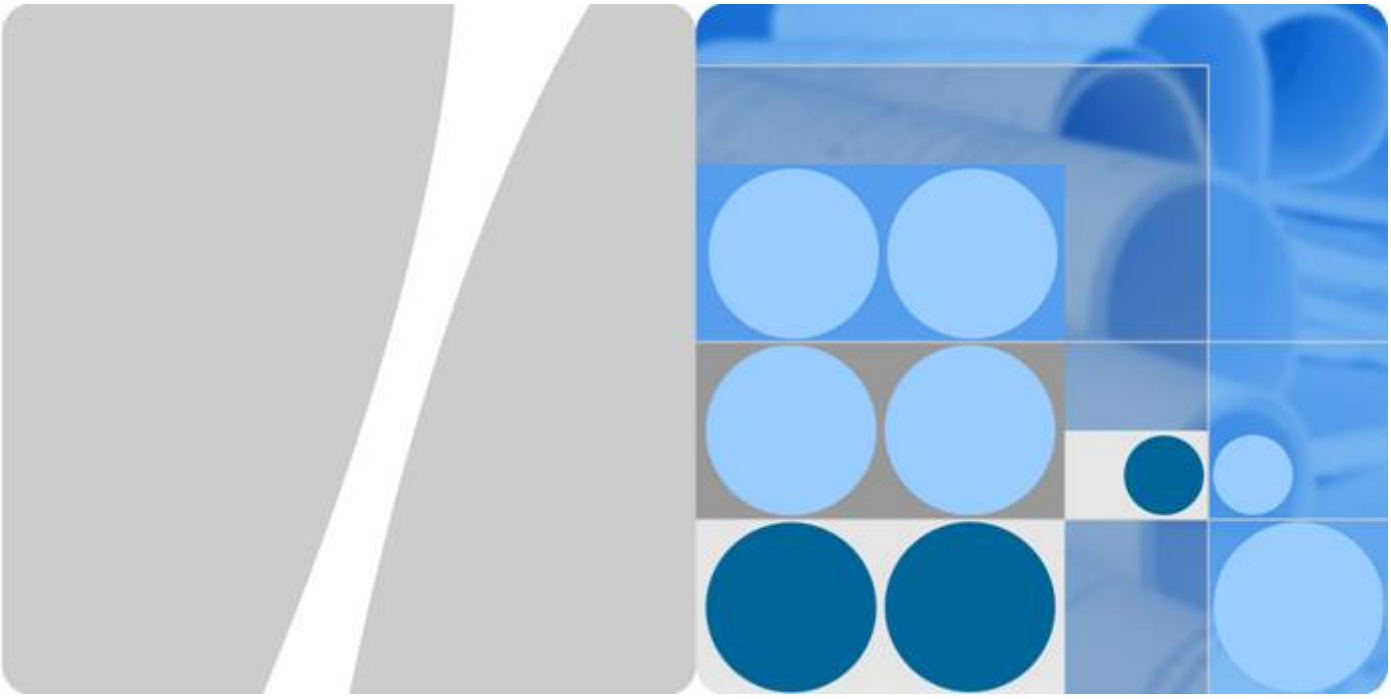


Distributed Message Service

API Reference

Issue 10

Date 2018-07-03

Contents

1 API Usage Guidelines

Public cloud application programming interfaces (APIs) comply with the RESTful API design principles. REST-based Web services are organized into resources. Each resource is identified by one or more Uniform Resource Identifiers (URIs). An application accesses a resource based on the resource's Unified Resource Locator (URL). A URL is usually in the following format: https://Endpoint/uri. In the URL, **uri** indicates the resource path, that is, the API access path.

Public cloud APIs use HTTPS as the transmission protocol. Requests/Responses are transmitted by using JSON messages, with media type represented by **Application/json**.

1.1 Service Usage

Public cloud services provide RESTful APIs.

Representational State Transfer (REST) allocates Uniform Resource Identifiers (URIs) to dispersed resources so that the resources can be located. Applications on clients use Unified Resource Locators (URLs) to obtain the resources.

