

# **TDMQ for RabbitMQ**

## **Getting Started**

### **Product Documentation**



## Copyright Notice

©2013-2019 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

## Trademark Notice



All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

## Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

# Contents

## Getting Started

Resource Creation and Preparation

Using SDK to Receive/Send Message (Java)

# Getting Started

## Resource Creation and Preparation

Last updated : 2022-07-04 15:32:58

### Overview

This document describes how to create resources such as cluster, vhost, exchange, and queue in the TDMQ console and what operations you need to perform in the console before running a client.

### Prerequisites

- You have [signed up for a Tencent Cloud account](#).

### Directions

#### Step 1. Create a cluster

- Log in to the [TDMQ console](#), enter the **Cluster** page, and select the target region.
- Click **Create Cluster** and enter the cluster name and description.

3. Click **Submit**.

### Create Cluster ✕

ⓘ There is currently 1 cluster, and 10 more can be created.

Cluster Name	<input type="text" value="Please enter the cluster name"/>
Exchange Capacity	1000
Queue Capacity	1000
TPS Per Vhost	8000
Cluster Description	<input type="text"/>
Resource Tag	<input type="text" value="Tag key"/> <input type="text" value="Tag value"/> <span>✕</span>

[+ Add](#)

4. On the **Cluster** list page, click **Access Address** in the **Operation** column of the cluster you just created to get the connection information of the server.

Cluster ID/Name	Exchange Count	Queue Count	Description	Resource Tag	Operation
<input type="checkbox"/> amqp-rabbitmq-test	Used: 1 Capacity: 1000	Used: 1 Capacity: 1000	test	<b>API Call Address</b> VPC Access Address amqp://amqp-rabbitmq-test-am8xdr.rabbitmq.ap-sh.tencentttdmq.com:5102 Public Network Access Address amqp://amqp-rabbitmq-test-8xdr.rabbitmq.ap-sh.public.tencentttdmq.com:5672	<a href="#">Access Address</a> <a href="#">Edit</a> <a href="#">Delete</a>

Total Items: 1

1 / 1 page

OK

## Step 2. Create a vhost

1. On the **Cluster** list page, click the ID of the cluster you just created to enter the cluster's basic information page.
2. Select the **Vhost** tab at the top, click **Create**, and enter the vhost information.
  - Vhost Name: Enter the vhost name, which cannot be modified after creation and can contain 3–64 letters, digits, "-", and "\_".
  - Message TTL: Set the retention time of unconsumed messages. Messages will be automatically deleted if not acknowledged after expiration. Value range: 60 seconds–15 days.
  - Remarks: Enter the vhost remarks.

3. Click **Submit**.

## Create Vhost ✕

**i** There is currently 1 vhost, and 10 more can be created.

Vhost Name \*   
It can contain 1-64 letters, digits, "-", and "\_".

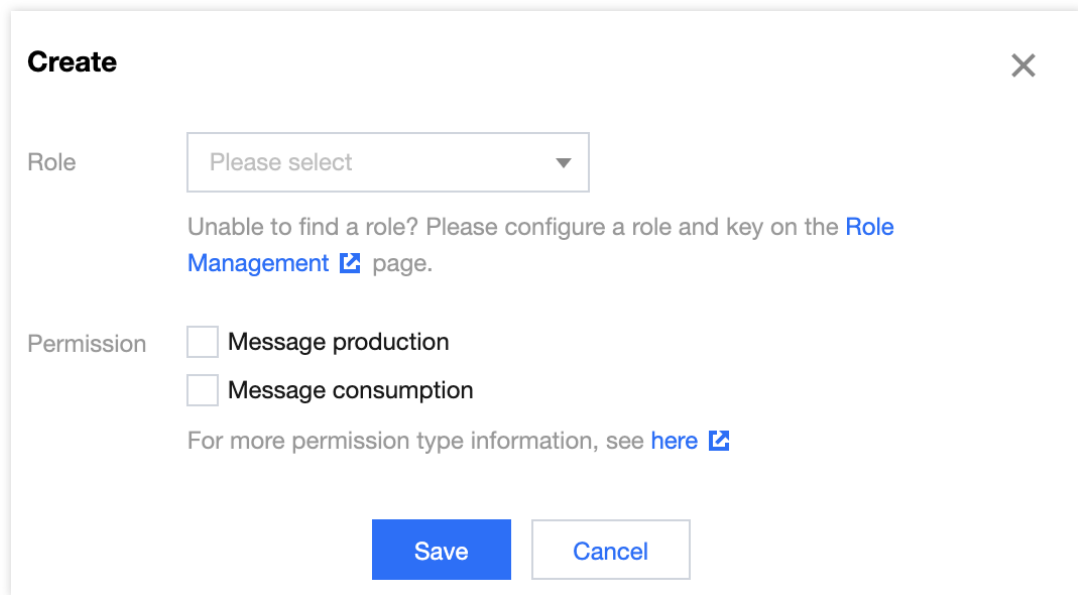
Message TTL **i** \*    
Retention period of unconsumed messages. They will be automatically deleted if they are not acknowledged within this period. Range: 60 seconds-15 days

