzkwMjkyLTEtOA==
Receive Message Succeed! ReceiptHandle:1-ODU4OTkzNDU5My0xNDczMzkwMjkyLTItOA== MessageBody:I
am test message 2. MessageID:CDAC88D223C0F9E3-2-1570CE4****-200000001
Delete Message Succeed!  ReceiptHandle:1-ODU4OTkzNDU5My0xNDczMzkwMjkyLTItOA==
Queue is empty!
```

步骤七：删除主题

运行*deletetopic.py*删除主题。如果步骤二中指定了主题名称，这里同样通过第一个参数指定主题名称。

- 运行以下命令：

```
python deletetopic.py MyTopic1
```

返回结果如下：

```
Delete Topic Succeed! TopicName:MyTopic1
```

- 核心代码：

  endpoint、accid、acckey和token从步骤一配置的文件*sample.cfg*中读取。

```
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
```

步骤八：删除队列

运行*deletequeue.py*删除队列。

- 运行以下命令：

```
python deletequeue.py MyQueue1
```

返回结果如下：

```
Delete Queue Succeed! QueueName:MyQueue1
```

- 核心代码：

  endpoint、accid、acckey和token从步骤一配置的文件*sample.cfg*中读取。

```
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
```

3.C# SDK

3.1. C# SDK版本说明

建议下载最新发布的SDK版本，以达到最佳性能。

运行帮助

1. 用VisualStudio导入工程，选择*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、AccessKeySecret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. 将要执行的Sample文件设置为VisualStudio工程的启动项，然后单击执行。

Version 1.3.8

- 更新日期

  2017-08-25

  SDK下载

- 更新内容

  添加RAM STS功能。

Version 1.3.7

- 更新日期

  2017-03-10

  SDK下载

- 更新内容

  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

  支持短信推送。

Version 1.3.5

- 更新日期

  2016-11-1

SDK下载

- 更新内容

  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

  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

  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

  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

  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
  - 支持LoggingEnabled属性。
  - 主题支持队列和邮件推送。

- 支持.net framework 3.5。

Version 1.1.3

- 更新日期

  2016-03-17

  SDK下载

- 更新内容

  为MNSClient添加GetNativeTopic。

Version 1.1.2

- 更新日期

  2016-02-19

  SDK下载

- 更新内容

  为主题添加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

  为主题添加PublishMessage的接口。

Version 1.1.0

- 更新日期

  2016-01-05

  SDK下载

- 更新内容

  添加对于主题功能的支持。

Version 1.0.0

- 更新日期

  2015-09-10

  SDK下载

- 更新内容

  首次发布。

3.2. 队列使用手册

本文介绍如何使用C# SDK创建队列、发送消息、接收和删除消息、以及删除队列。

步骤一：准备工作

1. 下载最新版C# SDK，解压后将工程导入到VisualStudio。
2. 找到工程里四个项目中SDK所在的项目*AliyunSDK_MNS*，单击鼠标右键，选择重新生成，在*bin*目录下生成*Aliyun.MNS.dll*。

   > **说明** 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。

3. 在项目*AliyunSDK_MNS_Sample*里进行配置。
   i. 将项目*AliyunSDK_MNS_Sample*设置为启动项，并将*SyncOperationSample*设置为启动对象。
   ii. 打开文件*SyncOperationSample.cs*，在文件的最上方配置AccessKeyID、AccessKeySecret和Endpoint。
   - AccessKeyId、AccessKeySecret
     - 访问阿里云API的密钥对。
     - 如果使用阿里云账号访问，请登录阿里云AccessKey管理页面创建和查看。
     - 如果使用RAM用户访问，请登录阿里云访问控制控制台查看。
   - Endpoint
     - 访问的接入地址，请登录MNS控制台查看。具体操作，请参见获取接入点。
     - 不同地域的接入地址不同。

步骤二：创建队列

如果之前未创建队列，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，请参见Queue。
var createQueueRequest = new CreateQueueRequest
{
    QueueName = _queueName,
    Attributes =
    {
        // VisibilityTimeout是最重要的属性，默认为30秒。
        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。如果不需要特别指定Queue的属性，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，可以根据具体的错误Code做处理，也可以分别Catch QueueAlreadyExistExceptio
n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);
}
```

步骤三：发送消息

创建队列后可以向队列中发送消息。

```
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。也可以直接使用nativeQueue.SendMessage("MessageBody")。
    var sendMessageResponse = nativeQueue.SendMessage(sendMessageRequest);
    Console.WriteLine("Send message succeed");
}
catch (MNSException me)
{
    // 4.如果创建队列出错，可以根据具体的错误Code做处理，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);
}
```

