ALIBABA CLOUD

阿里云

消息队列RabbitMQ版

常见问题

文档版本：20211216

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.QueueHasDiffField ---- 05
2.ExchangeHasDiffFields ---- 07
3.消费失败是否支持重试? ---- 10
4.否定应答并重入队列 ---- 11
5.设置Message ID ---- 12
6.删除队列消息 ---- 13
7.是否支持优先级队列? ---- 14
8.消息保存时长是多久? ---- 15
9.为什么在控制台页面查询不到监控信息? ---- 16
10.Headers Exchange绑定 ---- 17
11.计费FAQ ---- 19

1.QueueHasDiffField

介绍消息队列RabbitMQ版客户端报QueueHasDiffField类型错误的原因和处理方法。

Condition

使用消息队列RabbitMQ版客户端连接消息队列RabbitMQ版服务端时，报QueueHasDiffField类型错误。 例如QueueHasDiffField[OAutoDelete=false&NAutoDelete=true;] 。

> **说明** 报错中O开头的参数的属性值为要调用的Queue的参数的已设置属性值，N开头的参数的属性值为本次声明的Queue的参数的属性值。

Cause

要调用的Queue的参数的已设置属性值与本次声明的Queue的参数的属性值不一致，导致报QueueHasDiffField类型错误。可能出现不一致的Queue参数如下。

参数	类型	描述
queue	String	Queue的名称。
durable	boolean	Queue是否持久化： • true：持久化类型，在消息队列RabbitMQ版客户端重连消息队列RabbitMQ版服务端时被再次自动创建出来 。 • false：非持久化类型，在消息队列RabbitMQ版客户端重连消息队列RabbitMQ版服务端时不会被再次自动创建出来 。 > **说明** 调用CreateQueue或在消息队列RabbitMQ版控制台的**Queue管理**页面创建的Queue默认为持久化Queue。
exclusive	boolean	Queue是否具有排他性： • true：排他性类型，只对首次声明其的Connection可见，且会在Connection断开时自动删除。 • false：非排他性类型，对其他Connection可见，不会在Connection断开时自动删除。
autoDelete	boolean	Queue是否自动删除： • true：自动删除类型，在最后一个Consumer取消订阅后，自动删除。 • false：非自动删除类型，即使最后一个Consumer取消订阅后，也不会自动删除。

参数	类型	描述
arguments	Map	Queue的其他参数。包括死信Exchange、死信Routing Key和消息过期时间。

例如要调用的Queue的autoDelete参数的已设置属性值与本次声明的Queue的autoDelete参数的属性值不一致，导致报错QueueHasDiffField[OAutoDelete=false&NAutoDelete=true;]。

- 要调用的Queue的autoDelete参数的已设置属性值为false。

  > **说明** 您可以调用ListQueues或在消息队列RabbitMQ版控制台的Queue管理页面获取要调用的Queue的参数的属性值。

- 本次声明的Queue的autoDelete参数的属性值为true。
  示例代码如下：

```
channel.queueDeclare("test", false, false, true, null);
```

Remedy

操作步骤

1. 在代码中修改本次声明Queue的参数的属性值，使其与要调用的Queue的参数的已设置属性值保持一致。例如，在代码中将本次声明的Queue的autoDelete参数的属性值修改为false，使其与要调用的Queue的autoDelete参数的已设置属性值保持一致。

   示例代码如下：

```
channel.queueDeclare("test", false, false, false, null);
```

2.ExchangeHasDiffFields

介绍消息队列Rabbit MQ版客户端报ExchangeHasDiffFields类型错误的原因和处理方法。

Condition

使用消息队列Rabbit MQ版客户端连接消息队列Rabbit MQ版服务端时，报ExchangeHasDiffFields类型错误。例如ExchangeHasDiffFields[ODurable=true&NDurable=false;]。

> **说明** 报错中O开头的参数的属性值为要调用的Exchange的参数的已设置属性值，N开头的参数的属性值为本次声明的Exchange的参数的属性值。

Cause

要调用的Exchange的参数的已设置属性值与本次声明的Exchange的参数的属性值不一致，导致报ExchangeHasDiffField类型错误。可能出现不一致的Exchange参数如下：

