TDMQ for RabbitMQ

Getting Started

Product Documentation

Contents

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

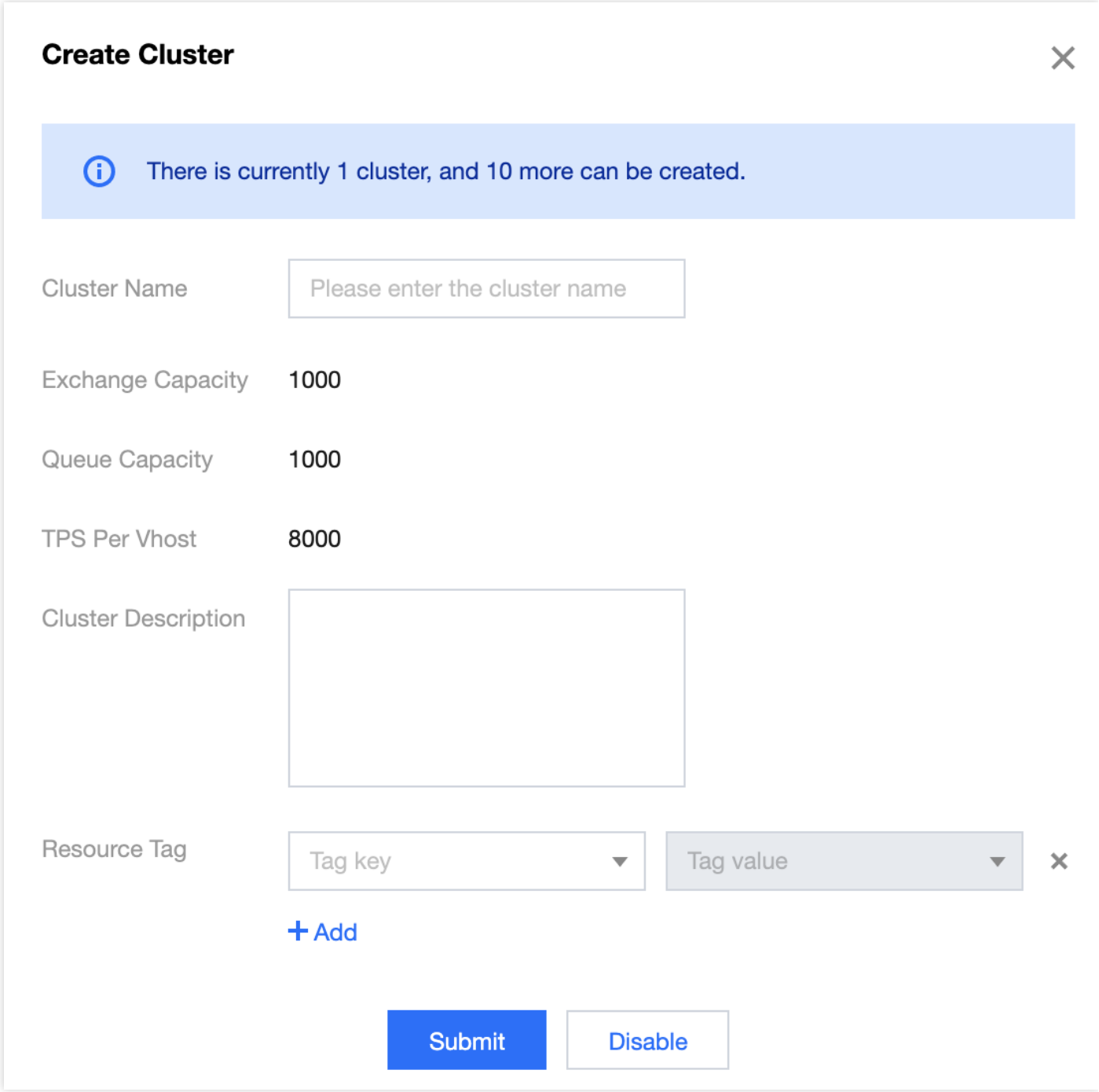
- You have signed up for a Tencent Cloud account.