步骤四：接收和删除消息

现在队列里已经有了一条消息，您可以尝试接收消息。

NextVisibleTime是的消息里一个非常重要的属性。更多信息，请参见QueueMessage。

```
try
{
    // 1.直接调用receiveMessage函数。
    // receiveMessage函数接受waitSeconds参数，无特殊情况建议设置为30。
    // 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。更多信息，请参见QueueMessage。
    _receiptHandle = message.ReceiptHandle;
}
catch (MNSException me)
{
    // 3.和CreateQueue和SendMessage一样，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);
}
// 这里是您自己处理消息的逻辑。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.消息处理完成，可以从队列里删除这条消息。
try
{
    // 5.直接调用deleteMessage。
    var deleteMessageResponse = nativeQueue.DeleteMessage(_receiptHandle);
}
catch (MNSException me)
{
    // 6.这里CatchException并做异常处理。
    // 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。
    // 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);
}
```

步骤五：删除队列

删除测试队列。

```
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);
}
```

3.3. 主题使用手册

本文介绍如何使用C# SDK中的sample代码，完成创建主题、创建队列、创建订阅和发布消息等操作。

步骤一：准备工作

1. 下载最新版C# SDK，解压后将工程导入到VisualStudio。
2. 找到工程里四个项目中SDK所在的项目*AliyunSDK_MNS*，右击项目名，选择重新生成，在*bin*目录下生成*Aliyun.MNS.dll*。

   > **说明** 其他几个项目里都需要引用*Aliyun.MNS.dll*，请配置好其他几个项目的引用。

3. 在项目*AliyunSDK_MNS_Sample*里进行配置。
   i. 将项目*AliyunSDK_MNS_Sample*设置为启动项，将*SyncTopicOperations.cs*设置为启动对象。
   ii. 打开*SyncTopicOperations.cs*文件，配置以下内容：
   - AccessKey ID、AccessKey Secret
     - 访问阿里云API的密钥对。
     - 如果使用阿里云账号访问，请登录阿里云AccessKey管理页面创建和查看。
     - 如果使用RAM用户访问，请登录阿里云访问控制控制台查看。
   - Endpoint
     - 访问的接入地址，请登录MNS控制台查看。具体操作，请参见获取接入点。
     - 不同地域的接入地址不同。

步骤二：创建主题

如果之前未创建主题，那么首先需要创建主题。默认创建的主题名称是TestCSharpTopic，也可以修改代码指定主题名称。

```
// 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，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;
}
```

步骤三：创建队列

如果之前未创建队列，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，请参见Queue。
var createQueueRequest = new CreateQueueRequest
{
    QueueName = _queueName,
    Attributes =
    {
        // VisibilityTimeout是最重要的属性，默认为30秒。
        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。如果不需要特别指定Queue的属性，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，可以根据具体的错误Code做处理，也可以分别Catch QueueAlreadyExistExceptio
n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);
}
```

步骤四：创建订阅

对已创建的主题进行订阅。

```
string region = "";
string accountId = "";
string queueName = "TestQueue";
try
{
    // 1.生成SubscriptionAttributes，第二个参数是Subscription的Endpoint。
    SubscribeResponse res = topic.Subscribe(_subscriptionName, "acs:mns:" + region +":" + a
ccountId +":queues/" + queueName);
    // 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);
}
```

步骤五：发布消息

发布消息到主题中。

```
try
{
    var response = topic.PublishMessage("message here </asdas\">");
    // 1.发布成功。
    Console.WriteLine("PublishMessage succeed! " + response.MessageId);
}
catch (MNSException me)
{
    // 2.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);
}
```

步骤六：从队列接收和删除消息

消息从主题推送到队列后，从队列中取出并删除该条消息。

```
try
{
    // 1.直接调用receiveMessage函数。
    // receiveMessage函数接受waitSeconds参数，无特殊情况建议设置为30。
    // 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。更多信息，请参见QueueMessage。
    _receiptHandle = message.ReceiptHandle;
}
catch (MNSException me)
{
    // 3.和CreateQueue和SendMessage一样，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);
}
// 这里是您自己处理消息的逻辑。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.消息处理完成，可以从队列里删除这条消息。
try
{
    // 5.直接调用deleteMessage。
    var deleteMessageResponse = nativeQueue.DeleteMessage(_receiptHandle);
}
catch (MNSException me)
{
    // 6.这里CatchException并做异常处理。
    // 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。
    // 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);
}
```

步骤七：删除主题

删除测试主题。

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

步骤八：删除队列

删除测试队列。

