ALIBABA CLOUD

阿里云

消息队列 MQ

常见问题

文档版本：20220329

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.常见问题

1.1. 快速开始

1.1.1. 新创建的Group ID从哪里开始消费？

- 如果这个Group ID是第一次启动，则会忽略启动之前发送的消息，也就是忽略历史消息，从启动之后发送的消息开始消费。
- 如果这个Group ID是第二次启动，那么从上次消费的位置开始消费。
- 如果想从特定位置开始消费，可以通过

  消息队列RocketMQ版

  控制台的消费位点重置功能，指定到具体的时间开始消费。每次重置只针对特定Group ID下的特定Topic，不会影响其他Group ID。

1.1.2. 消息队列RocketMQ版消费失败如何重新消费消息？

集群消费方式

消费业务逻辑代码如果返回 `Action.ReconsumerLater` ，或者 `NULL` ，或者抛出异常，消息都会走重试流程，至多重试16次，如果重试16次后，仍然失败，则消息丢弃。每次重试的间隔时间如下：

第几次重试	每次重试间隔时间
1	10秒
2	30秒
3	1 分钟
4	2 分钟
5	3 分钟
6	4 分钟
7	5 分钟
8	6 分钟
9	7 分钟
10	8 分钟
11	9 分钟
12	10 分钟

第几次重试	每次重试间隔时间
13	20 分钟
14	30 分钟
15	1 小时
16	2 小时

可以通过调用 message.getReconsumeTimes() 方法来获取消息的重试次数。

广播消费方式

广播消费方式仍然能保证一条消息至少被消费一次，但消费失败后不做重试操作。

1.1.3. 消息发送了，但是没有收到怎么办？

消息队列RocketMQ版

提供了多种消息查询方式：

- 使用Topic按时间范围进行查询，可以查询到一段时间内某Topic收到的所有消息。
- 使用Topic和Message ID对消息进行精准查询。
- 使用Topic和Message Key较为精准地查询具有相同Message Key的一类消息。

上述方式可以查询到消息的具体内容以及消费情况，如果需要追踪一条消息从生产者发出到被消费者消费的整个链路中各个相关节点的时间地点，可以使用

消息队列RocketMQ版

最新的消息轨迹查询功能，更多信息，请参见查询消息轨迹。

1.2. 配置相关

1.2.1. 由于.NET客户端的配置问题，导致dll加载失败或者其他运行错误时如何处理？

请参考.NET SDK压缩包中的文档：SDK_GUIDE.pdf，确认工程配置与文档说明一致。

1.3. 消息轨迹

1.3.1. 为什么查询不到轨迹数据？

当输入查询条件查不到轨迹数据时，请参考是否属于以下无法查询情况。

- 消息轨迹目前支持Java客户端（版本至少为1.2.2）。
- 查询条件是否正确，如Topic、Message ID、Message Key是否输入正确。
- 查询时间区间是否正确，为了提高查询速度，需要您输入消息的发送时间范围。如果查询不到，请尝试扩大时间范围重试。

- 如果确认上述情况无误还是无法查询到结果，请提交工单获得技术支持，并附带上日志文件，日志文件位于 */home/{user}/logs/ons.log*。

1.3.2. 确定已经消费了消息，但轨迹却显示没有消费，而且客户端IP地址和Producer ID不符合实际情况？

出现该现象是由于客户端本身并没有升级到支持轨迹功能的版本，因此

消息队列RocketMQ版

轨迹查询后端仅仅能够拿到部分不完整的轨迹数据，展现结果不正常。建议尽快升级客户端，详情请参见查询消息轨迹。

1.3.3. 消费者列表中为什么没有我的Group ID？

出现该情况的原因可能是消息的下游订阅方太多，轨迹图中空间有限，请将鼠标放在滚动条位置，滚动查看完整数据。

1.3.4. 之前的查询任务怎么看不到了？

为了防止历史查询任务太多，影响展示效果，

消息队列RocketMQ版

会对用户的历史查询任务做定期清理，默认只保留7天以内的查询任务，如果出现历史任务看不到，请重新查询即可。

1.4. 告警处理

1.4.1. 收到订阅关系不一致告警后怎么办？

消息队列RocketMQ版

里的一个Group ID代表一个Consumer实例群组。对大多数分布式应用来说，一个Group ID下通常会挂载多个Consumer实例。订阅关系一致指，同一个Group ID下所有的Consumer实例的Topic和Tag都必须完全一致。