参数	类型	描述
exchange	String	Exchange的名称。
type	String	Exchange的类型。取值： • fanout：该类型路由规则非常简单，会把所有发送到该Exchange的消息路由到所有与它绑定的Queue中，相当于广播功能。 • direct：该类型路由规则会将消息路由到Binding Key与Routing Key完全匹配的Queue中。 • topic：该类型与direct类型相似，只是规则没有那么严格，可以模糊匹配和多条件匹配，即该类型Exchange使用Routing Key模式匹配和字符串比较的方式将消息路由至绑定的Queue。

参数	类型	描述
durable	boolean	Exchange是否持久化。取值： • true：持久化类型，在消息队列RabbitMQ版客户端重连消息队列RabbitMQ版服务端时被再次自动创建出来 。 • false：非持久化类型，在消息队列RabbitMQ版客户端重连消息队列RabbitMQ版服务端时不会被再次自动创建出来 。 说明 调用CreateExchange或在消息队列RabbitMQ版控制台的Exchange管理页面创建的Exchange默认为持久化Exchange。
autoDelete	boolean	Exchange是否自动删除。取值： • true：自动删除类型，在最后一个绑定的Queue取消绑定后，自动删除。 • false：非自动删除类型，即使最后一个绑定的Queue取消绑定后，也不会自动删除。 说明 调用CreateExchange或在消息队列RabbitMQ版控制台的Exchange管理页面创建的Exchange默认为非自动删除类型。
internal	boolean	Exchange是否为Internal类型。默认值为false。取值： • true：内建类型，用于Exchange和Exchange之间的绑定。 • false：非内建类型，用于Exchange和Queue之间的绑定。

参数	类型	描述
arguments	Map	Exchange其他参数。包括Alternate Exchange。

例如要调用的Exchange的durable参数的已设置属性值与本次声明的Exchange的durable参数的属性值不一致，导致报错ExchangeHasDiffFields[ODurable=true&NDurable=false;]。

- 要调用的Exchange的durable参数的已设置属性值为false。

  > **说明** 您可以调用ListExchanges或在消息队列RabbitMQ版控制台的Exchange管理页面获取要调用的Exchange的参数的属性值。

- 本次声明的Exchange的durable参数的属性值为true。
  示例代码如下：

```
channel.exchangeDeclare("test", "direct", false, false, false, null);
```

Remedy

操作步骤

1. 在代码中修改本次声明Exchange的参数的属性值，使其与要调用的Exchange的参数的已设置属性值保持一致。例如，在代码中将本次声明的Exchange的durable参数的属性值修改为false，使其与要调用的Exchange的durable参数的已设置属性值保持一致。

   示例代码如下：

```
channel.exchangeDeclare("test", "direct", true, false, false, null);
```

3.消费失败是否支持重试？

消息队列RabbitMQ版默认支持消费失败后消息重试。

重试机制

消息队列RabbitMQ版服务端有默认的消息重试机制，不支持您在Consumer客户端重新配置消息重试机制和关闭消息重试机制。消息队列RabbitMQ版服务端默认的消息重试机制如下：

- 如果您没有开启Consumer客户端消费消息，就不会触发消息重试。
- 如果您开启了Consumer客户端消费消息，消费失败，即Consumer客户端一分钟内没有应答消息，则触发消息重试：
  - 重试期间，任何一次消费成功，即Consumer客户端应答消息，则立即停止消息重试。
  - 重试时间间隔为60秒。
  - 重试最多16次。超过16次，则停止重试。您可以选择：
    - 丢弃消息：如果您没有为重试失败的消息所在的Queue配置死信Exchange，则消息重试失败后被丢弃。
    - 将消息发送至死信Exchange：如果您为重试失败的消息所在的Queue配置了死信Exchange，则消息重试失败后被发送到死信Exchange，并根据RoutingKey和Binding Key被路由至目标Queue。目标Queue中的消息支持查询和导出。如何配置死信Exchange，请参见死信Exchange。