```
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);
}
```

4.PHP SDK

4.1. PHP SDK版本说明

建议下载最新发布的SDK版本以获得最佳性能。

注意事项

使用PHP 5.5 及以上版本。

Version 1.3.6

- 更新日期

  2019-04-26

  SDK下载

- 更新内容

  支持Composer安装。

- 使用帮助

  i. 安装PHP。更多信息，请参见安装PHP。

  ii. 安装Composer。更多信息，请参见安装Composer。

Version 1.3.5

- 更新日期

  2017-06-06

  SDK下载

- 更新内容

  在SendMessage的时候对于Priority增加判断。

Version 1.3.4

- 更新日期

  2017-04-13

  SDK下载

- 更新内容

  支持空变量模板的短信推送。

Version 1.3.3

- 更新日期

  2016-12-15

  SDK下载

- 更新内容

  支持短信推送，可以查看TopicTest.php里面的对应代码。

Version 1.3.2

- 更新日期

  2016-11-03

  SDK下载

- 更新内容
  - 增加超时时间设置；
  - GetSubscriptionAttr的response里修复对于Endpoint的解析。

Version 1.3.1

- 更新日期

  2016-05-19

  SDK下载

- 更新内容

  Samples改进，主要是修改了读取主题消息的HttpServerSample。

Version 1.3.0

- 更新日期

  2016-02-25

  SDK下载

- 更新内容

  Subscription增加Queue和Mail推送的支持。

- 运行帮助

  i. 下载并解压SDK到任意目录。

  ii. 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

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

- 运行帮助

  i. 下载并解压SDK到任意目录。

  ii. 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

  在PHP SDK里Queue的所有MessageBody都是被默认做了Base64处理。这个版本的队列提供了禁用Base64的选项，需要在getQueueRef 的时候传入参数$base64 = FALSE。

- 运行帮助

  i. 下载并解压SDK到任意目录。

  ii. 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

  PublishMessage接口不再对MessageBody做Base64编码。

- 运行帮助

  i. 下载并解压SDK到任意目录。

  ii. 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
  - 添加对于主题功能的支持。
  - 添加对于STS Token 的支持。

- 运行帮助

  i. 下载并解压SDK到任意目录。

  ii. 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

- 运行帮助

  i. 下载并解压SDK到任意目录。

  ii. 在使用SDK的文件里加上一行：require_once("/path_to_sdk/mns-autoloader.php")。

4.2. 队列使用手册

本文介绍如何使用PHP SDK创建队列、发送消息、接收和删除消息、以及删除队列。

步骤一：准备工作

1. 下载文件CreateQueueAndSendMessage.php。
2. 打开*CreateQueueAndSendMessage.php*文件，在文件的最下方配置AccessKeyId、AccessKeySecret和Endpoint。
   - AccessKeyId、AccessKeySecret
     - 访问阿里云API的密钥对。
     - 如果使用阿里云账号访问，请登录阿里云AccessKey管理页面创建、查看。
     - 如果使用RAM用户访问，请登录阿里云访问控制控制台查看。
   - Endpoint
     - 访问的接入地址，请登录MNS控制台查看。具体操作，请参见获取接入点。
     - 不同地域的接入地址不同。
3. CreateQueueAndSendMessage的代码最上方有一些设置，使用SDK的时候进行同样的设置。

```
//require SDK里自带的一个autoload文件。
require __DIR__ . '/vendor/autoload.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;
```

步骤二：创建队列

如果之前未创建队列，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。
// 对于queue的属性，请参见Queue。
$request = new CreateQueueRequest($queueName);
try
{
        $res = $this->client->createQueue($request);
        // 3.创建队列成功。
        echo "QueueCreated! \n";
}
catch (MnsException $e)
{
        // 4.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;
         return;
}
```

步骤三：发送消息

创建队列后，发送消息到队列。

```
// 1.首先获取Queue的实例。
// PHP SDK默认会对发送的消息做Base64 Encode，对接收到的消息做Base64 Decode。
// 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。
// SendMessageRequest对象本身也包含了DelaySeconds和Priority属性可以设置。
// 对于Message的属性，请参见QueueMessage。
$bodyMD5 = md5(base64_encode($messageBody));
$request = new SendMessageRequest($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;
        return;
}
```

步骤四：接收和删除消息

队列里已经有了一条消息，现在可以尝试接收消息。

NextVisibleTime是的消息的一个重要属性。更多信息，请参见QueueMessage。

```
$receiptHandle = NULL;
try
{
        // 1.直接调用receiveMessage函数。
        // receiveMessage函数接受waitSeconds参数，无特殊情况建议设置为30。
        // waitSeconds非0表示这次receiveMessage是一次http long polling，如果queue内没有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 = $queue->receiveMessage(30);
        echo "ReceiveMessage Succeed! \n";
        if (strtoupper($bodyMD5) == $res->getMessageBodyMD5())
        {
             echo "You got the message sent by yourself! \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。更多信息，请参见QueueMessage。
        $receiptHandle = $res->getReceiptHandle();
}
catch (MnsException $e)
{
        // 3.和CreateQueue和SendMessage一样，ReceiveMessage也有可能出错，所以加上CatchException并做对应的处理。
        echo "ReceiveMessage Failed: " . $e;
        return;
}
// 这里是您处理消息的逻辑。Sample里就直接略过这一步。
// 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.消息已经处理完成，从队列里删除这条消息。
try
{
        // 5.直接调用deleteMessage。
        $res = $queue->deleteMessage($receiptHandle);
        echo "DeleteMessage Succeed! \n";
}
catch (MnsException $e)
{
        // 6.这里CatchException并做异常处理。
        // 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。
        // 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;
        return;
}
```

