Message Queue

Quick Start

Quick Start

Quick start for primary accounts

This topic describes the complete process, from activating the MQ service and creating the MQ resources to sending and receiving messages using the SDK from the primary account perspective, helping you get started with MQ in the simplest and quickest way.

Step 1: Activate the MQ service

To activate the MQ service, follow the steps below:

Log on to the Alibaba Cloud website. Choose **Products** > **Middleware** > **Message Queue**. Then click **MQ**.

On the MQ homepage, click **Buy Now** to enter the MQ activation page, and activate the service as prompted.

If you have already activated the MQ service, log on to the MQ console directly.

Step 2: Create resources

Resource types

When a new application accesses MQ, you need to create the following MQ resources for the application:

Instance: As a VM resource of MQ, the instance stores the topics and group IDs of messages.

Topic: In the message system of MQ, the producer sends a message to a specified topic and the consumer subscribes to the topic to obtain and consume the message.

Group ID: It is used to identify the consumer or producer of the message.

Alibaba Cloud AccessKey: is used to authenticate the account to send or receive messages.

Note: When an instance is deleted, all topics and group IDs of the instance are cleared within 10 minutes. When a single topic or group ID is deleted, other resources are not affected.

Network access

Network access of MQ is constrained as follows:

A topic can be accessed only by a producer or consumer corresponding to the group on the same instance in the same region as the topic.

For example, when a topic is created in **Instance A** in the region **China North 2 (Beijing)**, the topic can be accessed only by the producer and consumer corresponding to the group ID created in **Instance A** in the region **China North 2 (Beijing)**.

If you want to perform a test or use MQ on a local server (other than an Alibaba Cloud ECS instance), you can create both the topic and group ID in an instance in the region Internet. Producers and consumers can be deployed on a local server or an ECS instance in any region, provided that the local server or ECS instance can access the Internet. However, a topic created in one instance cannot be accessed from another instance.

For more information about regions, see Regions and zones in the ECS documentation.

Select a region

Log on to the MQ console.

Select a region, for example, **Internet**, in which you want to create a resource.

Note: Currently, HTTP-based instances can only be deployed in the region China East 1 (Hangzhou).

Create instances

In the left-side navigation pane, choose **Instances**.

On the **Instances** page, click **Create Instance**.

In the **Create Instance** dialog box, select the instance type, enter a name and description for the instance, and click **OK**.

Create topics

Topic is the first-level identifier for classifying messages in MQ. For example, you can create a topic named *Topic_Trade* for transaction messages.

In the left-side navigation pane, choose **Topics**.

On the **Topics** page, select the instance you created.

Click **Create Topic**.

In the **Create Topic** dialog box, enter a name for the topic in the **Topic** field.

Note: The topic name must be unique in the same instance.

Select a value for **Message Type**. This message type defines which type of messages this topic sends and receives.

Note: Currently, HTTP supports sending and receiving only normal messages. To send and receive messages over HTTP, select **Normal messages**.

The following describes the available message types:

- [Normal messages](~~96359~~): Normal messages do not have any special features, differentiated from the other types of messages.

- [Transactional messages](~~43348~~): MQ provides the distributed transaction function which is similar to X/Open XA. You can achieve transaction consistency with transactional messages.

- [Scheduled and delayed messages](~~43349~~): Messages reach the consumer at a specific time point or after a certain period when they are sent.

- [Partitionally ordered messages](~~49319~~): Messages are partitioned by sharding keys to improve overall concurrency and performance. Messages in the same partition are produced and consumed in strict FIFO order.

- [Globally ordered messages](~~49319~~): All messages are produced and consumed in strict FIFO order.

Note: We recommend that you create separate topics to send messages of different types. For example, create Topic A for normal messages, Topic B for transactional messages, and Topic C for scheduled and delayed messages.

1. In the **Description**field, enter remarks for the topic. Then click **OK**. The created topic is displayed in the topic list.

Create group IDs

After you have created an instance and a topic, you need to create a group ID for the message consumer (or producer).

Note: A group ID is required for consumers but is optional for producers.