Directions

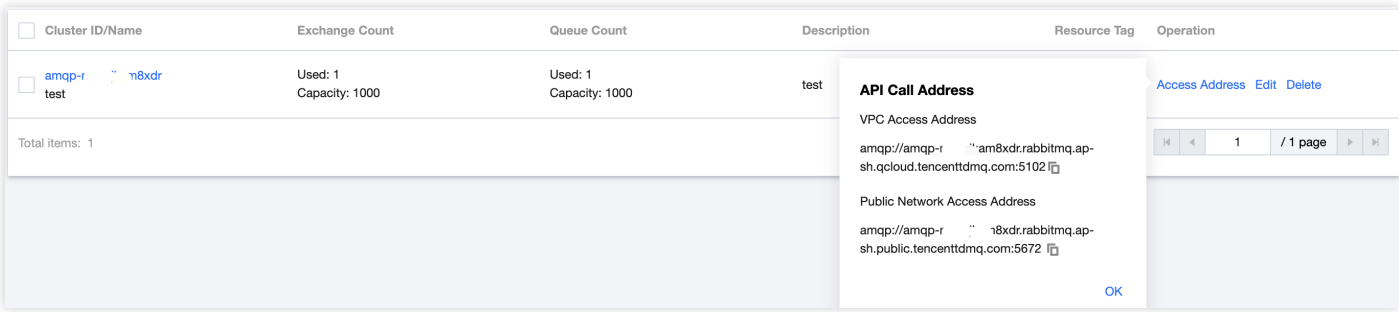
Step 1. Create a cluster

1. Log in to the TDMQ console, enter the **Cluster** page, and select the target region.
2. Click **Create Cluster** and enter the cluster name and description.

3. Click **Submit**.



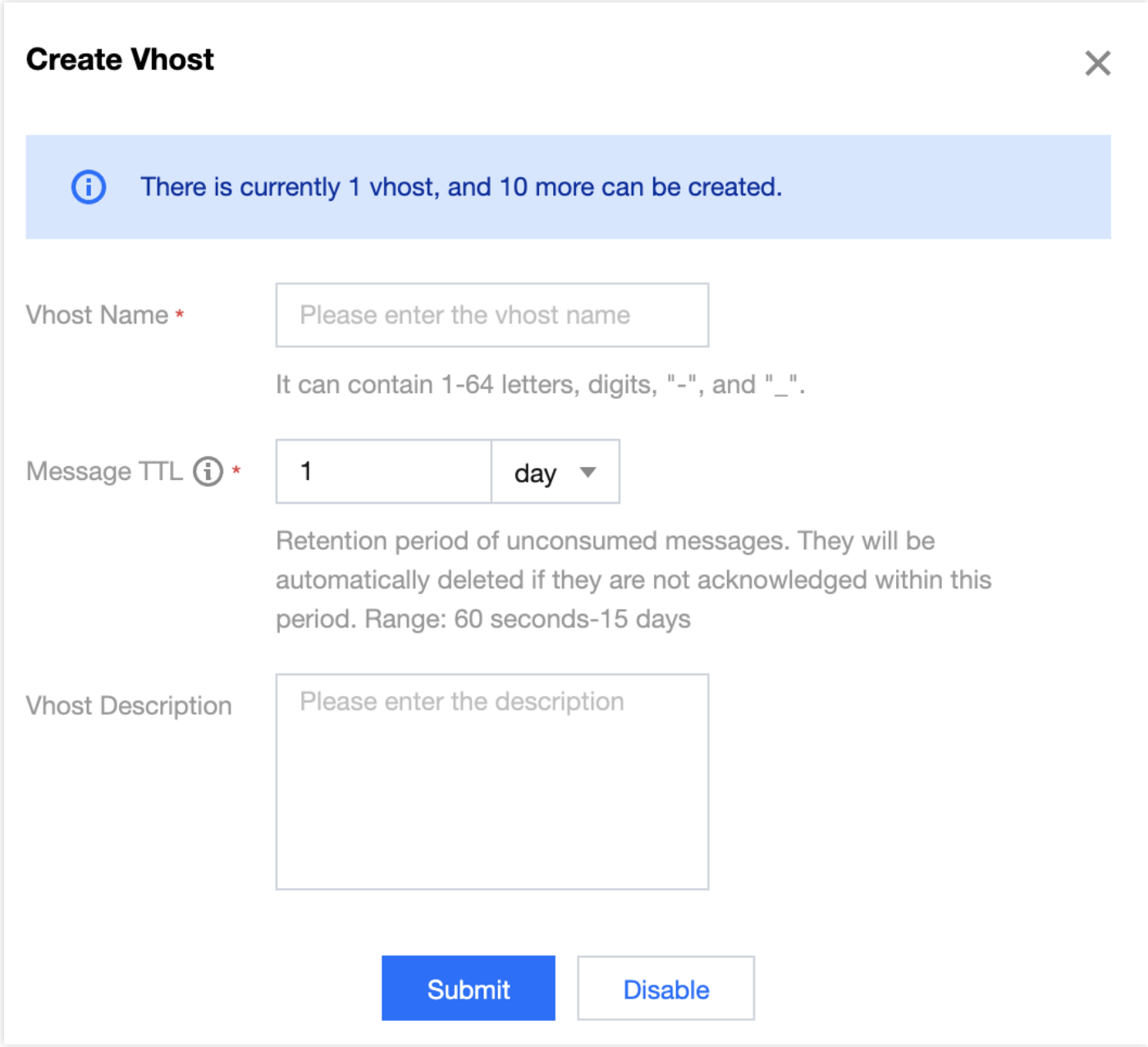
4. On the **Cluster** list page, click **Access Address** in the **Operation** column of the cluster you just created to get the connection information of the server.