步骤五：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;
}
```

4.3. 主题使用手册

本文介绍如何使用PHP SDK创建主题、创建队列、创建订阅和发布消息等操作。

步骤一：准备工作

1. 下载文件CreateTopicAndPublishMessage.php。
2. 打开*CreateTopicAndPublishMessage.php*文件，在文件中最下方配置以下内容：
   - AccessKey ID、AccessKey Secret
     - 访问阿里云API的密钥对。
     - 如果使用阿里云账号访问，请登录阿里云AccessKey管理页面创建和查看。
     - 如果使用RAM用户访问，请登录阿里云访问控制控制台查看。
   - Endpoint
     - 访问的接入地址，请登录MNS控制台查看。具体操作，请参见获取接入点。
     - 不同地域的接入地址不同。
3. 修改SDK设置。

   CreateTopicAndPublishMessage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。
require __DIR__ . '/vendor/autoload.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;
```

步骤二：创建主题

如果之前未创建过主题，那么首先需要创建主题。默认创建的主题名称是CreateTopicAndPublishMessageExample，也可以修改代码指定主题名称。

```
// 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;
```

步骤三：创建队列

如果之前未创建队列，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。
// 对于queue的属性，请参见Queue。
$request = new CreateQueueRequest($queueName);
try
{
        $res = $this->client->createQueue($request);
        // 3.创建队列成功。
        echo "QueueCreated! \n";
}
catch (MnsException $e)
{
        // 4.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;
         return;
}
```

步骤四：创建订阅

对已创建的主题进行订阅。

```
$regionId = "";
$accountId = "";
$queueName = "TestQueue";
$subscriptionName = "QueueEndpoint";
$attributes = new SubscriptionAttributes($subscriptionName, 'acs:mns:' . $regionId . ':' .
$accountId . ':queues/' . $queueName);
try
{
        $topic->subscribe($attributes);
        // 订阅成功。
        echo "Subscribed! \n";
}
catch (MnsException $e)
{
        // 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;
}
```

步骤五：发布消息

发布消息到主题中。

```
$messageBody = "test";
// 1.生成PublishMessageRequest。如果是推送到邮箱，还需要设置MessageAttributes，可以参照Tests/Topi
cTest.php里面的testPublishMailMessage。
$request = new PublishMessageRequest($messageBody);
try
{
        $res = $topic->publishMessage($request);
        // 2.发布消息成功。
        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;
}
```

步骤六：从队列接收和删除消息

消息从主题推送到队列后，从队列中取出并删除该条消息。

```
$receiptHandle = NULL;
try
{
        // 1.直接调用receiveMessage函数。
        // receiveMessage函数接受waitSeconds参数，无特殊情况建议设置为30。
        // waitSeconds非0表示这次receiveMessage是一次http long polling，如果queue内没有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 = $queue->receiveMessage(30);
        echo "ReceiveMessage Succeed! \n";
        if (strtoupper($bodyMD5) == $res->getMessageBodyMD5())
        {
             echo "You got the message sent by yourself! \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。更多信息，请参见QueueMessage。
        $receiptHandle = $res->getReceiptHandle();
}
catch (MnsException $e)
{
        // 3.和CreateQueue和SendMessage一样，ReceiveMessage也有可能出错，所以加上CatchException并做对应的处理。
        echo "ReceiveMessage Failed: " . $e;
        return;
}
// 这里是您处理消息的逻辑。Sample里就直接略过这一步。
// 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.消息已经处理完成，从队列里删除这条消息。
try
{
        // 5.直接调用deleteMessage。
        $res = $queue->deleteMessage($receiptHandle);
        echo "DeleteMessage Succeed! \n";
}
catch (MnsException $e)
{
        // 6.这里CatchException并做异常处理。
        // 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。
        // 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;
        return;
}
```

步骤七：删除主题

删除测试用的主题。

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

步骤八：删除队列

删除测试用的队列。

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

5.JMS SDK

本文介绍JMS SDK的获取方式、使用限制和示例代码。

获取方式

JMS库的获取方式如下：

- 在Maven项目的*pom.xml*文件中添加依赖

```
<dependency>
    <groupId>com.aliyun.mns</groupId>
    <artifactId>java-messaging-lib</artifactId>
    <version>0.2.0</version>
</dependency>
```

- 下载依赖的JAR文件

使用限制

- 只支持创建队列和收发消息操作。
- 只支持AUTO_ACKNOWLEDGE和MANUAL_ACKNOWLEDGE两种ACK模式：
  - AUTO_ACKNOWLEDGE：
    - MessageListener执行完成且无异常抛出时会自动ACK。
    - MessageListener抛出异常就不会ACK，在VisibilityTimeout后消息会重投。
  - MANUAL_ACKNOWLEDGE：您需要在代码中调用 `acknowledge()` 对消息进行ACK。

示例代码

- 创建队列