问题描述

- Group ID订阅的部分消息未收到。登录

  消息队列RocketMQ版

  控制台，单击**消息查询 > 按Message ID查询消息**，消息显示已至少被消费一次，但从消费逻辑判断未消费。
- 登录

  消息队列RocketMQ版

  控制台，单击**Group管理 > 消费者状态**，**订阅关系是否一致**栏显示否。

问题分析

若同一个Group ID下的Consumer实例配置了不同的Topic，或是Topic相同但Tag不同 ，订阅关系就会不一致。一旦订阅关系不一致，消息消费的逻辑就会混乱，导致消息丢失。

- 可能原因一

  配置了相同Group ID的不同客户端订阅的Topic不一致。

  同一个Group ID（GID-MQ-FAQ）订阅的Topic不同（分别是MQ-FAQ-TOPIC-1、MQ-FAQ-TOPIC-2）。

  错误示例：JVM-1上的代码

```
Properties properties = new Properties();
properties.put(PropertyKeyConst.GROUP_ID, "GID-MQ-FAQ");
Consumer consumer = ONSFactory.createConsumer(properties);
consumer.subscribe("MQ-FAQ-TOPIC-1", "NM-MQ-FAQ", new MessageListener() {
    public Action consume(Message message, ConsumeContext context) {
        System.out.println("Receive: " + message);
        return Action.CommitMessage;
    }
});
consumer.start();
```

  错误示例：JVM-2上的代码

```
 Properties properties = new Properties();
properties.put(PropertyKeyConst.GROUP_ID, "GID-MQ-FAQ");
Consumer consumer = ONSFactory.createConsumer(properties);
consumer.subscribe("MQ-FAQ-TOPIC-2", "NM-MQ-FAQ", new MessageListener() {
    public Action consume(Message message, ConsumeContext context) {
        System.out.println("Receive: " + message);
        return Action.CommitMessage;
    }
});
consumer.start();
```

- 可能原因二

  同一个Group ID下的不同客户端订阅的Topic相同，但Tag不同。

  同一个Group ID（GID-MQ-FAQ）订阅的Tag不同（分别是NM-MQ-FAQ1、NM-MQ-FAQ2）。

  错误示例：JVM-1上的代码

```
Properties properties = new Properties();
properties.put(PropertyKeyConst.GROUP_ID, "GID-MQ-FAQ");
Consumer consumer = ONSFactory.createConsumer(properties);
consumer.subscribe("MQ-FAQ-TOPIC-1", "NM-MQ-FAQ1", new MessageListener() {
    public Action consume(Message message, ConsumeContext context) {
        System.out.println("Receive: " + message);
        return Action.CommitMessage;
    }
});
consumer.start();
```

  错误示例：JVM-2上的代码

```
 Properties properties = new Properties();
properties.put(PropertyKeyConst.GROUP_ID, "GID-MQ-FAQ");
Consumer consumer = ONSFactory.createConsumer(properties);
consumer.subscribe("MQ-FAQ-TOPIC-1", "NM-MQ-FAQ2", new MessageListener() {
    public Action consume(Message message, ConsumeContext context) {
        System.out.println("Receive: " + message);
        return Action.CommitMessage;
    }
});
consumer.start();
```

解决方案

1. 检查不同客户端的订阅代码。保持配置相同Group ID的所有客户端的订阅关系一致，包括订阅的Topic与Tag。
2. 重启所有客户端应用。

结果验证

- 重启所有客户端应用。
- 登录

  消息队列RocketMQ版

  控制台，单击**Group管理 > 消费者状态**，**订阅关系是否一致**栏显示**是**。

1.4.2. 消息队列RocketMQ版如何添加消费堆积告警？

1. 登录

   消息队列RocketMQ版

   控制台。
2. 在左侧导航栏，单击**实例列表**。
3. 在顶部菜单栏，选择地域，如**华东1（杭州）**。
4. 在**实例列表**页面，找到目标实例，在其**操作**列，单击**详情**。
5. 在左侧导航栏，单击**监控报警**。
6. 在**监控报警**页面左上角，单击**配置报警规则**。
7. 在**创建报警规则**页面，设置报警规则和通知方式。

   更多信息，请参见监控报警。

1.4.3. 消息队列RocketMQ版用户收到堆积告警后怎么办？

消息队列RocketMQ版

的消息发送至Broker后，配置了Group ID的客户端根据当前的消费位点从Broker拉取部分消息到本地进行消费。消费过程中访问共享资源加锁、I/O和网络资源竞争、HTTP调用未设置超时时间等原因，都会导致单条消息的消费时间很长，消息开始在服务端堆积。