To create a group ID, follow the steps below:

1. In the left-side navigation pane, choose **Groups**.
2. On the **Groups** page, select the instance you created.
3. Click the TCP or HTTP tab based on the protocol you need.

Note: The group ID of TCP-based instances cannot be used for HTTP-based instances and vice versa. You need to create the group IDs for TCP-based instances and for HTTP-based instances separately.

4. Click **Create Group ID**.
5. In the **Create Group ID** dialog box, enter a group ID and description, and click **OK**.

Note:

- The group ID must be unique in the same instance.
- Group IDs and topics implement N:N mapping. A consumer can subscribe to multiple topics and a topic can be subscribed by multiple consumers. A producer can send messages to multiple topics and a topic can receive messages from multiple producers.

Create Alibaba Cloud AccessKeys

When using SDKs or calling APIs to send and subscribe to messages, you not only need to specify the topic and group ID, but also need to enter your identity verification information that you have created in the RAM console, that is, AccessKey. An AccessKey consists of an AccessKeyId and an AcessKeySecret.

For more information about how to create AccessKeys, see Create AccessKey.

Step 3: Obtain endpoints

After creating resources in the console, you need to obtain the endpoint of the instance or region in the console. To access services in an instance or a region when sending or receiving messages, you need to configure the endpoints for the producer and consumer. The endpoints vary with the protocol. The following describes the endpoints when different protocols are used:

TCP: The endpoint displayed in the console is the endpoint of a specific instance in the region. Different instances in the same region have different endpoints.

HTTP: The endpoint displayed in the console is the endpoint of a region, instead of a specific instance. You need to configure an ID for the instance when sending and receiving messages.

To view the endpoint, perform the following operations:

In the left-side navigation pane, choose **Instances**.

On the **Instances** page, select the instance you created.

The **Instance Information** tab page is displayed by default. In the **Endpoint Information** area, you can view the endpoints of the TCP- and HTTP-based instances.

In the Endpoint area of the corresponding protocol, click **Copy**.

As for TCP-based endpoints, you can click **Sample Code** to view how to set endpoints in different programming languages.

Once the preceding preparation is done, you can run the sample code and use MQ to send and subscribe to messages.

Step 4: Send messages

You can send messages in the console, or by using SDKs or calling APIs.

To quickly verify the availability of the topics, you can send messages in the console.

In a production environment, the SDK/API method is recommended for sending messages.

Send messages in the console

In the left-side navigation pane, choose **Topics**.

On the **Topics** page, locate the row that contains the topic you created and click **Send** in the **Actions** column.

In the **Send Message** dialog box, enter the message content in the **Message Body** field and click **OK**.

The console returns a success prompt message and the corresponding message ID once the message is sent.

Send messages by using SDKs or calling APIs

In a production environment, the SDK/API method is recommended for sending messages. This topic describes how to send messages by using a Java SDK over TCP.

Note:

If you want to use the C/C ++ and .NET SDKs to send messages over TCP, see Send and receive normal messages through C/C++ SDK and Send and receive normal messages through .NET SDK.

If you want to send messages over HTTP, see SDK guide (HTTP) to obtain the SDK and sample code for sending messages.

Send messages by using the Java SDK over TCP

1. Use either of the following methods to add the dependency of the Java SDK:

- Use Maven:

```
<dependency>
<groupId>com.aliyun.openservices</groupId>
<artifactId>ons-client</artifactId>
<version>"XXX"</version>
// Enter the latest Java SDK version.
</dependency>
```

For more information about the version of the latest Java SDK, see [Release Notes](~~60953~~).

Download a dependency JAR package:

For more information about the download URL of the latest Java SDK, see Release Notes.

