Message Queue for Apache RocketMQ

Access control

Access control

RAM sub-account authorization

MQ allows a cloud account (primary account) to authorize RAM users (sub-accounts) to use the topic resources of the cloud account. Authorized RAM users can manage resources in the MQ console and can publish or subscribe to messages by using SDKs.

For the basic concepts of RAM, see Terms of RAM.

For more information about authorization and related terms for user group authorization, see Authorize RAM users.

System authorization policies

MQ currently provides three default authorization policies.

Policy name	Remarks	Description
AliyunMQFullAccess	Permission for managing MQ	Equivalent to the permissions of the primary account, this policy provides the permission to send and receive all types of messages and the permission to operate all functions in the MQ console.
AliyunMQPubOnlyAccess	Permission for publishing MQ messages	RAM users with this permission can use all resources of the primary account to publish messages by using SDKs.
AliyunMQSubOnlyAccess	Permission for subscribing to MQ messages	RAM users with this permission can use all the resources of the primary account to subscribe to messages by using SDKs.

Custom authorization policies

In most cases, the preceding authorization policies provided by MQ are sufficient to meet the service requirements. However, if you have authorization requirements with finer granularity, you can create a custom policy for access control.

MQ resources

The following describes how to name MQ resources and relevant description.

MQ resources	Naming format		Remarks
	With namespace	Without namespace	
Instance	acs:mq:*:*:{instanceId}	acs:mq:*:*:{instanceId}	MQ instances must have the permission mq: OnsInstanceBaseInfo before they can authorize topics.
Topic	acs:mq:*:*:{instanceId}%{topic}	acs:mq:*:*:{topic}	Before authorizing a topic, you must authorize the instance of the topic.

Mapping between MQ resources and actions.

When creating a custom authorization policy, you can select different resources and actions based on the specific product. The following lists the options for MQ resources and actions.

Resource	Action	Description
Instance	mq:OnsInstanceBaseInfo	This permission is used to query basic information of instances. You must authorize the permission mq:OnsInstanceBaseInfo of the instance to a RAM user before you can authorize the topic permissions to the user.
	mq:OnsIntanceUpdate	This permission is used to update the instance.
	mq:OnsIntanceDelete	This permission is used to delete the instance. Perform this operation with caution.
Topic	mq:PUB	This permission is used to publish messages.
	mq:SUB	This permission is used to subscribe to messages.

Examples of common policies

Example 1: Authorize the permissions of a topic under an instance

- This policy is applicable to instances with namespaces.

```
{
"Version": "1",
"Statement": [
{
"Effect": "Allow",
"Action": [
"mq:PUB", //(Optional) Grant the permission for publishing messages.
"mq:SUB", //(Optional) Grant the permission for subscribing to messages.
"mq:OnsInstanceBaseInfo" //(Required) Query the basic information of the instance.
],
"Resource": [
"acs:mq:*:*:{instanceId}", //(Required) Grant the permission of an instance. Enter the ID of your instance in
{instanceId}.
"acs:mq:*:*:{instanceId}%{topic}", //(Required) Grant the permission of a topic in the instance. Enter the topic name
in {topic}.
......
]
}
]
}
```

- This policy is applicable to instances with no namespaces.

```
{
"Version": "1",
"Statement": [
{
"Effect": "Allow",
"Action": [
"mq:PUB", //(Optional) Grant the permission for publishing messages.
"mq:SUB", //(Optional) Grant the permission for subscribing to messages.
"mq:OnsInstanceBaseInfo" //(Required) Query the basic information of the instance.
],
"Resource": [
"acs:mq:*:*:{instanceId}", //(Required) Grant the permission of an instance. Enter the ID of your instance in
{instanceId}.
"acs:mq:*:*:{topic}", //(Required) Grant the permission of a topic in the instance. Enter the topic name in {topic}.
......
]
}
]
}
```

Example 2: Authorize all the permissions of an instance

To grant the permissions for operating all the resources in an instance, set the policy as follows.

- This policy is only applicable to instances with namespaces.

```
{
"Version": "1",
"Statement": [
{
"Effect": "Allow",
"Action": [
"mq:*"
],
"Resource": [
"acs:mq:*:*:{instanceId}*" //Grant permissions of the instance. Enter the ID of your instance in {instanceId}.
]
}
]
}
```

Note: The sample policy is only applicable to instances having no namespaces.

Related documents

For more information about how to create a custom policy, see (Optional) Create a custom policy.

When creating a custom policy, you need to reference to the RAM policy structure and syntax. For more information, see Policy structure and syntax.

For more information about RAM, see What is RAM.

Temporary access authorization

Security Token Service (STS) is responsible for the temporary access authorization of Alibaba Cloud accounts (primary account) and RAM users (sub-account).

Comparison Between RAM and STS

The critial issue that both RAM and STS have resolved is how to securely grant access without leaking AccessKey (AK) of the primary account. Once the AK of the primary account is leaked, there is great risk that others can operate on all the resources of the primary account and steal important information. Using RAM and STS greatly improves management security and flexibility.

RAM provides an access control mechanism that is available permanently. This mechanism divides the primary accounts into many sub-accounts with defferent permissions granted. Even if information about one of the sub-account is leaked, information about the rest sub-accounts is still secure. For better maintenance, the RAM sub-accounts are avaiable permanently.

Instead of offering permanent access permissions like RAM, STS adopts a temporary solution by providing temporary AK and SecurityToken (Token). As a result, STS is often more rigorous and time constraint with less impact even after information leakage.

Cross-account authorization

STS also applies to cross-account authorization. For details, see Cross-account resource access and authorization.

Temporary access authorization

For the prerequisites for using STS, including creating roles, AK and Token, see Getting started and AssumeRole.

Use STS in MQ

Note: STS is supported only by Java SDK 1.7.8.Final or above.

To use STS when sending or receiving messages via the API, fill out the properties below with your AK and Token.

```
Properties properties = new Properties();
......
// The AccessKeyId of STS
properties.put(PropertyKeyConst.AccessKey,"XXX");
// The AccessKeySecret of STS
properties.put(PropertyKeyConst.SecretKey, "XXX");
// The SecurityToken of STS
properties.put(PropertyKeyConst.SecurityToken, "XXX");
......
```