问题确认

- 登录

  消息队列RocketMQ版

  控制台，选择**Group管理 > 消费者状态**，**堆积量**栏的值高于预期。
- 登录

  消息队列RocketMQ版

  控制台，选择**消息轨迹 > 创建查询任务 > 按Message ID查询**，部分消息已发送至Broker，但未投递给下游消费者。

问题分析

消息队列RocketMQ版

的消息发送至Broker后，配置了Group ID的客户端根据当前的消费位点从Broker拉取部分消息到本地进行消费。消费过程中访问共享资源加锁、I/O和网络资源竞争、HTTP调用未设置超时时间等原因，都会导致单条消息的消费时间很长，消息开始在服务端堆积。

解决方案

请按照如下方法排查并解决问题：

- 登录

  消息队列RocketMQ版

  控制台，选择**资源报表 > 消息消费**，查询历史消费记录。如果消息写入速度大于消息消费速度，调整业务代码或对消费者进行扩容。
- 在应用打印Jstack信息jstack -l {pid} | grep ConsumeMessageThread。如果有消息阻塞现象，连续打印5次Jstack信息，确认消费线程卡在哪里，解决后可尝试重启应用观察消费是否恢复。
- 如果消息已经没有堆积，检查阈值是否设置过小导致消息堆积，单击**监控报警**，单击**编辑**，增大消息堆积的报警阈值。

结果验证

- 在应用打印Jstack信息`jstack -l {pid} | grep ConsumeMessageThread`，无消费线程阻塞现象。
- 登录

  消息队列RocketMQ版

  控制台，选择**Group管理 > 消费者状态 > 消费TPS**， 栏的值上升，**堆积量**栏的值下降。

1.5. 消息堆积

1.5.1. 如何处理消息堆积？

除了异步解耦功能，

消息队列RocketMQ版

还有挡住前端数据洪峰的重要功能，以此保证后端系统的稳定性。这要求

消息队列RocketMQ版

具有一定的消息堆积能力。

消息队列RocketMQ版

能支持10亿级别的消息堆积，不会因为消息堆积导致性能明显下降。

问题描述

在

消息队列RocketMQ版

控制台的**消费者状态**页面，看到Group ID的**实时消息堆积量**的值高于预期，且性能明显下降。

解决方法

面对消息堆积，且有明显性能下降的情况，可采取以下措施处理：

1. 在

   消息队列RocketMQ版

   控制台，通过查看消费者状态获取消息堆积的消费者实例所对应的宿主机IP地址，并登录该宿主机或容器。

2. 执行以下任一命令查看进程pid。
   - `ps -ef |grep java`
   - `jps -lm`
3. 执行以下命令查看堆栈信息。

   `jstack -l pid > /tmp/pid.jstack`
4. 执行以下命令查看 `ConsumeMessageThread` 的信息，重点关注线程的状态及堆栈。

   `cat /tmp/pid.jstack|grep ConsumeMessageThread -A 10 --color`

   命令回显如下图所示。

```
"ConsumeMessageThread_28" #906 daemon prio=5 os_prio=0 tid=0x00007f08085d9000 nid=0x42c1 waiting for monitor entry [0x00007f07cc30c000]
   java.lang.Thread.State: BLOCKED (on object monitor)
        at com.taobao.txc.a.b.g.c(Unknown Source)
        - waiting to lock <0x0000000702f7d9d0> (a java.lang.Object)
        at com.taobao.txc.a.b.g.a(Unknown Source)
        at com.taobao.txc.a.b.g.b(Unknown Source)
        at com.taobao.txc.a.b.g.a(Unknown Source)
        at com.taobao.txc.client.a.a.a.a(Unknown Source)
        at com.taobao.txc.client.a.a.a.b(Unknown Source)
        at com.taobao.txc.client.a.a.a.a(Unknown Source)
        at com.taobao.txc.client.b.b.a(Unknown Source)
        at com.taobao.txc.client.aop.b.invoke(Unknown Source)
        at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:179)
        at org.springframework.aop.framework.JdkDynamicAopProxy.invoke(JdkDynamicAopProxy.java:207)
        at com.sun.proxy.$Proxy159.orderAutoAdaptation(Unknown Source)
        at com.xtep.o2o.order.adaptation.orderAutoAdaptation.mq.ConsumerOrderAdaptationMQ.consumerMassage(ConsumerOrderAdaptationMQ.java:84)
        at com.xtep.mq.OnsConsumerSpringBean$1.consume(OnsConsumerSpringBean.java:79)
        at com.aliyun.openservices.ons.api.impl.rocketmq.ConsumerImpl$MessageListenerImpl.consumeMessage(ConsumerImpl.java:179)
        at com.alibaba.rocketmq.client.impl.consumer.ConsumeMessageConcurrentlyService$ConsumeRequest.run(ConsumeMessageConcurrentlyService.java:414)
        at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511)
        at java.util.concurrent.FutureTask.run(FutureTask.java:266)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
        at java.lang.Thread.run(Thread.java:745)
```