Set related parameters and run the sample code according to the following instructions:

```
import com.aliyun.openservices.ons.api.Message;
import com.aliyun.openservices.ons.api.Producer;
import com.aliyun.openservices.ons.api.SendResult;
import com.aliyun.openservices.ons.api.ONSFactory;
import com.aliyun.openservices.ons.api.PropertyKeyConst;

import java.util.Properties;

public class ProducerTest {
public static void main(String[] args) {
Properties properties = new Properties();
// The group ID you created in the console.
properties.put(PropertyKeyConst.GROUP_ID, "XXX");
// The AccessKeyId you created in the Alibaba Cloud console.
properties.put(PropertyKeyConst.AccessKey,"XXX");
// The AccessKeySecret you created in the Alibaba Cloud console.
properties.put(PropertyKeyConst.SecretKey, "XXX");
// Set the TCP endpoint: Go to the Instances page in the MQ console, select the target instance, and
view the endpoint in the Endpoint Information area.
properties.put(PropertyKeyConst.NAMESRV_ADDR,"XXX");

Producer producer = ONSFactory.createProducer(properties);
//Before sending a message, call the start() method once to start the producer.
producer.start();

//Send messages cyclically.
while(true){
Message msg = new Message( //
// The topic you created in the console. This is the name of the topic to which the message belongs.
"TopicTestMQ",
// The message tag.
// The message tag is similar to a tag in Gmail, and is used for consumers to filter messages on the MQ
broker by setting filtering conditions.
"TagA",
// The message body.
// It may be any data in binary form, and MQ makes no intervention.
// The producer and consumer must negotiate the consistent serialization and deserialization methods.
"Hello MQ".getBytes());
// Set a key service property representing the message, that is, the message key, and try to keep it
globally unique. In this way, you can query and resend the message through the MQ console when you
cannot receive the message.
// Note: Messages can still be sent and received if you do not set this attribute.
msg.setKey("ORDERID_100");
// The message is sent if no exception is thrown.
// Print the message ID to facilitate querying the message sending status.
SendResult sendResult = producer.send(msg);
System.out.println("Send Message success. Message ID is: " + sendResult.getMessageId());
}
```

```
// You can destroy the producer before exiting the application.
// Note: You can choose not to destroy the producer object.
producer.shutdown();
}
}
```

Check whether messages are sent

Once a message is sent, you can check its sending status in the console by performing the following operations:

In the left-side navigation pane, choose **Message Query**.

On the **Message Query** page, click the **By Message ID** tab.

In the search box, enter the message ID returned after the message is sent, and click **Search** to query the sending status of the message.

Storage Time indicates the time period when the MQ broker stores the message. If the message can be queried out, the message has been sent to the broker.

Note: This step demonstrates the scenario where MQ is used for the first time, when the consumer has not been started yet. Therefore, no consumption data is displayed in the message status information. To start the consumer and subscribe to messages, see step 5. For more information about the message status, see Message Query.

Step 5: Subscribe to messages

Once a message is sent, you need to start the consumer to subscribe to messages. This topic describes how to complete message subscription by using the SDK or calling the API of the desired programming language over the target protocol. The following uses the TCP Java SDK as an example.

Note:

If you want to use the C/C ++ and .NET SDKs to subscribe to messages over TCP, see Subscribe to messages through C/C++ SDK and Subscribe to messages through .NET SDK.

If you want to subscribe to messages over HTTP, see SDK guide (HTTP) to obtain the SDK and sample code for subscribing to messages.

Subscribe to messages by using the TCP Java SDK

You can run the following sample code to start the consumer and test the message subscription function. You must set related parameters correctly according to the instructions. Currently, the console provides sample codes of Java, C++, and .NET.

```
import com.aliyun.openservices.ons.api.Action;
import com.aliyun.openservices.ons.api.ConsumeContext;
import com.aliyun.openservices.ons.api.Consumer;
import com.aliyun.openservices.ons.api.Message;
import com.aliyun.openservices.ons.api.MessageListener;
import com.aliyun.openservices.ons.api.ONSFactory;
import com.aliyun.openservices.ons.api.PropertyKeyConst;

import java.util.Properties;

public class ConsumerTest {
public static void main(String[] args) {
Properties properties = new Properties();
// The group ID you created in the console.
properties.put(PropertyKeyConst.GROUP_ID, "XXX");
// The AccessKeyId you created in the Alibaba Cloud console.
properties.put(PropertyKeyConst.AccessKey, "XXX");
// The AccessKeySecret you created in the Alibaba Cloud console.
properties.put(PropertyKeyConst.SecretKey, "XXX");
// Set the TCP endpoint: Go to the Instances page in the MQ console, select the target instance, and view the
endpoint in the Endpoint Information area.
properties.put(PropertyKeyConst.NAMESRV_ADDR,"XXX");

Consumer consumer = ONSFactory.createConsumer(properties);
consumer.subscribe("TopicTestMQ", "*", new MessageListener() {
public Action consume(Message message, ConsumeContext context) {
System.out.println("Receive: " + message);
return Action.CommitMessage;
}
});
consumer.start();
System.out.println("Consumer Started");
}
}
```