4.否定应答并重入队列

当某个Consumer不能立即消费消息且其他Consumer可以消费时，您可以使该Consumer否定应答消息并使其重入队列，从而使其他Consumer可以消费消息。本文说明Consumer如何否定应答消息并使其重入队列。

Consumer否定应答单条消息并使其重入队列

basicReject方法可以否定应答一条消息。您可以使用basicReject方法实现Consumer否定应答单条消息并使其重入队列。

- basicReject方法的参数说明如下：
  - deliveryTag：消息的Tag。
  - requeue：被否定应答的消息是否重入队列。如果设置为true，则消息重入队列；如果设置为false，则消息被丢弃或发送到死信Exchange，请参见死信Exchange。
- basicReject方法的示例代码如下：

```
channel.basicConsume("test", false, "consumertag", new DefaultConsumer(channel) {
   @Override
    public void handleDelivery(String consumerTag, Envelope envelope,
                      AMQP.BasicProperties properties, byte[] body)
            throws IOException {
        System.out.println("Rejected: " + new String(body, "UTF-8") + ", deliveryTag: " +
envelope.getDeliveryTag() + ", messageId: " + properties.getMessageId());
        channel.basicReject(envelope.getDeliveryTag(), true);
    }
});
```

Consumer否定应答多条消息并使其重入队列

basicNack方法可以否定应答一条或多条消息。您可以使用basicNack方法实现Consumer否定应答多条消息并使其重入队列。

- basicNack方法的参数说明如下：
  - deliveryTag：消息的Tag。
  - multiple：是否否定应答多条消息。如果设置为true，则否定应答带指定deliveryTag的消息及该deliveryTag之前的多条消息；如果设置为false，则仅否定应答带指定deliveryTag的单条消息。
  - requeue：被否定应答的消息是否重入队列。如果设置为true，则消息重入队列；如果设置为false，则消息被丢弃或发送到死信Exchange，请参见死信Exchange。
- basicNack方法的示例代码如下：

```
channel.basicConsume("test", false, "consumertag", new DefaultConsumer(channel) {
   @Override
    public void handleDelivery(String consumerTag, Envelope envelope,
                      AMQP.BasicProperties properties, byte[] body)
            throws IOException {
        System.out.println("Rejected: " + new String(body, "UTF-8") + ", deliveryTag: " +
envelope.getDeliveryTag() + ", messageId: " + properties.getMessageId());
        channel.basicNack(envelope.getDeliveryTag(), true, true);
    }
});
```

5.设置Message ID

如需追踪和识别消息，您可以在消息队列RabbitMQ版的Producer客户端设置Message ID属性，为每条消息设置唯一标识符。本文介绍Message ID的相关概念和设置方法。

什么是Message ID

Message ID（消息标识符）是消息的可选属性，类型为short string。Message ID在业务上通常被设置为唯一，适用于追踪和识别销售单、工单等需要保证消息唯一的场景。消息队列RabbitMQ版服务端不会对消息进行幂等处理。如需实现消息幂等，即如果消息重试多次，消费者端对该重复消息消费多次与消费一次的结果是相同的，并且多次消费没有对系统产生副作用，在为每条消息设置唯一Message ID的基础上，您还需要在消息队列RabbitMQ版的Consumer客户端对消息进行幂等处理，详情请参见消息幂等。

设置方法

在消息队列RabbitMQ版的Producer客户端设置 `Basic.Properties` 的 `message-id` 属性。示例代码如下：

Java | Python | PHP | Go | Node.js

```
AMQP.BasicProperties props = new AMQP.BasicProperties.Builder().messageId("messageid").build();
channel.basicPublish("${ExchangeName}", "BindingKey", true, props, ("消息发送Body").getBytes(StandardCharsets.UTF_8));
```