The URL is in the following format: https://***Endpoint***/***uri***

Table 1-1 describes the parameters in a URL.

Table 1-1 Parameter description

Parameter	Description
Endpoint	Specifies the name of the requested Distributed Message Service (DMS). Obtain the value from **Regions and Endpoints**.
uri	Specifies the API access path for performing a specified operation. Obtain this value from the URI of the API.

NOTE

If an incorrect endpoint is used, the error message "Failed connect to {*Endpoint*}; Connection refused" is returned. If an incorrect URI is used, the error message "Parse failed" or "API not found" is returned.

1.2 AK/SK Authentication

When you use an API gateway to send requests to DMS, the requests are signed using the AK and SK pair. In addition, the **x-project-id** field will be added to the request header during the API calling.

NOTE

AK: indicates the unique ID of the secret access key. AK is used together with SK to obtain an encrypted signature for a request.

SK: indicates the secret access key used together with the access key ID to sign requests. AK and SK can be used together to identify a request sender to prevent the request from being modified.

x-project-id: ID of the current project.

1.2.1 AK and SK Generation

1. Log in to the management console.
2. Click the username and select **My Credential** from the drop-down list.
3. Click **Access Keys**.
4. Click **Add Access Key** to switch to the **Add Access Key** page.
5. Enter the password and the short message verification code, and click **OK** to download the access key. Keep the access key secure.

1.2.2 Request Signing Procedure

Preparations

1. Download the API gateway signing tool from the following website:

   https://support.telefonicaopencloud.com/doc/java-sdk-core.zip
2. Extract the package.
3. Create a Java project, and reference the extracted JAR to the dependency path.

Sign a Request

1. Create a request **com.cloud.sdk.DefaultRequest (JAVA)** used for signing.

2. Set the target API URL, HTTPS method, and content of request **com.cloud.sdk.DefaultRequest (JAVA)**.
3. Sign request **com.cloud.sdk.DefaultRequest (JAVA)**.
   a. Call **SignerFactory.getSigner(String serviceName, String regionName)** to generate a signing tool instance.
   b. Call **Signer.sign(Request<?> request, Credentials credentials)** to sign the request created in step 1.

      The following code shows the details.

      ```
      //Select an algorithm for request signing.
      Signer signer = SignerFactory.getSigner(serviceName, region);
      //Sign the request. The request will change after the signing.
      signer.sign(request, new BasicCredentials(this.ak, this.sk));
      ```
4. Convert the request signed in the previous step to a new request that can be used to make an API call and copy the header of the signed request to the new request.

   For example, if Apache HttpClient is used, convert DefaultRequest to HttpRequestBase and copy the header of the signed DefaultRequest to HttpRequestBase.

   For details, see descriptions of AccessServiceImpl.java in section **1.7 Sample Code**.

1.3 Obtaining a Project ID

A project ID needs to be specified in the URIs of some APIs. Therefore, you need to obtain the project ID before calling APIs.

The following procedure describes how to obtain a project ID:

1. Log in to the management console.
2. Click the username and select **My Credential** from the drop-down list.
3. On the **My Credential** page, view project IDs in the project list.

1.4 Request Construction

A request consists of three parts, namely, a request line, a request header, and a request body (optional).

Request Line

A request line starts with a method token, which is followed by a request uniform resource identifier (URI) and protocol version. The three parts are separated by spaces. The format of the request line is as follows:

```
Method Request-URI HTTP-Version CRLF
```

- Method: indicates a request method. All methods are capitalized. Their meanings are as follows:
  - GET: obtains resources identified by a Request-URI.
  - POST: adds new data to a resource identified by a Request-URI.
  - PUT: requests a server to store a resource and uses a Request-URI to identify the resource.
  - DELETE: requests the server to delete the resource identified by a Request-URI.