Check whether message subscription is successful

Once the preceding steps have been completed, you can check whether the consumer has been started in the console, that is, whether the message subscription is successful.

In the left-side navigation pane, choose **Groups**.

Locate the row that contains the group ID of the target consumer and click **Consumer Status** in the **Actions** column.

If **Online** is displayed, the consumer has been started. If **Offline** is displayed, the consumer has not been started or has failed to start.

When all the preceding steps are completed, you have accessed the MQ service and can use MQ for sending and receiving messages.

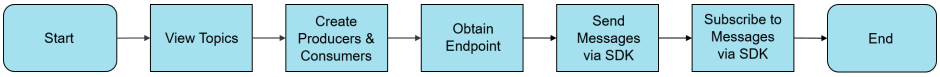
Quick start for sub-accounts

If a primary account grants a RAM sub-account permissions to operate topic resources on MQ instances, the sub-account cannot directly use the group ID created by the primary account. In this case, log on to the MQ console, locate the instances and topics whose operation permissions have been granted to the RAM sub-account, and create another group ID. Then, you can use an SDK to send and receive messages.

Note:

> For more information about RAM primary accounts and sub-accounts, see Grant permissions to RAM sub-accounts.
>
> Currently, HTTP is supported only in China East 1 (Hangzhou).

Quick start flowchart



Prerequisites

Your RAM sub-account already meets the following requirements:

- The primary account has created an AccessKey for the RAM sub-account.

If not, use the primary account to create an AccessKey for the RAM sub-account by following the instructions provided in Create a RAM user.

You need the AccessKey for authentication when using SDKs or calling APIs to send and receive messages.

> The primary account has granted the RAM sub-account the permission to operate topic resources.

If not, grant permissions to the RAM sub-account by following the instructions provided in Grant permissions to RAM sub-accounts.

Step 1: View the target instances and topics

To view the instances and topics whose resource operation permissions have been granted to the RAM sub-account, perform the following operations. These instances and topics can be used for sending and receiving messages.

Log on to the RAM console.

In the left-side navigation pane, choose **Message Queue** to go to the MQ console.

If your RAM sub-account already logs on, click the MQ console directly.

In the left-side navigation pane, choose **Instances** to view the basic information about the instances whose resource operation permissions have been granted to the RAM sub-account.

In the left-side navigation pane, choose **Topics** to view the topics whose resource operation permissions have been granted to the RAM sub-account.

Step 2: Create group IDs

Next, you need to create a group ID for the message consumers (or producers).

Note: A group ID is required for consumers but is optional for producers.

To create a group ID, perform the following operations:

In the left-side navigation pane, choose **Groups**.

On the **Groups** page, select the instance you created.

Click the TCP or HTTP tab based on the protocol you need.

Note: The group ID of TCP-based instances cannot be used for HTTP-based instances and vice versa. You need to create the group IDs for TCP-based instances and for HTTP-based instances separately.

Click **Create Group ID**.

In the **Create Group ID** dialog box, enter a group ID and description, and click **OK**.

Note:

The group ID must be unique in the same instance.

Group IDs and topics implement N:N mapping. A consumer can subscribe to multiple topics and a topic can be subscribed by multiple consumers. A producer can send messages to multiple topics and a topic can receive messages from multiple producers.

Step 3: Obtain endpoints

After creating resources in the console, you need to obtain the endpoint of the instance or region in the console. To access services in an instance or a region when sending or receiving messages, you need to configure the endpoints for the producer and consumer. The endpoints vary with the protocol. The following describes the endpoints when different protocols are used:

TCP: The endpoint displayed in the console is the endpoint of a specific instance in the region. Different instances in the same region have different endpoints.