线程状态的解释说明请参见Java官方文档。

更多信息

如按以上操作还未解决因消息堆积而导致的性能下降问题，请提交工单。提交时，请附带以下信息：

- `heap.bin` 文件

该文件可通过执行 `jmap -dump:format=b,file=heap.bin [pid]` 命令获取，再执行 `gzip heap.bin` 命令生成压缩包。[pid] 是消息堆积处理第一步中找到的进程pid。

- 发生消息堆积的消费者客户端 `ons.log` 本地日志
- 消费者客户端的版本

2.故障处理

2.1. 使用异常

2.1.1. 无法连接Broker

可能产生的原因

- 您使用的阿里云云主机（ECS）与

  消息队列RocketMQ版

  所属服务器不在同一地域（Region）。
- 您可能在非阿里云主机上访问

  消息队列RocketMQ版

  服务，且您创建的Topic不支持非阿里云主机访问。

建议解决方案

按如下步骤操作：

1. 请确保阿里云主机与创建的Topic在同一个地域。
2. 非阿里云主机访问

   消息队列RocketMQ版

   ，请确保Topic所在地域为**公网**。

2.1.2. 启动Producer、Consumer失败，Group ID重复

可能产生的原因

在同一个JVM进程里面启动多个Producer或Consumer 实例，且这些实例配置了同一个Group ID，从而导致客户端启动失败。

建议解决方案

按如下步骤操作：

1. 确保在一个JVM进程中只启动了同一个Group ID的一个Producer或Consumer实例，即可以在一个JVM进程中同时启动同一个Group ID的一个Producer和一个Consumer实例，但不能同时启动多个Producer或Consumer实例。
2. 重启应用。

2.1.3. 广播模式下，消费者启动加载JSON文件异常

可能产生的原因

Fastjson版本太低导致广播消费者加载本地的offsets.json文件异常，导致启动失败。

建议解决方案

将Fastjson的版本升级到ons-client所依赖的版本，保证本地的offsets.json能够被正常加载。默认情况下offsets.json 在*/home/{user}/.rocketmq_offsets/*下。

2.1.4. 主动订阅消息，获取队列列表失败

可能产生的原因

可能未在控制台上创建该Topic，导致订阅方启动时获取Topic的队列信息失败。

建议解决方案

按如下步骤操作：

1. 登录

   消息队列RocketMQ版

   控制台，在左侧导航栏选择 **Topic 管理 > 创建 Topic**，按提示创建Topic。
2. 在左侧导航栏选择 **Group 管理 > 创建 Group ID** ，按提示创建Group ID。
3. 重启应用。

2.1.5. 消息队列RocketMQ版报错 “Can not find name server”

如果onsaddr配置错误，日志中会报以下错误：

```
 "Exception in thread "main" com.aliyun.openservices.ons.api.exception.ONSClientException:
Cannot find name server. Please check your network connection."
```

这时，请检查以下几点。

1. 是否违背部署限制，详见主账号快速入门中主账号快速入门关于创建Topic的注意事项。
2. 检查本地和接入点之间的网络连接情况。
   - 若Topic在公网环境：

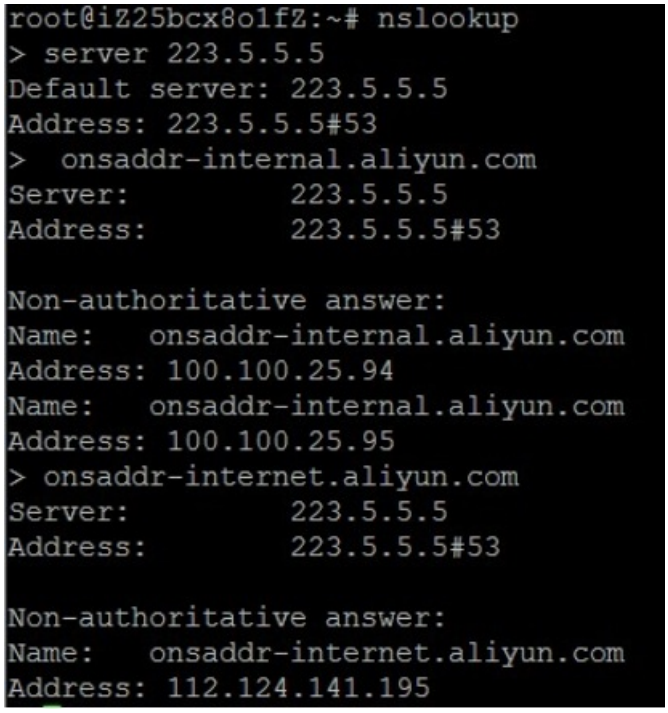
     措施： `ping onsaddr-internet.aliyun.com`