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.

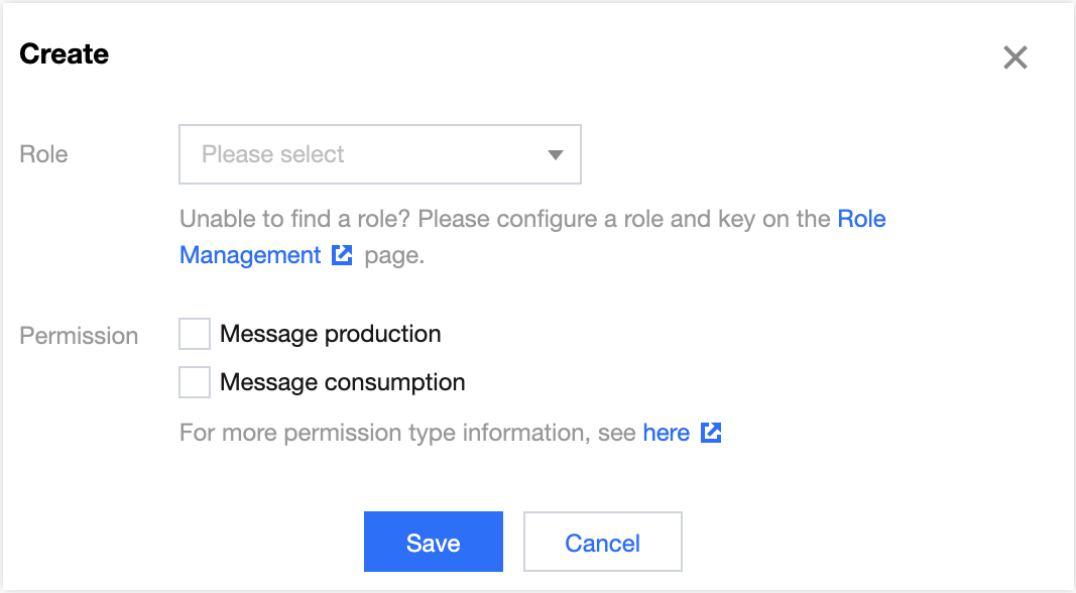
3. Click **Submit**.