HTTP: The endpoint displayed in the console is the endpoint of a region, instead of a specific instance. You need to configure an ID for the instance when sending and receiving messages.

To view the endpoint, perform the following operations:

In the left-side navigation pane, choose **Instances**.

On the **Instances** page, select the instance you created.

The **Instance Information** tab page is displayed by default. In the **Endpoint Information** area, you can view the endpoints of the TCP- and HTTP-based instances.

In the Endpoint area of the corresponding protocol, click **Copy**.

You can click **Sample Code** to view how to set endpoints in different programming languages.

Once the preceding preparation is done, you can run the sample code and use MQ to send and subscribe to messages.

Step 4: Send messages

You can send messages in the console, or by using SDKs or calling APIs.

To quickly verify the availability of the topics, you can send messages in the console.

In a production environment, the SDK/API method is recommended for sending messages.

Send messages in the console

In the left-side navigation pane, choose **Topics**.

On the **Topics** page, locate the row that contains the topic you created and click **Send** in the **Actions** column.

In the **Send Message** dialog box, enter the message content in the **Message Body** field and click **OK**.

The console returns a success prompt message and the corresponding message ID once the message is sent.

Send messages by using SDKs or calling APIs

In a production environment, the SDK/API method is recommended for sending messages. This topic describes how to send messages by using a Java SDK over TCP.

Note:

If you want to use the C/C ++ and .NET SDKs to send messages over TCP, see **Send and receive normal messages** through C/C++ SDK and **Send and receive normal messages** through .NET SDK.

If you want to send messages over HTTP, see **SDK guide (HTTP)** to obtain the SDK and sample code for sending messages.

Send messages by using the Java SDK over TCP

Use either of the following methods to add the dependency of the Java SDK:

Use Maven:

```
<dependency>
    <groupId>com.aliyun.openservices</groupId>
    <artifactId>ons-client</artifactId>
    <version>"XXX"</version>
     // Enter the latest Java SDK version.
 </dependency>
```

For more information about the version of the latest Java SDK, see [Release Notes](~~60953~~).

Download a dependency JAR package:

For more information about the download URL of the latest Java SDK, see Release Notes.

Set related parameters and run the sample code according to the following instructions:

```
import com.aliyun.openservices.ons.api.Message;
 import com.aliyun.openservices.ons.api.Producer;
 import com.aliyun.openservices.ons.api.SendResult;
 import com.aliyun.openservices.ons.api.ONSFactory;
 import com.aliyun.openservices.ons.api.PropertyKeyConst;

 import java.util.Properties;

 public class ProducerTest {
    public static void main(String[] args) {
        Properties properties = new Properties();
        // The group ID you created in the console.
        properties.put(PropertyKeyConst.GROUP_ID, "XXX");
        // The AccessKeyId used for RAM sub-account authentication, which is created by the primary
account.
        properties.put(PropertyKeyConst.AccessKey,"XXX");
        // The AccessKeySecret used for RAM sub-account authentication, which is created by the primary
account.
        properties.put(PropertyKeyConst.SecretKey, "XXX");
        // Set the TCP endpoint: Go to the Instances page in the MQ console, select the target instance, and
view the endpoint in the Endpoint Information area.
        properties.put(PropertyKeyConst.NAMESRV_ADDR,"XXX");

        Producer producer = ONSFactory.createProducer(properties);
        //Before sending a message, call the start() method once to start the producer.
        producer.start();

        //Send messages cyclically.
        while(true){
            Message msg = new Message( //
                // The topic you created in the console. This is the name of the topic to which the message
belongs.
```

```
                "TopicTestMQ",
                // The message tag.
                // The message tag is similar to a tag in Gmail, and is used for consumers to filter messages on
the MQ broker by setting filtering conditions.
                "TagA",
                // The message body.
                // It may be any data in binary form, and MQ makes no intervention.
                // The producer and consumer must negotiate the consistent serialization and deserialization
methods.
                "Hello MQ".getBytes());
            // Set a key service property representing the message, that is, the message key, and try to keep
it globally unique. In this way, you can query and resend the message in the MQ console when you
cannot receive the message.
            // Note: Messages can still be sent and received if you do not set this attribute.
            msg.setKey("ORDERID_100");
            // The message is sent if no exception is thrown.
            // Print the message ID to facilitate querying the message sending status.
            SendResult sendResult = producer.send(msg);
            System.out.println("Send Message success. Message ID is: " + sendResult.getMessageId());
        }

        // You can destroy the producer before exiting the application.
        // Note: You can choose not to destroy the producer object.
        producer.shutdown();
    }
}
```