     正常情况下，会解析到 `112.124.141.195` 。
   - 若Topic在生产环境：

     措施： `ping onsaddr-internal.aliyun.com`

     正常情况下，会解析到 `100.100.25.94/95` 。

     如果无法解析接入点地址，请在本地机器上增加DNS 223.5.5.5，增加DNS 223.5.5.5成功后，可查看到：



- 另外，

  消息队列RocketMQ版

  无法设置代理，如果您使用公网环境，在申请开通安全策略时，需要将以下四个地址（端口80和8080）加入开通列表：

  112.124.141.191

  112.124.141.195

  115.28.250.94

  115.28.250.95

3. 尝试通过curl的方式从接入点获取name server的元数据信息。
   - 若Topic在生产环境：

     ```
     curl http://onsaddr-internal.aliyun.com:8080/rocketmq/nsaddr4client-internal
     ```

     返回 `100.100.26.1:8080;100.100.26.2:8080;100.100.25.96:8080` 则为正常。
   - 若Topic在公网环境：

     ```
     curl http://onsaddr-internet.aliyun.com/rocketmq/nsaddr4client-internet
     ```

     返回 `112.124.141.191:80` 则为正常。

如问题还未解决，请提交工单。

2.1.6. 消息显示_Consumed_,但消费端未感知到

消息状态显示“Consumed”，但是消费端业务日志显示没有收到消息。

这种问题一般是下面三个原因导致的。

- 业务代码在接收到消息后，不立即打印消息

收到消息后，如果直接进入业务逻辑，一旦代码遗漏某个逻辑分支，就会导致消息信息没有被留在业务日志里，造成没有收到消息的假象。

建议您收到消息后，立即打印消息信息留存messageId, timestamp, reconsumeTime 等。

- 消费端部署了多个消费实例

  尤其是在调试阶段，消费端不可避免会多次重启，一旦多个消费进程同时存在（进程未退出），那么相当于进入集群的消费模式，多个消费实例会共同分担消费消息。以为没有收到的消息，其实是被另一个消费端接收了。

  到

  消息队列RocketMQ版

  控制台，进入**Group 管理 > 消费者状态 > 连接状态**，会显示消费端的实例部署情况（有几个消费实例，各自的连接IP等等），然后可以自行排查。

- 消息消费过程出现未被Catch的异常，导致消息被重新投递。

```
public class MessageListenerImpl implements MessageListener {
    @Override
    public Action consume(Message message, ConsumeContext context) {
        //消息处理逻辑抛出异常，消息将重试
        doConsumeMessage(message);
        //如果在 doConsumeMessage() 方法中出现未被Catch的异常，这行日志将不会打印。
        log.info("Receive Message, messageId:", message.getMsgID());
        return Action.CommitMessage;
    }
}
```

如果问题还未能解决，请提供本地SDK日志，联系售后技术支持。

2.2. 资源不存在

2.2.1. Group ID不存在

可能产生的原因

未在控制台上创建该Group ID，导致使用该Group ID与 Broker创建连接的时候，服务器校验不通过。

建议解决方案

按如下步骤操作：

1. 登录

   消息队列RocketMQ版

   的控制台，进入**Group 管理**，然后创建Group ID。

2. 重启应用。

2.2.2. 主机名不存在

可能产生的原因

可能是无法正确获取主机名或者主机IP地址导致，请尝试使用此命令来证实：`hostname`

如果无法正常输出，就说明确实是此原因；如果可以正常输出，可能是其他原因，请提交工单获得技术支持。

建议解决方案

按如下步骤操作：

1. 在报错机器上执行查看主机名命令：

   ```
   [root@iZ231wxgt6mZ ~]# hostname
   iZ231wxgt6mZ
   ```

   如果执行命令报错，请检查是否给hostname定义了别名，比如在.bash_profile或者.bashrc中alias xxx='hostname'；或者命令路径不在$PATH下面。