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

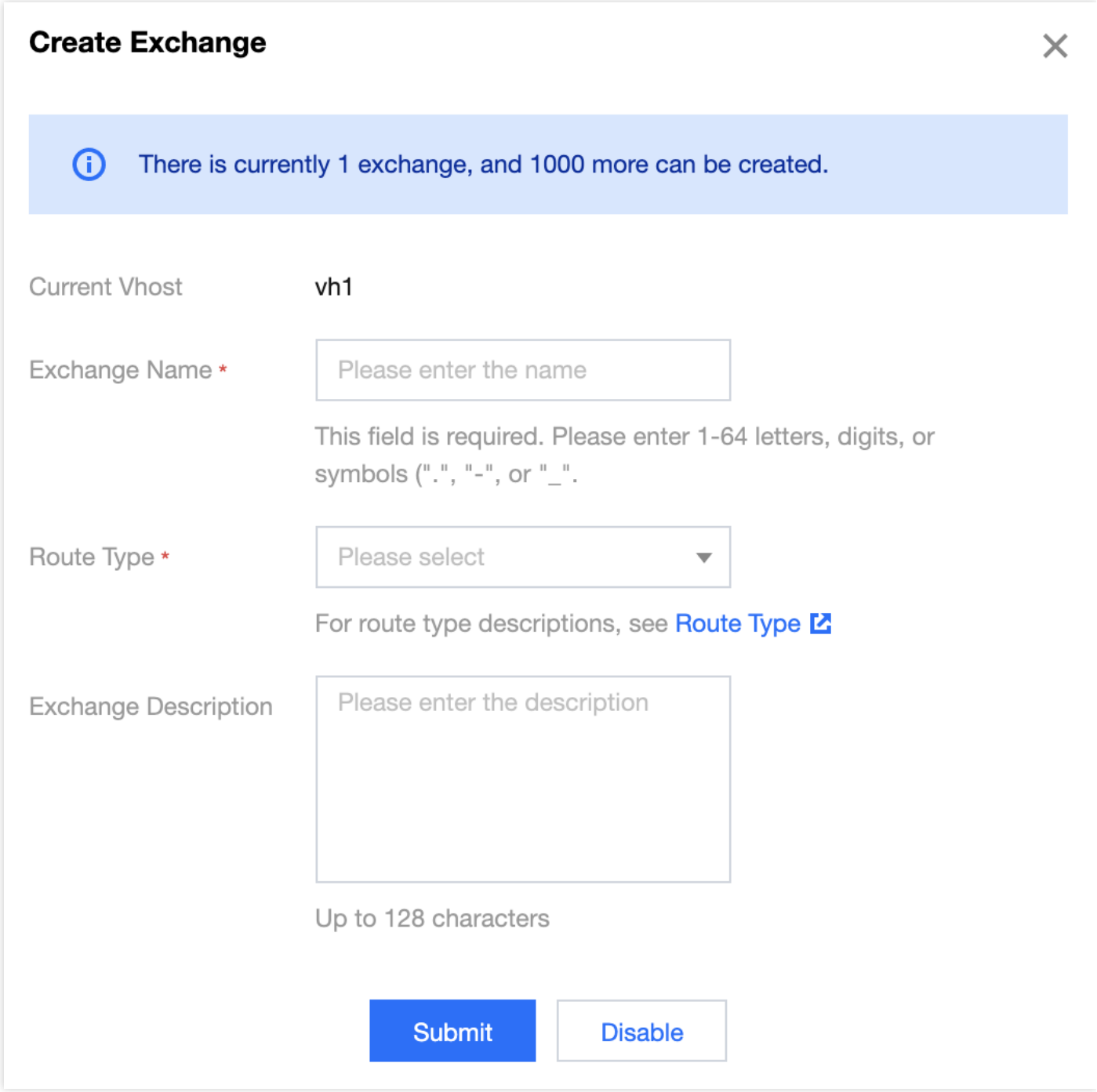


you just created.

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**.