```
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String endpoint = "<yourEndpoint>";
String queueName = "<yourQueueName>";
MNSConnectionFactory factory = MNSConnectionFactory.builder()
    .withAccessKeyId(accessKeyId)
    .withAccessKeySecret(accessKeySecret)
    .withEndpoint(endpoint)
    .build();
MNSQueueConnection connection = factory.createQueueConnection();
MNSClientWrapper mnsClientWrapper = connection.getMNSClientWrapper();
mnsClientWrapper.createQueue(queueName);
```

- 发送普通消息

```
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String endpoint = "<yourEndpoint>";
String queueName = "<yourQueueName>";
MNSConnectionFactory factory = MNSConnectionFactory.builder()
    .withAccessKeyId(accessKeyId)
    .withAccessKeySecret(accessKeySecret)
    .withEndpoint(endpoint)
    .build();
try {
    MNSQueueConnection connection = factory.createQueueConnection();
    QueueSession session = connection.createQueueSession(false, Session.AUTO_ACKNOWLEDGE)
;
    Queue queue = session.createQueue(queueName);
    MessageProducer producer = session.createProducer(queue);
    TextMessage textMessage = session.createTextMessage("Hello JMS! ");
    textMessage.setDoubleProperty("TestFloat", 0.127);
    producer.send(textMessage);
    System.out.println(textMessage.getJMSMessageID());
} catch (JMSException e) {
    e.printStackTrace();
}
```

- 发送延迟消息

```
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String endpoint = "<yourEndpoint>";
String queueName = "<yourQueueName>";
MNSConnectionFactory factory = MNSConnectionFactory.builder()
    .withAccessKeyId(accessKeyId)
    .withAccessKeySecret(accessKeySecret)
    .withEndpoint(endpoint)
    .build();
try {
    MNSQueueConnection connection = factory.createQueueConnection();
    QueueSession session = connection.createQueueSession(false, Session.AUTO_ACKNOWLEDGE)
;
    Queue queue = session.createQueue(queueName);
    MessageProducer producer = session.createProducer(queue);
    TextMessage textMessage = session.createTextMessage("Hello JMS! ");
    textMessage.setDoubleProperty("TestFloat", 0.127);
    //设置DelaySeconds。
    MNSMessageHelper.setDelaySeconds(textMessage, 10);
    producer.send(textMessage);
    System.out.println(textMessage.getJMSMessageID());
} catch (JMSException e) {
    e.printStackTrace();
}
```

- AUTO_ACKNOWLEDGE模式消费

```
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String endpoint = "<yourEndpoint>";
String queueName = "<yourQueueName>";
MNSConnectionFactory factory = MNSConnectionFactory.builder()
    .withAccessKeyId(accessKeyId)
    .withAccessKeySecret(accessKeySecret)
    .withEndpoint(endpoint)
    .build();
try {
    MNSQueueConnection connection = factory.createQueueConnection();
    QueueSession session = connection.createQueueSession(false, MNSQueueSession.AUTO_ACKN
OWLEDGE);
    Queue queue = session.createQueue(queueName);
    MessageConsumer consumer = session.createConsumer(queue);
    MessageListener listener = new MessageListener() {
        @Override
        public void onMessage(Message message) {
            try {
                if (message instanceof TextMessage) {
                    MNSTextMessage textMessage = (MNSTextMessage) message;
                    System.out.println(new Date() + " Receive in listener: " + textMessag
e.getText());
                } else if (message instanceof BytesMessage) {
                    MNSBytesMessage bytesMessage = (MNSBytesMessage) message;
                    char readChar = bytesMessage.readChar();
                    System.out.println("Read Char: " + readChar);
                    int readInt = bytesMessage.readInt();
                    System.out.println("Read Int: " + readInt);
                } else if (message instanceof ObjectMessage) {
                    MNSObjectMessage objectMessage = (MNSObjectMessage) message;
                    TestSerializable object = (TestSerializable) objectMessage.getObject(
);
                    System.out.println(object.getName());
                }
            } catch (JMSException e) {
                e.printStackTrace();
            }
        }
    };
    consumer.setMessageListener(listener);
    connection.start();
} catch (Exception e) {
    e.printStackTrace();
}
```

- MANUAL_ACKNOWLEDGE模式消费消息