2. Ping主机：

   ```
   [root@iZ231wxgt6mZ ~]# ping iZ231wxgt6mZ
   ```

   如果无法正常ping通主机名，则需要将本机地址绑定到*/etc/hosts*文件中。 默认ECS机器都会有一个本地地址和主机名的绑定关系，切勿手动的将其去除。

3. 确认系统配置

   检查*/etc/sysconfig/network*中的记录的hostname是否和*/etc/hosts*中的主机名绑定一致，如果不一致请修改。 如果需要修改*/etc/sysconfig/network*中的内容，修改后需要重启机器才能生效。 推荐不要随意修改系统文件里的配置，可以会引发一些其他异常问题。

以上三个步骤确认完毕后，客户端启动就不在会报UnknownHostException的异常了。

2.3. 状态不一致

2.3.1. 非预期的客户端连接

现象

- 用户侧反馈部分消息发送后未收到。登录

  消息队列RocketMQ版

  控制台。单击**消息轨迹 > 创建查询任务 > 按 Message ID 查询** ，发现部分消息已发送至broker，但未投递给下游消费者。

- 登录

  消息队列RocketMQ版

  控制台。单击**消费者管理 > 消费者状态** ，**连接状态**栏下方出现不在预期范围内的客户端IP，同时存在部分消息堆积只在非预期范围内的客户端上。

分析

- 原因

  在同一套环境中，以错误的方式（AccessKeyId、AccessKeySecret、Topic 等信息配置错误）启动了Group ID的客户端，导致该客户端进程占用了 Topic 下的部分消息队列而无法正确消费消息，消息在服务端堆积，无法实时投递给下游正确的消费者。

- 确认方式

  先根据连接状态定位有问题的进程，然后通过*/{user.home}/logs/ons.log*以及程序代码确认配置的AK、SK、Topic等信息。

- 恢复方法

  可以先将有问题的客户端进程先关闭，此时之前堆积的消息会立马进行Rebalance投递至正常的客户端，待定位到问题修复后再重启有问题的进程。（快速恢复）

验证

1. 登录

   消息队列RocketMQ版

   控制台。

2. 选择**消费者管理 > 消费者状态** 。**连接状态**栏下方显示的所有已连接的客户端都能在预期范围内，并能正常消费消息。同时**订阅关系是否一致**栏显示**是**。

2.3.2. 消息不合法

可能产生的原因

一般是消息属性、消息内容不合法，不合法的情况有：

- 消息为空
- 消息内容为空
- 消息内容长度为0
- 消息内容超过限定长度

建议解决方案

请确保消息没有以上的不合法情况，并根据异常提示进行解决。

2.3.3. 参数不合法

可能产生的原因

参数不合法的情况有以下几种：

嵌套的异常说明	异常描述
consumeThreadMin Out of range [1, 1000]	消费端线程数设置不合理
consumeThreadMax Out of range [1, 1000]	消费端线程数设置不合理
messageListener is null	未设置messageListener
consumerGroup is null	未设置Group ID
msg delay time more than 40 day	定时消息延时不能超过40天

建议解决方案

按照如下步骤操作：

1. 按照异常提示修改客户端对应参数的配置，确保其在合理范围内。
2. 重启应用。

2.3.4. 客户端状态异常

可能产生的原因

- 创建Consumer、Producer之后未调用 `start()` 方法来启动客户端；
- 创建Consumer、Producer之后 `start()` 过程有异常导致客户端启动失败；
- 创建Consumer、Producer并成功调用 `start()` 方法后，显示调用了 `shutdown()` 方法关闭了客户端。

建议解决方案

按如下步骤操作：

1. 确保创建Consumer、Producer之后调用 `start()` 保证客户端处于启动状态；
2. 查看*ons.log*判断在客户端在启动过程中是否有异常。

2.3.5. 订阅关系不一致

原因描述

在不同的JVM中启动了多个Consumer，并且给相同的Group ID配置了不同的Topic，或者是相同的Topic但Tag不同，最终导致订阅关系不一致，消息不符合预期。

错误代码示例

- 错误示例一：同一个Group ID（GID-MQ-FAQ）订阅的Topic不同（分别是MQ-FAQ-TOPIC-1、MQ-FAQ-TOPIC-2）。

  JVM-1上的代码：

```
Properties properties = new Properties();
properties.put(PropertyKeyConst.GROUP_ID, "GID-MQ-FAQ");
Consumer consumer = ONSFactory.createConsumer(properties);
consumer.subscribe("MQ-FAQ-TOPIC-1", "NM-MQ-FAQ", new MessageListener() {
    public Action consume(Message message, ConsumeContext context) {
        System.out.println("Receive: " + message);
        return Action.CommitMessage;
    }
});
consumer.start();
```