  - PATCH: requests a server to update a resource or create a resource if the target resource does not exist.
  - HEAD: requests only the page header.
  - OPTION: allows the client to view the server performance.
- Request-URI: indicates a unified resource identifier.
- HTTP-Version: indicates the version of the HTTP protocol used by a request.
- CRLF: indicates the carriage return and new-line characters. (CRLF can be placed only at the end of a line, and a separate CR or LF is not allowed.)

Request Header

A request header consists of several header fields. Each header field consists of a domain name, a colon (:), and a field value.

Table 1-2 shows the common request headers.

Table 1-2 Common request headers

Parameter	Description	Mandatory or Not	Example
X-Sdk-Date	Specifies the time when the request is sent. The time is in **YYYYMMDD'T'HHMMSS'Z'** format. The value is the current UTC time of the system.	No This header field is mandatory if AK/SK authentication is in use.	20151222T034042Z
Authorization	Specifies the authentication information. The value can be obtained from the request signing result.	No This header field is mandatory if AK/SK authentication is in use.	SDK-HMAC-SHA256 Credential=ZIRRKMTWPTQFQI1WKNKB/20151222/cn-north-1/ec2/sdk_request, SignedHeaders=connection;content-type;host;x-sdk-date, Signature=7972cc9145876d174b3862188a0f61819431fa71c8a8a060809ea8b898e3eaa9

Parameter	Description	Mandatory or Not	Example
Host	Specifies the information about the requested server. The value can be obtained from the URL of the service API. The value is *hostname[:port]*. If the port number is not specified, the default port is used. The default port number for **https** is port **443**.	Yes.	code.test.com or code.test.com:443
Content-type	Specifies the MIME type of the request body.	Yes	application/json
Content-Length	Specifies the size of the request body (unit: byte).	This header field is mandatory for POST and PUT requests. This parameter does not need to be set for GET requests.	3495
x-project-id	ID of the current project.	No This header field is mandatory if AK/SK authentication is in use.	d025c1ff85a24e1d86e15de3bd308a27

Request Body

A request body is encapsulated in the JSON format and the basic syntax is the key:value nested format. The mandatory fields and optional fields in an HTTP request vary depending on the URI object.

Response Message Header

Table 1-3 describes the common response message headers of DMS.

Table 1-3 Common response message headers

Parameter	Description
Content-Length	Size of the response message body (unit: byte).
Date	Specifies the date and time when the message was sent.
Content-type	Specifies the MIME type of the response body.

1.5 Initiating Requests

Use the encoding manner to initiate the request based on constructed request messages. You can call the API through code to assemble, send, and process request messages.

1.6 Parsing Response Messages

After receiving the request message, the server will return an HTTP response message.

A response message consists of three parts: status line, response header, and response body.

Status Line

The format of a status line is as follows:

```
HTTP-Version Status-Code Reason-Phrase CRLF
```

- HTTP-Version: indicates the version of the HTTP protocol used by the server.
- Status-Code: indicates the status code in the response returned by the server.

  Status-Code consists of three digits. The first digit defines the class of a response. There are five values for the first digit:
- 1xx: indication information, indicating that the request has been received and can be further processed
- 2xx: success, indicating that the request has been received, understood, and accepted.
- 3xx: redirection, indicating that the request requires further operations before it can be completed.
- 4xx: client error, indicating that there is a syntax error in the request or the request cannot be implemented.
- 5xx: server error, indicating that the server fails to implement a valid request.
- Reason-Phrase: indicates the text description of the status code.

Response Header

In most cases, response headers are as follows:

- Date: *Mon, 12 Nov 2007 15:55:01 GMT*

  Standard HTTP response header. This header indicates the date and time when the message was sent. The format of the date and time is defined in RFC 822.

- Content-Length: ***xxx***

  Standard HTTP response header. This header indicates the size of the entity body. Decimal type. Unit: byte.

- Content-Type: *application/json; charset=UTF-8*

  Standard HTTP response header. This header indicates the media type of the entity body sent to the recipient.

Response Body

A response body is essentially the text of the JSON format. The coding format of the output content of the response body is UTF-8.