```
String accessKeyId = "<yourAccessKeyId>";
String accessKeySecret = "<yourAccessKeySecret>";
String endpoint = "<yourEndpoint>";
String queueName = "<yourQueueName>";
MNSConnectionFactory factory = MNSConnectionFactory.builder()
    .withAccessKeyId(accessKeyId)
    .withAccessKeySecret(accessKeySecret)
    .withEndpoint(endpoint)
    .build();
try {
    MNSQueueConnection connection = factory.createQueueConnection();
    QueueSession session = connection.createQueueSession(false, MNSQueueSession.MANUAL_AC
KNOWLEDGE);
    Queue queue = session.createQueue(queueName);
    MessageConsumer consumer = session.createConsumer(queue);
    MessageListener listener = new MessageListener() {
        @Override
        public void onMessage(Message message) {
            try {
                if (message instanceof TextMessage) {
                    MNSTextMessage textMessage = (MNSTextMessage) message;
                    System.out.println(new Date() + " Receive in listener: " + textMessag
e.getText());
                } else if (message instanceof BytesMessage) {
                    MNSBytesMessage bytesMessage = (MNSBytesMessage) message;
                    char readChar = bytesMessage.readChar();
                    System.out.println("Read Char: " + readChar);
                    int readInt = bytesMessage.readInt();
                    System.out.println("Read Int: " + readInt);
                } else if (message instanceof ObjectMessage) {
                    MNSObjectMessage objectMessage = (MNSObjectMessage) message;
                    TestSerializable object = (TestSerializable) objectMessage.getObject(
);
                    System.out.println(object.getName());
                }
                //消费成功后需手动ACK。
                message.acknowledge();
            } catch (JMSException e) {
                e.printStackTrace();
            }
        }
    };
    consumer.setMessageListener(listener);
    connection.start();
} catch (Exception e) {
    e.printStackTrace();
}
```

6.C++ SDK

建议下载最新发布的SDK版本以获得最佳性能。

Version 1.3.6

- 更新日期

  2019-12-16

  SDK 下载

- 更新内容

  支持Mac平台。

- 使用需知

  i. Linux系统建议使用g++4.1.2及以上版本，Windows系统需要安装VS2015。

  ii. 安装scons。

- 运行帮助

  i. 下载并解压SDK。

  ii. 在SDK目录执行**scons**命令。

  iii. lib会被自动编译到SDK的*lib*目录。

- 运行Sample

  i. 在sample目录修改*aliyun-mns.properties*，填上正确的Endpoint、AccessKey ID、AccessKey Secret。

  ii. 在sample目录下执行 `./mns_sample` 。

Version 1.3.5

- 更新日期

  2017-04-11

  SDK下载

- 更新内容

  i. 支持MessageTag功能。

  ii. 支持短信推送功能。

- 使用需知

  i. Linux系统建议使用g++ 4.1.2及以上版本，Windows系统需要安装VS2015。

  ii. 安装scons。

- 运行帮助

  i. 下载并解压SDK。

  ii. 在SDK目录执行**scons**命令。

  iii. lib会被自动编译到SDK的*lib*目录。

- 运行Sample

  i. 在sample目录修改*aliyun-mns.properties*，填上正确的Endpoint、AccessKey ID、AccessKey Secret。

ii. 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

  更新了签名时使用的Base64Encode方法，避免极端情况下的出错。

- 使用需知

  i. Linux系统建议使用g++ 4.1.2及以上版本，Windows系统需要安装VS2015。

  ii. 安装scons。

- 运行帮助

  i. 下载并解压SDK。

  ii. 在SDK目录执行**scons**命令。

  iii. lib会被自动编译到SDK的*lib*目录。

- 运行Sample

  i. 在sample目录修改*aliyun-mns.properties*，填上正确的Endpoint、AccessKey ID、AccessKey Secret。

  ii. 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

  对于InvalidEndpoint做了容错处理。

- 使用需知

  i. Linux系统建议使用g++ 4.1.2及以上版本，Windows系统需要安装VS2015。

  ii. 安装scons。

- 运行帮助

  i. 下载并解压SDK。

  ii. 在SDK目录执行**scons**命令。

  iii. lib会被自动编译到SDK的*lib*目录。

- 运行Sample

  i. 在sample目录修改*aliyun-mns.properties*，填上正确的Endpoint、AccessKey ID、AccessKey Secret。

  ii. 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

  在ServiceException增加GetMessage函数。

- 使用需知

  i. Linux系统建议使用g++ 4.1.2及以上版本，Windows系统需要安装VS2015。

  ii. 安装scons。

- 运行帮助

  i. 下载并解压SDK。

  ii. 在SDK目录执行**scons**命令。

  iii. lib会被自动编译到SDK的*lib*目录。

- 运行Sample

  i. 在sample目录修改*aliyun-mns.properties*，填上正确的Endpoint、AccessKey ID、AccessKey Secret。

  ii. 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

  为MnsClient增加超时设置。

- 使用需知

  i. Linux系统建议使用g++ 4.1.2及以上版本，Windows系统需要安装VS2015。

  ii. 安装scons。

- 运行帮助

  i. 下载并解压SDK。

  ii. 在SDK目录执行**scons**命令。

  iii. lib会被自动编译到SDK的*lib*目录。

- 运行Sample

  i. 在sample目录修改*aliyun-mns.properties*，填上正确的Endpoint、AccessKey ID、AccessKey Secret。

  ii. 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
  i. Subscription增加Queue和Mail推送的支持。
  ii. 编译方式修改为scons，兼容Windows。
  iii. 部分兼容性改动。
- 使用需知
  i. Linux系统建议使用g++ 4.1.2及以上版本，Windows系统需要安装VS2015。
  ii. 安装scons。
- 运行帮助
  i. 下载并解压SDK。
  ii. 在SDK目录执行**scons**命令。
  iii. lib会被自动编译到SDK的*lib*目录。
- 运行Sample
  i. 在sample目录修改*aliyun-mns.properties*，填上正确的Endpoint、AccessKey ID、AccessKey Secret。
  ii. 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
  i. 添加对于主题功能的支持。
  ii. 添加对于STS Token的支持。
- 前置需求
  i. 使用g++ 4.1.2及以上版本。
- 运行帮助
  i. 下载并解压SDK。
  ii. 在SDK目录执行 **./configure && make && sudo make install**。
  iii. 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
  i. 在sample目录修改*aliyun-mns.properties*，填上正确的Endpoint、AccessKey ID、AccessKey Secret。
  ii. 在sample目录下执行**make**。
  iii. 运行 `./mns_sample` 。

Version 1.0.0

- 更新日期

2015-12-14

SDK下载

- 前置需求

  使用g++ 4.1.2及以上版本。

- 运行帮助

  i. 下载并解压SDK。

  ii. 在SDK目录执行 `./configure && make && sudo make install` 。

  iii. 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、AccessKey ID、AccessKey Secret。
2. 在sample目录下执行**make**。
3. 运行 `./mns_sample` 。

7.Go SDK

本文介绍Go SDK的版本说明、使用说明和示例代码。

Version 1.0.2

- 更新日期

  2021-03-05

  SDK 下载

- 更新内容

  支持OpenService接口。

Version 1.0.1

- 更新日期

  2021-01-15

  SDK 下载

- 更新内容
  - 支持设置超时。
  - 在返回中添加Request ID。

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。

使用说明

1. 下载最新版Go SDK，解压后进入*aliyun-mns-go-sdk-master*目录。
2. 修改*app.conf.example*文件，配置url、access_key_id和access_key_secret。
   - url

     访问的接入地址，登录MNS控制台查看。具体操作，请参见获取接入点。
   - access_key_id和access_key_secret

     阿里云身份验证，在RAM控制台创建。获取方式，请参见获取AccessKey。

3. 进入*example*目录，后续使用的脚本都在该目录。

示例代码

本示例包括创建队列、发送消息、接收和删除消息、以及删除队列等操作。