Vhost Description

### Step 3. Create a role and configure permissions

1. Select **Role Management** on the left sidebar in the [TDMQ console](#) and click **Create** to create a role.
2. On the **Cluster** page, click the ID of the cluster you just created to enter the cluster details page.
3. Select the **Vhost** tab at the top and click **Configure Permissions** in the **Operation** column of the vhost you just created.

4. On the **Configure Permission** page, click **Add Role** to add production and consumption permissions to the role



The screenshot shows a 'Create' dialog box with a close button (X) in the top right corner. It contains a 'Role' dropdown menu with the text 'Please select'. Below the dropdown is a message: 'Unable to find a role? Please configure a role and key on the [Role Management](#) page.' Underneath, there are two unchecked checkboxes: 'Message production' and 'Message consumption'. At the bottom, there is a link: 'For more permission type information, see [here](#).' At the very bottom of the dialog are two buttons: 'Save' (in blue) and 'Cancel' (in white with a blue border).

you just created.

#### Step 4. Create an exchange

1. On the **Vhost** list page, select the **Exchange** tab at the top to enter the **Exchange** list page.
2. Select the vhost you just created, click **Create**, enter the exchange name and description, and select the route type.
  - Direct: A direct exchange will route messages to the queue whose binding key exactly matches the routing key.
  - Fanout: A fanout exchange will route messages to all queues bound to it.
  - Topic: A topic exchange supports multi-condition match and fuzzy match; that is, it will route messages to the queues bound to it by using routing key pattern match and string comparison.
  - X-delayed-message: By declaring this exchange type, you can customize the header attribute **x-delay** of the message to specify the time period for delayed message delivery in milliseconds. The message will be delivered to the corresponding queue based on the routing rule after the time period defined in **x-delay** elapses. The routing rule is subject to the route type specified in **x-delayed-type**.



3. Click **Submit**.

## Create Exchange ✕

**i** There is currently 1 exchange, and 1000 more can be created.

Current Vhost **vh1**

Exchange Name \*   
This field is required. Please enter 1-64 letters, digits, or symbols (".", "-", or "\_").

Route Type \*   
For route type descriptions, see [Route Type](#)

Exchange Description   
Up to 128 characters

### Step 5. Create a queue

1. On the **Exchange** list page, select the **Queue** tab at the top to enter the **Queue** list page.
2. Select the Vhost you just created, click **Create**, and enter the queue name, remarks, and dead letter exchange.

3. Click **Submit**.

## Create Queue ✕

**i** There is currently 1 queue, and 1000 more can be created.

Vhost **vh1**

Queue Name \*

This field is required. Please enter 1-64 letters, digits, or symbols (".", "-", or "\_").

Automatic Deletion

The queue will be immediately deleted after the last consumer unsubscribes from it.

Queue Description

Up to 128 characters

Advanced Settings ▼

**Submit** **Disable**

## Step 6. Create a routing

1. On the **Queue** list page, select the **Routing** tab at the top to enter the **Routing** list page.
2. Select the Vhost you just created and click **Create**.

Select the exchange you just created as the source exchange, **Queue** as the binding type, and the queue you just

created as the binding target.

3. Click **Submit**.

### Create Binding ✕

Vhost vh1

Source Exchange \* Please select ▼

Binding Target Type

Binding Target \* Please select ▼

# Using SDK to Receive/Send Message (Java)

Last updated : 2022-07-04 15:32:58

## Overview

This document describes how to use open-source SDK to send and receive messages using the SDK for Java as an example and helps you better understand the message sending and receiving processes.

### Note

The following takes the Java client as an example. For clients in other languages, see [TDMQ for RabbitMQ](#).