1.7 Sample Code

The following code shows how to sign a request and how to use an HTTP client to send an HTTPS request.

Downloading the Example Code

Step 1 Log in to the management console.

Step 2 On the **Distributed Message Service** page, choose **Using APIs**.

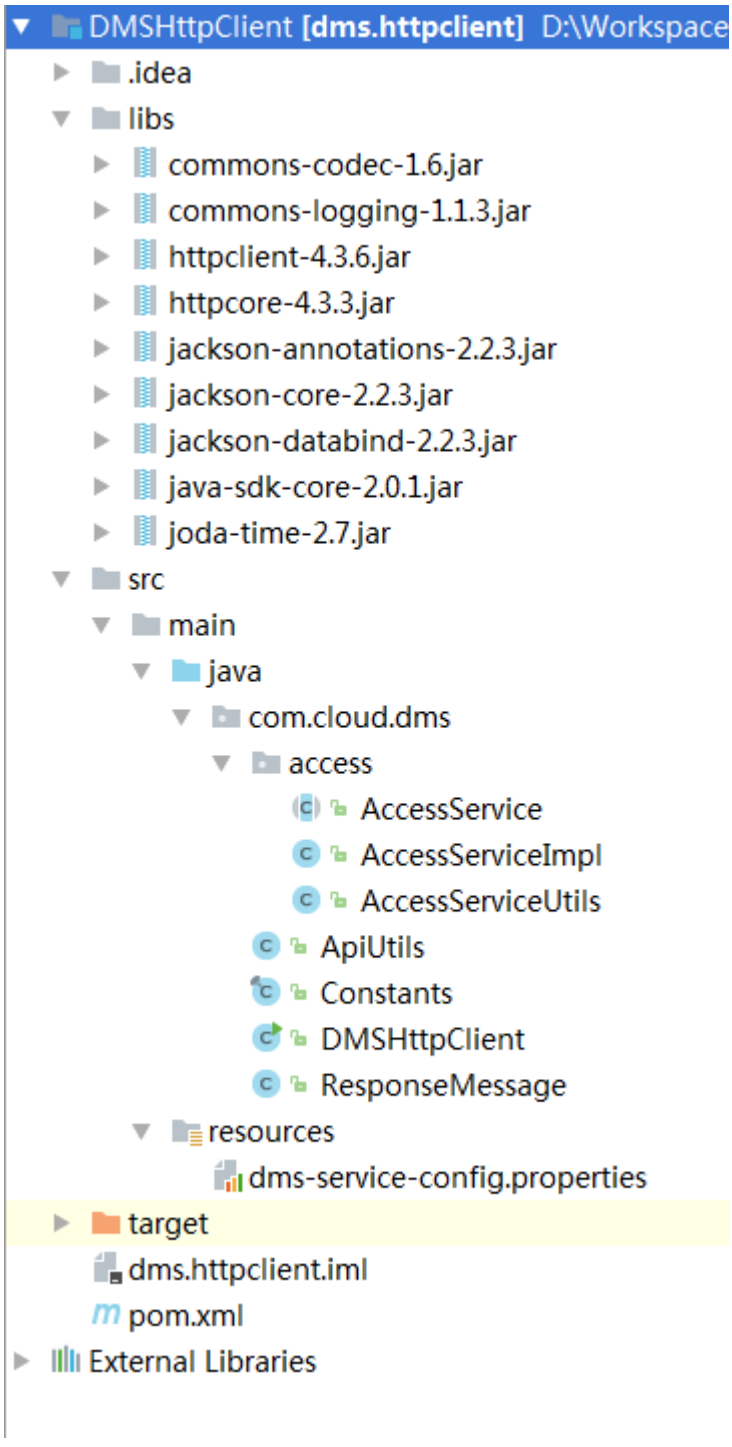
Step 3 Click **Download** to download the example code.

----End

Code Structure

Figure 1-1 shows the code structure.