6.删除队列消息

本文介绍如何删除某个队列的所有消息。

您可以使用Java客户端库中 `queuePurge` 方法删除某个队列的所有消息。示例代码如下：

```
channel.queuePurge("queue-name");
```

7.是否支持优先级队列？

消息队列RabbitMQ版不支持优先级队列。

消息队列RabbitMQ版内部暂未实现消息优先级的处理逻辑，您的客户端通过可选队列参数x-max-priority声明的优先级队列实际是无效的。

8.消息保存时长是多久？

消息队列Rabbit MQ版成功消费与未成功消费的所有消息的最大保留时长为3天。

更多使用限制信息，请参见使用限制。

9.为什么在控制台页面查询不到监控信息？

当前登录用户为RAM用户，RAM用户没有被授予查看监控信息的权限。
阿里云账号给RAM用户授予只读访问云监控的权限（AliyunCloudMonitorReadOnlyAccess）后，RAM用户即可看到监控信息。具体操作，请参见步骤二：为RAM用户添加权限。

云监控查看权限策略内容如下：

```
{
    "Version": "1",
    "Statement": [
        {
            "Action": "cms:QueryMetric*",
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

10.Headers Exchange绑定

如果不使用Routing Key去做绑定，而是根据消息Headers属性和Binding Headers属性的匹配规则路由消息，需要使用Headers Exchange。本文介绍Headers Exchange的使用示例。

背景信息

- 向Headers Exchange发送消息时，可以在Headers中定义键值对。Headers Exchange将根据消息Headers属性键值对和绑定属性键值对的匹配情况路由消息。
  匹配算法由一个特殊的绑定属性键值对控制。该属性为x-match，只有以下两种取值：
  - *all*：所有除x-match以外的绑定属性键值对必须和消息Headers属性键值对匹配才会路由消息。
  - *any*：只要有一组除x-match以外的绑定属性键值对和消息Headers属性键值对匹配就会路由消息。

  更多信息，请参见Headers Exchange。
- 绑定成功后，您可以在消息队列RabbitMQ版控制台的**消息查询**页面，按照Queue查询消息，验证绑定结果。具体操作，请参见查询消息。

示例代码

Headers Exchange绑定Java示例代码如下：

```
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.HashMap;
import java.util.Map;
import java.util.UUID;
import java.util.concurrent.TimeoutException;
import com.rabbitmq.client.AMQP.BasicProperties;
import com.rabbitmq.client.Channel;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.ConnectionFactory;
import com.rabbitmq.client.ReturnListener;
public class headersTestNew {
    public static void main(String[] args)
        throws IOException, TimeoutException, NoSuchAlgorithmException, InvalidKeyException
, InterruptedException {
        ConnectionFactory factory = new ConnectionFactory();
        // 设置接入点，在消息队列RabbitMQ版控制台实例详情页面查看。
        factory.setHost("xxx.xxx.aliyuncs.com");
        // ${instanceId}为实例ID，在消息队列RabbitMQ版控制台获取。
        // ${AccessKey}阿里云身份验证，在阿里云RAM访问控制台创建。
        // ${SecretKey}阿里云身份验证，在阿里云RAM访问控制台创建。
        // 注意：开启Connection才能自动恢复。
        factory.setCredentialsProvider(new AliyunCredentialsProvider("${AccessKey}", "${Sec
retKey}", "${instanceId}"));
        factory.setAutomaticRecoveryEnabled(true);
        factory.setNetworkRecoveryInterval(5000);
        // 设置Vhost名称，请确保已在消息队列RabbitMQ版控制台创建。
        factory.setVirtualHost("${VhostName}");
        // 默认端口，非加密端口5672，加密端口5671。
        factory.setPort(5672);
        // 基于网络环境合理设置超时时间。
```