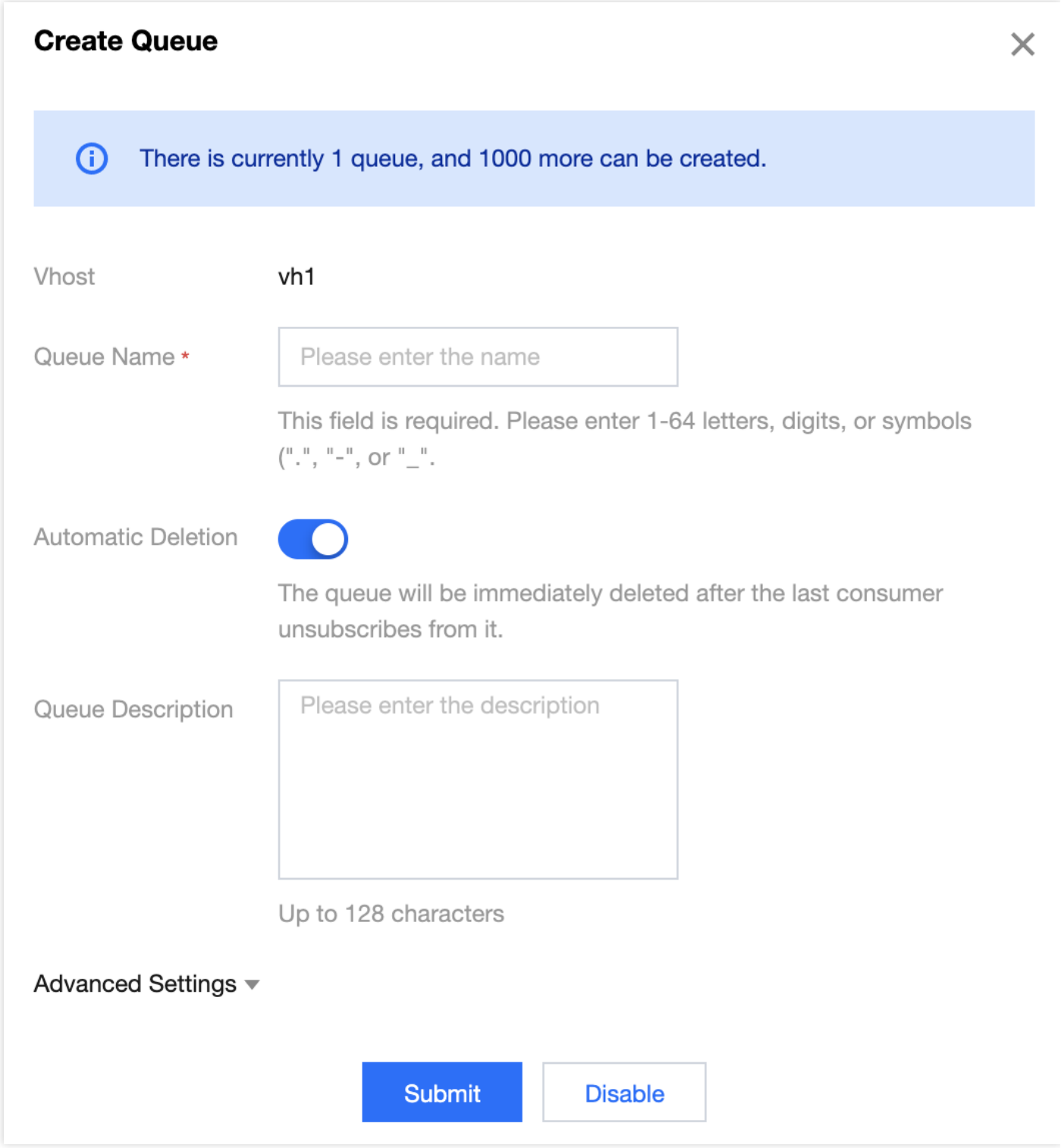
3. Click **Submit**.



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**.