Figure 1-1 Code structure



AccessService: abstract class that converts the GET, POST, PUT, and DELETE methods in to the access method.

AccessServiceImpl: implementation of the **access** method. Code required for DMS gateway communication is in the access method.

AccessServiceUtils: a tool class that indicates the implementation of the GET, POST, PUT, and DELETE methods, and the parsing method of returned results.

ApiUtils: tool class of the DMS API services, which is used to construct API URIs and parse the information returned by APIs.

DMSHttpClient: execution entry used to simulate GET, POST, PUT, and DELETE request sending.

Constants: class that defines all constants.

ResponseMessage: returned HTTP result object and includes **success**, **body**, and **statusCode** fields.

dms-service-config.properties: service name, region, AK, SK, and DMS endpoint that are essential to DMS gateway signature.

1.8 Status Code

Table 1-4 describes the status code.

Table 1-4 Status code

Status Code	Description
400	1. The request could not be understood by the server due to malformed syntax. The client should not repeat the request without modifications. 2. Request parameters are incorrect.
401	The request requires user authentication.
403	The server understood the request, but is refusing to fulfill it.
404	The server has not found the resources matching the Request-URI.
405	The method specified in the Request-Line is not allowed for the resource identified by the Request-URI.
406	The resource identified by the request is only capable of generating response entities which have content characteristics not acceptable according to the accept headers sent in the request.
409	The request could not be completed due to a conflict with the current state of the resource.
410	The requested resource is no longer available at the server and no forwarding address is known.
412	The server does not meet one of the preconditions that the requester put on the request.
500	The server encountered an unexpected condition which prevented it from fulfilling the request.
501	The server does not support the functionality required to fulfill the request.
503	The server is currently unable to handle the request due to a temporary overloading or maintenance of the server.

2 Managing Queues and Messages

2.1 Creating a Queue

Function

This API is used to create a queue.

By default, a maximum of five queues can be created for a project.

NOTE

It takes one to three seconds to initialize a newly created queue. Therefore, after creating a queue, wait three seconds before performing any operations on the queue. If operations are initiated immediately after a queue is created, these operations will fail. These operations include producing messages, consuming messages, querying queue details, creating consumer groups, and deleting queues.

URI

POST /v1.0/{project_id}/queues

Table 2-1 describes the parameters of this API.

Table 2-1 Parameter description

Parameter	Type	Mandatory or Not	Description
project_id	String	Yes	Indicates the ID of a project.

Request

Request parameters:

Table 2-2 describes the parameters of this API.

Table 2-2 Parameter description

Parameter	Type	Mandatory or Not	Description
name	String	Yes	Indicates the unique name of a queue. A string of 1 to 64 characters that contain a-z, A-Z, 0-9, hyphens (-), and underscores (_). The name cannot be modified once specified.
queue_mode	String	No	Indicates the queue type. Options: ● NORMAL: Standard queue. Best-effort ordering. Messages might be retrieved in an order different from which they were sent. Select standard queues when throughput is important. ● FIFO: First-ln-First-out (FIFO) queue. FIFO delivery. Messages are retrieved in the order they were sent. Select FIFO queues when the order of messages is important. Default value: NORMAL.

Parameter	Type	Mandatory or Not	Description
description	String	No	Indicates the basic information about a queue. The queue description must be 0 to 160 characters in length, and does not contain angle brackets (<) and (>). **NOTE** "\" is defined as an escape character in the queue description. If you need to enter a backward slash (\) or a double quotation mark (") in the queue description, enter \\ or \"
redrive_policy	String	No	Indicates whether to enable dead letter messages. Dead letter messages indicate messages that cannot be normally consumed. If a message fails to be consumed after the number of consumption attempts of this message reaches the maximum value, DMS stores this message into the dead letter queue. This message will be retained in the deal letter queue for 72 hours. During this period, consumers can consume the dead letter message. Dead letter messages can be consumed only by the consumer group that generates these dead letter messages. Dead letter messages of a FIFO queue are stored and consumed based on the FIFO sequence. Options: ● enable ● disable Default value: disable.
max_consume_count	Integer	No	This parameter is mandatory only when **redrive_policy** is set to **enable**. This parameter indicates the maximum number of allowed message consumption failures. When a message fails to be consumed after the number of consumption attempts of this message reaches this value, DMS stores this message into the dead letter queue. Value range: 1 – 100.