## Prerequisites

- [You have created the required resources.](#)
- [You have installed JDK 1.8 or later.](#)
- [You have installed Maven 2.5 or later.](#)
- [You have downloaded the demo.](#)

## Directions

### Step 1. Install the Java dependency library

Add the following dependencies to the `pom.xml` file:

```
<!-- in your <dependencies> block -->
<dependency>
<groupid>com.rabbitmq</groupid>
<artifactid>amqp-client</artifactid>
<version>5.13.0</version>
</dependency>
```

### Step 2. Produce messages

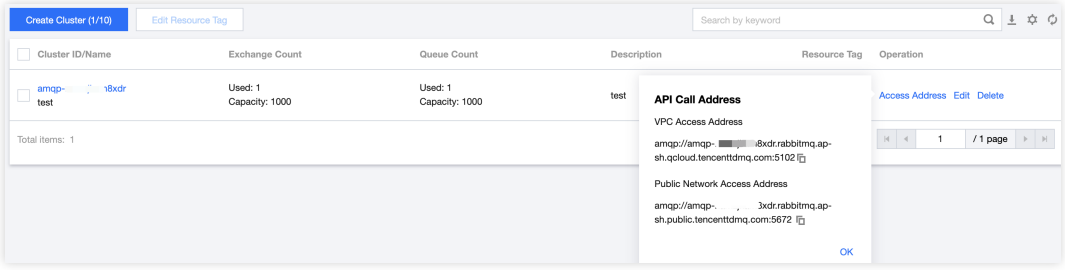
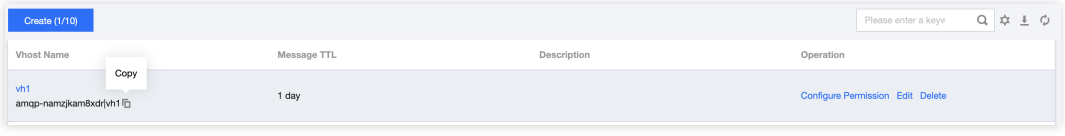
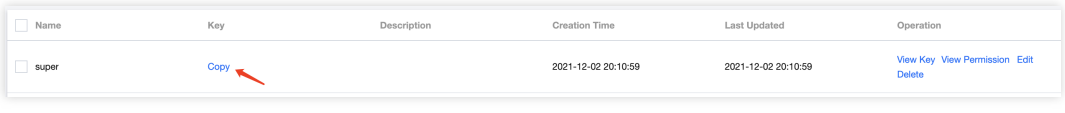
Compile and run `MessageProducer.java` .

```
import com.rabbitmq.client.Channel;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.ConnectionFactory;
import com.tencent.tdmq.demo.cloud.Constant;

/**
 * Message producer
 */
public class MessageProducer {

    /**
     * Exchange name
     */
    private static final String EXCHANGE_NAME = "exchange_name";

    public static void main(String[] args) throws Exception {
        // Connection factory
        ConnectionFactory factory = new ConnectionFactory();
        // Set the service address (replace with the access point address copied in the
        console)
        factory.setUri("amqp://***");
        // Set vhost (replace with the vhost name copied in the console)
        factory.setVirtualHost(VHOST_NAME);
        // Set the username (use the role name in the permission configuration of the vhost)
        factory.setUsername(USERNAME);
        // Set the password (use the role key)
        factory.setPassword("eyJh****");
        // Get the connection address and establish the channel
        try (Connection connection = factory.newConnection(); Channel channel = connection.createChannel()) {
            // Bind the message exchange (`EXCHANGE_NAME` must exist in the TDMQ for RabbitMQ console, and the exchange type must be the same as that in the console)
            channel.exchangeDeclare(EXCHANGE_NAME, "fanout");
            for (int i = 0; i < 10; i++) {
                String message = "this is rabbitmq message " + i;
                // Publish a message to the exchange, which will automatically deliver the message to the corresponding queue
                channel.basicPublish(EXCHANGE_NAME, "", null, message.getBytes());
                System.out.println(" [producer] Sent '" + message + "'");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Parameter	Description
EXCHANGE_NAME	Exchange name, which can be obtained from the exchange list in the console.
factory.setUri	<p>Cluster access address, which can be obtained in the console by clicking <b>Get Access Address</b> in the <b>Operation</b> column on the <b>Cluster</b> page.</p> 
factory.setVirtualHost	<p>Vhost name in the format of "<b>cluster ID +   + vhost name</b>".</p> 
factory.setUsername	Role name, which can be copied on the <b>Role Management</b> page.
factory.setPassword	<p>Role token, which can be copied in the <b>Token</b> column on the <b>Role Management</b> page.</p> 

### Step 3. Consume messages

Compile and run `MessageConsumer.java`.

```
import com.rabbitmq.client.AMQP;
import com.rabbitmq.client.Channel;
import com.rabbitmq.client.Connection;
import com.rabbitmq.client.ConnectionFactory;
import com.rabbitmq.client.DefaultConsumer;
import com.rabbitmq.client.Envelope;
import com.tencent.tdmq.demo.cloud.Constant;