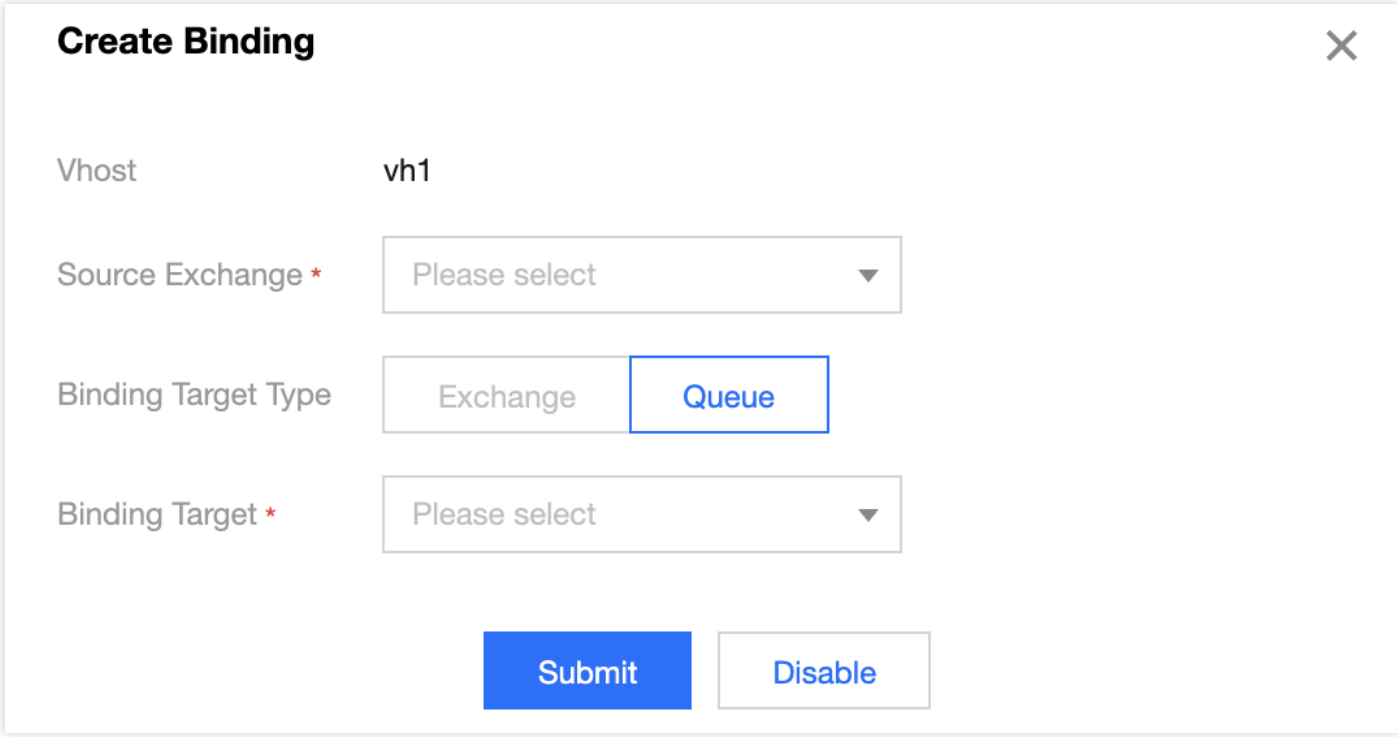
Step 6. Create a routing

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**.