```
        factory.setConnectionTimeout(30 * 1000);
        factory.setHandshakeTimeout(30 * 1000);
        factory.setShutdownTimeout(0);
        // 请尽可能使用长期存活的Connection，以免每次发送消息都创建新的Connection，导致大量的网络资源和服务端资源消耗，甚至引起服务端SYN Flood防护。
        Connection connection = factory.newConnection();
        Channel channel = connection.createChannel();
        String exchangeName = "${ExchangeName}";
        String exchangeType = "headers";
        String queueName = "${QueueName}";
        String routingKey = "${RoutingKey}";
        Map<String, Object> argument = new HashMap<>();
        argument.put("format", "pdf");
        argument.put("type", "log");
        argument.put("x-match", "all");
        channel.queueDeclare(queueName, true, false, false, null);
        channel.exchangeDeclare(exchangeName, exchangeType, true, false, false, null);
        channel.queueBind(queueName, exchangeName, routingKey, argument);
        // 当mandatory=true，消息没有路由时，将会返回给客户端。
        channel.addReturnListener(new ReturnListener() {
            @Override
            public void handleReturn(int replyCode, String replyText, String exchange, Stri
ng routingKey,
                                     BasicProperties properties, byte[] body) throws IOExce
ption {
                System.out.println("no route,  msgId=" + properties.getMessageId());
            }
        });
        // 设置消息Headers属性键值对。
        // 1.当注释(type, log)键值对，仅(format, pdf)一组键值对与argument匹配，执行该代码后，消息将无法接收到。
        // 2.当注释(type, log)键值对被取消，即(format, pdf)和(type, log)两组键值对与argument完全匹配，执行该代码后，将接收到消息。
        Map<String, Object> headers = new HashMap<>();
        headers.put("format", "pdf");
        //headers.put("type", "log");
        BasicProperties props = new BasicProperties.Builder().messageId(UUID.randomUUID().t
oString()).headers(headers).build();
        channel.basicPublish(exchangeName, routingKey, true, props, ("消息发送Body").getByte
s(StandardCharsets.UTF_8));
        Thread.sleep(10000);
        connection.close();
    }
}
```

11.计费FAQ

本文汇总了消息队列RabbitMQ版计费的常见问题。

向10个Queue发送同一条消息，为什么收10次API调用费？

将1条消息发送给10个Queue，实际上是10次API调用。

为什么删除的Vhost、Exchange、Queue会被自动重建并重新开始计费？

虽然删除了Vhost、Exchange、Queue，但Producer和Consumer依然与消息队列RabbitMQ版保持连接，即代码逻辑层面还在主动发起连接到消息队列RabbitMQ版的请求。连接的参数指向之前被删除的资源，根据AMQP协议这些资源将被重建。您需要断开Producer和Consumer与消息队列RabbitMQ版的连接以停止计费。

我昨天删除了Queue，为什么今天会收到账单并被扣费？

Queue资源占用费以每天00:00:00~23:59:59为周期进行计费，次日收取费用。所以，如果您昨天删除Queue，昨天的计费系统已经计入，今天会出账单进行扣费，明天就不会出账单扣费。

我没有使用消息队列RabbitMQ版服务，为什么会扣费？

请登录费用管理中心查看账单，检查您是否有使用消息队列RabbitMQ版服务。

如果您不需要使用消息队列RabbitMQ版服务，请及时删除消息队列RabbitMQ版控制台上所有资源，避免不必要的支出。

实际使用量超过购买的包年包月实例的规格怎么办？

您实际使用量超过购买的包年包月实例的规格时，将会被限流，超出部分不会转按量付费。

我有看到消息队列RabbitMQ版的扣费，但是我在控制台上看到没有开通消息队列RabbitMQ版？

消息队列RabbitMQ版处于欠费状态超过72小时，阿里云将暂停为您提供服务，即您不能再访问消息队列RabbitMQ版控制台与消息队列RabbitMQ版的API。但是您的消息队列RabbitMQ版服务被释放之前的欠费仍需结清。

如何关闭消息队列RabbitMQ版服务？

请删除所有地域的Vhost、Exchange和Queue资源，并停止所有发送端和消费端。

为什么不能再开通按量付费实例？

消息队列RabbitMQ版不再支持开通新的按量付费实例，已有的按量付费实例可以继续使用，推荐您使用包年包月实例。