import java.io.IOException;
import java.nio.charset.StandardCharsets;
```

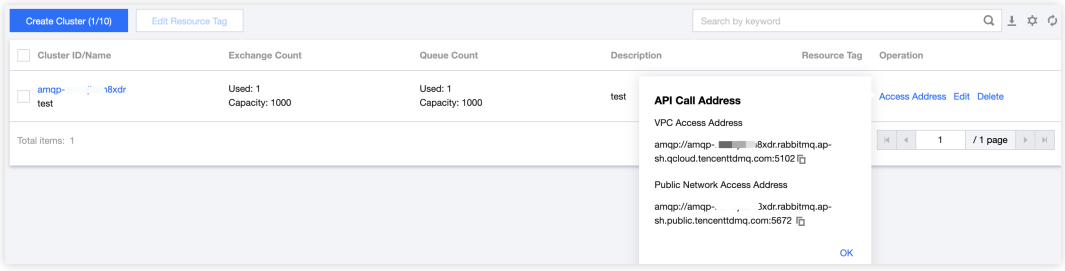
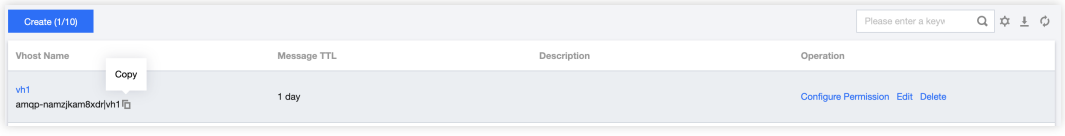
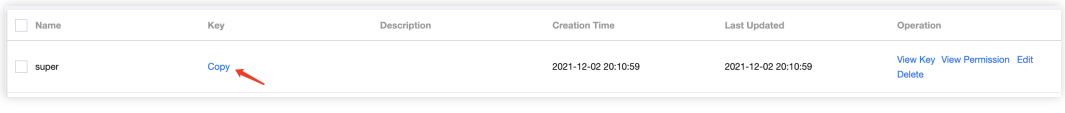
```
/**
 * Message consumer
 */
public class MessageConsumer1 {

    /**
     * Queue name
     */
    public static final String QUEUE_NAME = "queue_name";

    /**
     * Exchange name
     */
    private static final String EXCHANGE_NAME = "exchange_name";

    public static void main(String[] args) throws Exception {
        // Connection factory
        ConnectionFactory factory = new ConnectionFactory();
        // Set the service address (replace with the access point address copied in the
        console)
        factory.setUri("amqp://***");
        // Set vhost (replace with the vhost name copied in the console)
        factory.setVirtualHost(VHOST_NAME);
        // Set the username (use the role name in the permission configuration of the v
        ost)
        factory.setUsername(USERNAME);
        // Set the password (use the role key)
        factory.setPassword("eyJh****");
        // Get the connection address
        Connection connection = factory.newConnection();
        // Establish a channel
        Channel channel = connection.createChannel();
        // Bind the message exchange
        channel.exchangeDeclare(EXCHANGE_NAME, "fanout");
        // Declare the queue message
        channel.queueDeclare(QUEUE_NAME, true, false, false, null);
        // Bind the message exchange (`EXCHANGE_NAME` must exist in the TDMQ for RabbitM
        Q console, and the exchange type must be the same as that in the console)
        channel.queueBind(QUEUE_NAME, EXCHANGE_NAME, "");
        System.out.println(" [Consumer1] Waiting for messages.");
        // Subscribe to the message
        channel.basicConsume(QUEUE_NAME, false, "ConsumerTag", new DefaultConsumer(chann
        el) {
            @Override
            public void handleDelivery(String consumerTag, Envelope envelope,
            AMQP.BasicProperties properties, byte[] body)
```

```
throws IOException {
    // Received message for business logic processing
    System.out.println("Received: " + new String(body, StandardCharsets.UTF_8) + ",
deliveryTag: " + envelope.getDeliveryTag() + ", messageId: " + properties.getMes
sageId());
    channel.basicAck(envelope.getDeliveryTag(), false);
}
});
}
```

Parameter	Description
QUEUE_NAME	Queue name, which can be obtained from the queue list in the console.
EXCHANGE_NAME	Exchange name, which can be obtained from the exchange list in the console.
factory.setUri	<p>Cluster access address, which can be obtained in the console by clicking <b>Get Access Address</b> in the <b>Operation</b> column on the <b>Cluster</b> page.</p> 
factory.setVirtualHost	<p>Vhost name in the format of "cluster ID +   + vhost name".</p> 
factory.setUsername	Role name, which can be copied on the <b>Role Management</b> page.
factory.setPassword	<p>Role token, which can be copied in the <b>Token</b> column on the <b>Role Management</b> page.</p> 

### Step 4. Query messages



To check whether messages are sent to TDMQ for RabbitMQ successfully, view the connected consumer status on the [Cluster](#) > [Queue](#) page in the console.

Note :

Above is a sample based on the pub/sub pattern of RabbitMQ. For more samples, see [Demo](#) or [RabbitMQ Tutorials](#).