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

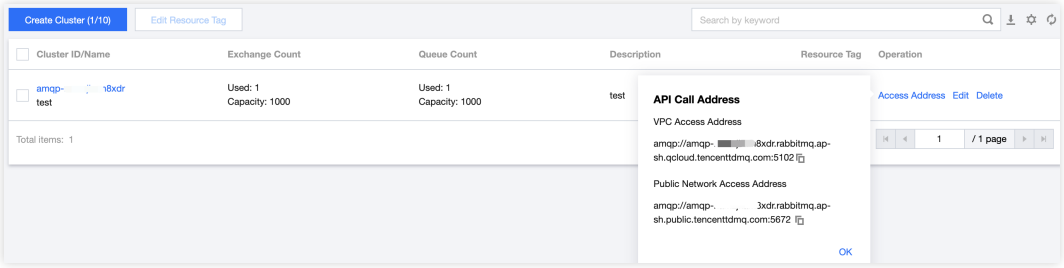
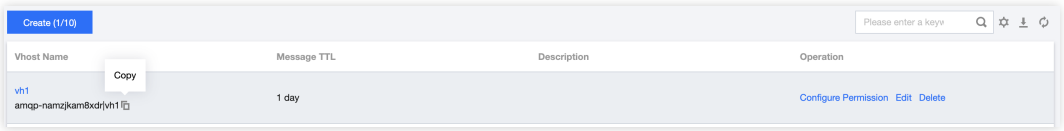
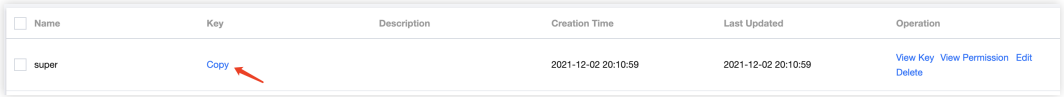
Compile and run `MessageProducer.java` .

```
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 vh
ost)
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 = connecti
on.createChannel()) {
// 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.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 messa
ge 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	Cluster access address, which can be obtained in the console by clicking **Get Access Address** in the **Operation** column on the **Cluster** page. 
factory.setVirtualHost	Vhost name in the format of **"cluster ID + | + vhost name"**. 
factory.setUsername	Role name, which can be copied on the **Role Management** page.
factory.setPassword	Role token, which can be copied in the **Token** column on the **Role Management** page. 

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;
```



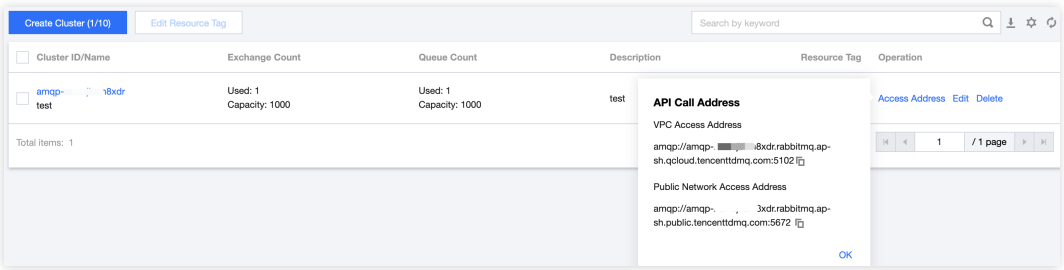
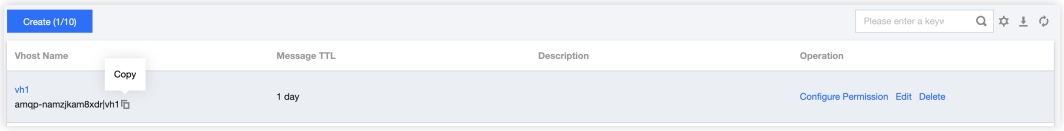
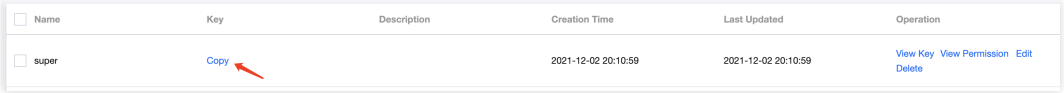
```
/**
* Message consumer
*/
public class MessageConsumer1 {

/**
* 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 vh
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	Cluster access address, which can be obtained in the console by clicking **Get Access Address** in the **Operation** column on the **Cluster** page. 
factory.setVirtualHost	Vhost name in the format of **"cluster ID + | + vhost name"**. 
factory.setUsername	Role name, which can be copied on the **Role Management** page.
factory.setPassword	Role token, which can be copied in the **Token** column on the **Role Management** page. 

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.