```
package main
import (
    "encoding/json"
    "fmt"
    "io/ioutil"
    "log"
    "net/http"
    _ "net/http/pprof"
    "time"
    "github.com/gogap/logs"
    "github.com/aliyun/aliyun-mns-go-sdk"
)
type appConf struct {
    Url             string `json:"url"`
    AccessKeyId     string `json:"access_key_id"`
    AccessKeySecret string `json:"access_key_secret"`
}
func main() {
    go func() {
        log.Println(http.ListenAndServe("localhost:8080", nil))
    }()
    conf := appConf{}
    if bFile, e := ioutil.ReadFile("app.conf.example"); e != nil {
        panic(e)
    } else {
        if e := json.Unmarshal(bFile, &conf); e != nil {
            panic(e)
        }
    }
    client := ali_mns.NewAliMNSClient(conf.Url,
        conf.AccessKeyId,
        conf.AccessKeySecret)
    queueManager := ali_mns.NewMNSQueueManager(client)
    // 创建队列，队列名称为test。
    err := queueManager.CreateQueue("test", 0, 65536, 345600, 30, 0, 3)
    time.Sleep(time.Duration(2) * time.Second)
    if err != nil && !ali_mns.ERR_MNS_QUEUE_ALREADY_EXIST_AND_HAVE_SAME_ATTR.IsEqual(err) {
        fmt.Println(err)
        return
    }
    msg := ali_mns.MessageSendRequest{
        MessageBody:  "hello <\"aliyun-mns-go-sdk\">",
        DelaySeconds: 0,
        Priority:     8}
    queue := ali_mns.NewMNSQueue("test", client)
    // 发送消息。
    for i := 1; i < 10000; i++ {
        ret, err := queue.SendMessage(msg)
        go func() {
```

```
            fmt.Println(queue.QPSMonitor().QPS())
        }()
        if err != nil {
            fmt.Println(err)
        } else {
            logs.Pretty("response:", ret)
        }
        endChan := make(chan int)
        respChan := make(chan ali_mns.MessageReceiveResponse)
        errChan := make(chan error)
        go func() {
            select {
            case resp := <-respChan:
                {
                    logs.Pretty("response:", resp)
                    logs.Debug("change the visibility: ", resp.ReceiptHandle)
                    if ret, e := queue.ChangeMessageVisibility(resp.ReceiptHandle, 5); e !=
nil {
                        fmt.Println(e)
                    } else {
                        logs.Pretty("visibility changed", ret)
                        logs.Debug("delete it now: ", ret.ReceiptHandle)
                        if e := queue.DeleteMessage(ret.ReceiptHandle); e != nil {
                            fmt.Println(e)
                        }
                        endChan <- 1
                    }
                }
            case err := <-errChan:
                {
                    fmt.Println(err)
                    endChan <- 1
                }
            }
        }()
        //接收消息。
        queue.ReceiveMessage(respChan, errChan, 30)
        <-endChan
    }
}
```

本示例包括创建队列、创建主题、订阅主题和发布消息等操作。