  JVM-2上的代码

```
 Properties properties = new Properties();
properties.put(PropertyKeyConst.GROUP_ID, "GID-MQ-FAQ");
Consumer consumer = ONSFactory.createConsumer(properties);
consumer.subscribe("MQ-FAQ-TOPIC-2", "NM-MQ-FAQ", new MessageListener() {
    public Action consume(Message message, ConsumeContext context) {
        System.out.println("Receive: " + message);
        return Action.CommitMessage;
    }
});
consumer.start();
```

- 错误示例二：同一Group ID（GID-MQ-FAQ）订阅的Topic相同，但Tag不同（分别NM-MQ-FAQ-1、NM-MQ-FAQ-2）。

  JVM-1上的代码：

```
Properties properties = new Properties();
properties.put(PropertyKeyConst.GROUP_ID, "GID-MQ-FAQ");
Consumer consumer = ONSFactory.createConsumer(properties);
consumer.subscribe("MQ-FAQ-TOPIC-1", "NM-MQ-FAQ-1", new MessageListener() {
    public Action consume(Message message, ConsumeContext context) {
        System.out.println("Receive: " + message);
        return Action.CommitMessage;
    }
});
consumer.start();
```

  JVM-2上的代码

```
 Properties properties = new Properties();
properties.put(PropertyKeyConst.GROUP_ID, "GID-MQ-FAQ");
Consumer consumer = ONSFactory.createConsumer(properties);
consumer.subscribe("MQ-FAQ-TOPIC-1", "NM-MQ-FAQ-2", new MessageListener() {
    public Action consume(Message message, ConsumeContext context) {
        System.out.println("Receive: " + message);
        return Action.CommitMessage;
    }
});
consumer.start();
```

建议解决方案

请确保在不同JVM中使用相同的Group ID启动多个Consumer时，配置的Topic和Tag是一致的。

2.4. 日志相关

2.4.1. 如何根据客户端日志判断当前状态？

消息队列RocketMQ版

客户端日志文件是ons.log，包括INFO、WARN、ERROR级别的日志。

本文提供常见的客户端日志打印信息，旨在帮助您更好地从打印的日志中获取信息，并判断当前状态，从而排查故障。

下表列举了ons.log日志信息说明（持续更新）。

日志级别	打印信息	说明	解决方案
INFO	[persistAll] Group: CID_XXXX ClientId: 10.31.40.100@171374#14159XXX#-2036649XXX#20931314294957XXX updateConsumeOffsetToBroker MessageQueue [topic=XXXX, brokerName=qdinternetorder-XX, queueId=X] 1013XXX	这种现象说明消息已经消费成功，并且在 消息队列RocketMQ版 服务端已持久化消费进度；MessageQueue里包括了消息主题、对应的brokerName、消费队列的ID	不涉及
INFO	[PULL_TPS] [CID_XXXX@CID_XXXX] Stats In One Minute, SUM: 0 TPS: 0.00 AVGPT: 0.00</br> [PULL_RT] [%RETRY%CID_XXXX@CID_XXXX] Stats In One Minute, SUM: 0 TPS: 0.00 AVGPT: 0.00	该类信息打印的是从consumeQueue中拉取消息时的TPS（每秒Request的数量）	不涉及
WARN	[TIMEOUT_CLEAN_QUEUE]broker busy, start flow control for a while, period in queue: 905ms, size of queue: 1164	消息队列RocketMQ版 服务端压力过大，处理不了过多的请求；由于服务端在存储数据时是先写入pageCache，然后去刷盘，因此每隔10s会去清理过期的请求（此过程会判断缓存页是否繁忙）	1. 扩容，增加Broker，分担压力 2. osPageCacheBusyTimeOutMills属性值调大
WARN	execute the pull request exception com.aliyun.openservices.shade.com.alibaba.rocketmq.client.exception.MQBrokerException: CODE: 25 DESC: the consumer's subscription not latest	Broker每隔一段时间就会向NameServer上报自己的路由信息，如果此过程网络抖动，拉不到最新的订阅信息，导致消费者消费的时候，会出现该警告	不涉及