Check whether messages are sent

Once a message is sent, you can check its sending status in the console by performing the following operations:

In the left-side navigation pane, choose **Message Query**.

On the **Message Query** page, click the **By Message ID** tab.

In the search box, enter the message ID returned after the message is sent, and click **Search** to query the sending status of the message.

Storage Time indicates the time period when the MQ broker stores the message. If the message can be queried out, the message has been sent to the broker.

Note: This step demonstrates the scenario where MQ is used for the first time, when the consumer has not been started yet. Therefore, no consumption data is displayed in the message status information. To start the consumer and subscribe to messages, see step 5. For more information about the message status, see Message Query.

Step 5: Subscribe to messages

Once a message is sent, you need to start the consumer to subscribe to messages. This topic describes how to complete message subscription by using the SDK or calling the API of the desired programming language over the target protocol. The following uses the TCP Java SDK as an example.

Note:

If you want to use the C/C ++ and .NET SDKs to subscribe to messages over TCP, see Subscribe to messages through C/C++ SDK and Subscribe to messages through .NET SDK.

If you want to subscribe to messages over HTTP, see SDK guide (HTTP) to obtain the SDK and sample code for subscribing to messages.

Subscribe to messages by using the TCP Java SDK

You can run the following sample code to start the consumer and test the message subscription function. You must set related parameters correctly according to the instructions. Currently, the console provides sample codes of Java, C++, and .NET.

```
import com.aliyun.openservices.ons.api.Action;
import com.aliyun.openservices.ons.api.ConsumeContext;
import com.aliyun.openservices.ons.api.Consumer;
import com.aliyun.openservices.ons.api.Message;
import com.aliyun.openservices.ons.api.MessageListener;
import com.aliyun.openservices.ons.api.ONSFactory;
import com.aliyun.openservices.ons.api.PropertyKeyConst;

import java.util.Properties;

public class ConsumerTest {
    public static void main(String[] args) {
        Properties properties = new Properties();
        // The group ID you created in the console.
        properties.put(PropertyKeyConst.GROUP_ID, "XXX");
        // The AccessKeyId used for RAM sub-account authentication, which is created by the primary account.
        properties.put(PropertyKeyConst.AccessKey, "XXX");
        // The AccessKeySecret used for RAM sub-account authentication, which is created by the primary account.
        properties.put(PropertyKeyConst.SecretKey, "XXX");
        // Set the TCP endpoint: Go to the Instances page in the MQ console, select the target instance, and view the
endpoint in the Endpoint Information area.
        properties.put(PropertyKeyConst.NAMESRV_ADDR,"XXX");

        Consumer consumer = ONSFactory.createConsumer(properties);
        consumer.subscribe("TopicTestMQ", "*", new MessageListener() {
            public Action consume(Message message, ConsumeContext context) {
                System.out.println("Receive: " + message);
                return Action.CommitMessage;
            }
```

```
        });
        consumer.start();
        System.out.println("Consumer Started");
    }
}
```

Check whether message subscription is successful

Once the preceding steps have been completed, you can check whether the consumer has been started in the console, that is, whether the message subscription is successful.

1. In the left-side navigation pane, choose **Groups**.
2. Locate the row that contains the group ID of the target consumer and click **Consumer Status** in the **Actions** column.

If **Online** is displayed, the consumer has been started. If **Offline** is displayed, the consumer has not been started or has failed to start.

When all the preceding steps are completed, you have accessed the MQ service and can use MQ for sending and receiving messages.