```
package main
import (
    "encoding/json"
    "fmt"
    "io/ioutil"
    "time"
    "github.com/gogap/logs"
    "github.com/aliyun/aliyun-mns-go-sdk"
)
type appConf struct {
```

```
    Url             string  json:"url"
    AccessKeyId     string `json:"access_key_id"`
    AccessKeySecret string `json:"access_key_secret"`
}
func main() {
    conf := appConf{}
    if bFile, e := ioutil.ReadFile("app.conf.example"); e != nil {
        panic(e)
    } else {
        if e := json.Unmarshal(bFile, &conf); e != nil {
            panic(e)
        }
    }
    client := ali_mns.NewAliMNSClient(conf.Url,
        conf.AccessKeyId,
        conf.AccessKeySecret)
    // 创建队列，队列名称为testQueue。
    queueManager := ali_mns.NewMNSQueueManager(client)
    err := queueManager.CreateSimpleQueue("testQueue")
    if err != nil && !ali_mns.ERR_MNS_QUEUE_ALREADY_EXIST_AND_HAVE_SAME_ATTR.IsEqual(err) {
        fmt.Println(err)
        return
    }
    // 创建主题，主题名称为testTopic。
    topicManager := ali_mns.NewMNSTopicManager(client)
    // topicManager.DeleteTopic("testTopic")
    err = topicManager.CreateSimpleTopic("testTopic")
    if err != nil && !ali_mns.ERR_MNS_TOPIC_ALREADY_EXIST_AND_HAVE_SAME_ATTR.IsEqual(err) {
        fmt.Println(err)
        return
    }
    // 订阅主题，本示例中的endpoint设置为队列名称。
    topic := ali_mns.NewMNSTopic("testTopic", client)
    sub := ali_mns.MessageSubsribeRequest{
        Endpoint: topic.GenerateQueueEndpoint("testQueue"),
        NotifyContentFormat: ali_mns.SIMPLIFIED,
    }
    // topic.Unsubscribe("SubscriptionNameA")
    err = topic.Subscribe("SubscriptionNameA", sub)
    if err != nil && !ali_mns.ERR_MNS_SUBSCRIPTION_ALREADY_EXIST_AND_HAVE_SAME_ATTR.IsEqual
(err) {
        fmt.Println(err)
        return
    }
    time.Sleep(time.Duration(2) * time.Second)
    // 发布消息。
    msg := ali_mns.MessagePublishRequest{
        MessageBody: "hello topic <\"aliyun-mns-go-sdk\">",
    }
    _, err = topic.PublishMessage(msg)
    if err != nil {
        fmt.Println(err)
        return
    }
    // 从队列中接收消息。
```

```
    // 从队列中接收消息
    queue := ali_mns.NewMNSQueue("testQueue", client)
    endChan := make(chan int)
    respChan := make(chan ali_mns.MessageReceiveResponse)
    errChan := make(chan error)
    go func() {
        select {
        case resp := <-respChan:
            {
                logs.Pretty("response:", resp)
                fmt.Println("change the visibility: ", resp.ReceiptHandle)
                if ret, e := queue.ChangeMessageVisibility(resp.ReceiptHandle, 5); e != nil
{
                    fmt.Println(e)
                } else {
                    logs.Pretty("visibility changed", ret)
                    fmt.Println("delete it now: ", ret.ReceiptHandle)
                    if e := queue.DeleteMessage(ret.ReceiptHandle); e != nil {
                        fmt.Println(e)
                    }
                    endChan <- 1
                }
            }
        case err := <-errChan:
            {
                fmt.Println(err)
                endChan <- 1
            }
        }
    }()
    queue.ReceiveMessage(respChan, errChan, 30)
    <-endChan
}
```

队列模型

主题模型

8.Android SDK

8.1. 通知

不再提供Android SDK，如果您的客户端是Android操作系统，推荐您使用微消息队列MQTT。

阿里云消息队列（MQ）推出微消息队列MQTT，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。
- 如有其它疑问，您可以提交工单来反馈。

9.Objective-C iOS SDK

9.1. 通知

不再提供Objective-C iOS SDK，如果您的客户端是iOS操作系统，推荐您使用微消息队列MQTT。

阿里云消息队列（MQ）推出微消息队列MQTT，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。
- 如有其它疑问，您可以提交工单来反馈。

10.FAQ

10.1. SDK下载

提供了以下语言版本SDK供您使用，一般建议下载最新发布的版本以获得最佳性能和稳定性。

- Java SDK
- Python SDK
- C# SDK
- PHP SDK
- C++ SDK
- Go SDK

如果问题未能解决，请提交工单联系售后技术支持，请参见提交工单。