日志级别	打印信息	说明	解决方案
WARN	`[WRONG]mq is consuming, so can not unlock it, MessageQueue [topic=XX, brokerName=szorder2-02, queueId=1]. maybe hanged for a while, 2`	进行负载均衡时，对消息处理队列尝试加锁，如果1s内还未加锁成功，说明当前消息处理队列已经有消费者在访问，不能进行解锁	不涉及
WARN	`doRebalance, XXX-CID, add a new mq failed, MessageQueue [topic=XXXX, brokerName=szorder2-XX, queueId=X], because lock failed`	当前使用的是顺序Topic，为了保证单个分区中消息的顺序消费，会有个Lock的机制。客户端有这个日志说明其中某个分区已经有客户端在消费了。	不涉及
WARN	`get Topic [XXXXXX] RouteInfoFromNameServer is not exist value com.aliyun.openservices.shade.com.alibaba.rocketmq.client.exception.MQClientException: CODE: 17 DESC: No topic route info in name server for the topic: TOPIC_XXXXX</br>See http://rocketmq.apache.org/docs/faq/ for further details.`	• AccessKey（包含AccessKeyId和AccessKeySecret）配置错误 • 没有控制台于当前实例下创建的Group ID（GID） • 实例化的代码中，NameServerAddr没有配置正确	• 配置正确的AccessKey • 在当前实例下创建GID • Java SDK 1.8.0及以上版本，推荐配置NameServerAddr，此参数可从 消息队列RocketMQ版 控制台获取，与之前版本配置的ONSAddr是不一致的
WARN	`com.aliyun.openservices.ons.api.impl.authority.exception.AuthenticationException: signature validate by dauth failed`	AccessKey（包含AccessKeyId和AccessKeySecret）配置错误	AccessKey要配置创建该GID使用的AccessKey

日志级别	打印信息	说明	解决方案
WARN	`NettyClientPublicExecutor_3 - execute the pull request exception com.aliyun.openservices.shade.com.alibaba.rocketmq.client.exception.MQBrokerException: CODE: 26 DESC: subscription group [CID_XXX] does not exist, See http://rocketmq.apache.org/docs/faq/ for further details.`	订阅关系没有推送到消息队列RocketMQ版broker上	subscription.json文件里直接添加GID对应的信息即可
WARN	`execute the pull request exception com.aliyun.openservices.shade.com.alibaba.rocketmq.client.exception.MQBrokerException: CODE: 24 DESC: the consumer's subscription not exist`	缺少订阅关系	不涉及

2.5. 应用内存不足

现象

- 在应用部署的机器上通过查看内存已消耗完。
- 在*/{user.home}/logs/ons.log*能搜索到Out Of Memory关键字。
- 用户侧反馈

  消息队列RocketMQ版

  控制台：进入**消费者管理 > 消费者状态**，**堆积量**栏显示消息堆积较多，**连接状态**栏显示了各个已连接客户端的消息堆积。通过Jstack排查，`ConsumeMessageThread_`线程无消费卡住现象。

分析

在1.7.0.Final版本之前，默认客户端最多会给每个Topic的每个队列缓存1000条消息。假设每个Topic的队列数是16个（集群2主2备，每台broker上8个队列），该Topic下单条消息平均大小为64 KB，那么最终该Topic在客户端缓存的消息Size：16 * 1000 * 64 KB = 1 GB。如果用户同时订阅了8个Topic都在客户端内存缓存消息，最终占用内存将超过用户的JVM配置，导致OOM。

- 原因1

依赖1.7.0.Final之前的ons-client版本，Topic的平均消息大小超过4 KB，并且消息消费较慢容易在客户端内存缓存消息。

确认方式

在*/{user.home}/logs/ons.log*能搜索到Out Of Memory关键字；或者通过 `jmap -dump:live,format=b,file=heap.bin <pid>` 命令确认哪些对象占用了大量内存。

恢复方案

升级ons-client版本至1.7.0.Final或以上，同时给对应的ConsumerBean配合设置 `com.aliyun.openservices.ons.api.PropertyKeyConst#MaxCachedMess ageSizeInMiB` 参数，然后重启应用。

- 原因2

  依赖1.7.0.Final及以上的ons-client版本，默认最大消耗内存512 MB（Group ID订阅的所有Topic缓存总和）；如果应用仍然出现OOM现象，可在ConsumerBean启动时，配置 `com.aliyun.openservices.ons.api.PropertyKeyConst#MaxCachedMessageSizeInMiB` 参数自行定义最大消耗内存（范围在16 MB ~ 2048 MB）。

  确认方式

  确认应用依赖的ons-client版本，并通过JVM确认给进程分配的内存大小。

  恢复方案

  根据应用机器的内存使用清况给对应的ConsumerBean设置 `com.aliyun.openservices.ons.api.PropertyKeyConst#MaxCachedMessageSizeInMiB` 参数，然后重启应用。

验证

- 在*/{user.home}/logs/ons.log*中Out Of Memory关键字不再出现。
- 登录

  消息队列RocketMQ版

  控制台，进入**消费者管理 > 消费者状态**，看到**消费TPS**栏的值上升，同时**堆积量**的值下降。