Example request:

```
"name" : "queue-001",
"description" : "This is a FIFO queue.",
```

```
  "queue_mode" : "FIFO",
  "redrive_policy" : "enable",
  "max_consume_count" : 3
}
```

Response

Response parameters:

Table 2-3 describes the response parameters.

Table 2-3 Parameter description

Parameter	Type	Description
id	String	Indicates the ID of a queue.
name	String	Indicates the name of a queue.

Example response:

```
{
    "id": "9bf46390-38a2-462d-b392-4d5b2d519c55",
    "name": "queue_001"
}
```

Status Code

Table 2-4 illustrates the status code of successful operations. For details about the status code of failed operations, see **1.8 Status Code**.

Table 2-4 Status code

Status code	Description
201	The queue is created successfully.

2.2 Viewing All Queues

Function

This API is used to view all queues.

URI

GET /v1.0/{project_id}/queues?include_deadletter={include_deadletter}

Table 2-5 describes the parameters of this API.

Table 2-5 Parameter description

Parameter	Type	Mandatory or Not	Description
project_id	String	Yes	Indicates the ID of a project.
include_deadletter	Boolean	No	Indicates whether to list dead letter parameters in the response message. Default value: false.

Example

```
GET v1.0/b78a90ae2a134b4b8b2ba30acab4e23a/queues?include_deadletter=true
```

Request

Request parameters:

None.

Example request:

None.

Response

Response parameters:

Table 2-6 and Table 2-7 describe the response parameters.

Table 2-6 Parameter description

Parameter	Type	Description
total	Integer	The total number of all queues that belong to the tenant.
queues	Array	The total number of all queue arrays that belong to the tenant.

Table 2-7 queues parameter

Parameter	Type	Description
id	String	Indicates the ID of a queue.
name	String	Indicates the name of a queue.
created	Timestamp	Indicates the time when a queue is created.
description	String	Indicates the basic information about a queue.
queue_mode	String	Indicates the queue type.

Parameter	Type	Description
reservation	Integer	Indicates the retention period (unit: min) of a message in a queue.
max_msg_size_byte	Integer	Indicates the maximum message size (unit: byte) that is allowed in queue.
produced_messages	Integer	Indicates the total number of messages (not including the messages that have expired and been deleted) in a queue.
redrive_policy	String	Indicates whether to enable dead letter messages. This parameter is displayed only when **include_deadletter** is set to **true**. Options: ● enable ● disable
max_consume_count	Integer	This parameter indicates the maximum number of allowed message consumption failures. When a message fails to be consumed after the number of consumption attempts of this message reaches this value, DMS stores this message into the dead letter queue. This parameter is displayed only when **include_deadletter** is set to **true**.
group_count	Integer	Indicates the total number of consumer groups in a queue.

Example response:

```
{
  "queues" : [{
      "id" : "ef808d2d-58c2-4a36-9e58-d018b2193f80",
      "name" : "aaa_fifo_525",
      "description" : "test_fifo_detail",
      "queue_mode" : "NORMAL",
      "reservation" : 4320,
      "created" : 1495701557000,
      "max_msg_size_byte" : 524288,
      "produced_messages" : 1,
      "redrive_policy" : "enable",
      "max_consume_count" : 3,
      "group_count" : 0
    }, {
      "id" : "bc0ac1ec-a4d6-4490-84cb-9d475f1ec3c5",
      "name" : "aaa_normal_525",
      "description" : "test",
      "queue_mode" : "NORMAL",
      "reservation" : 4320,
      "created" : 1495701490000,
      "max_msg_size_byte" : 524288,
      "produced_messages" : 0,
      "redrive_policy" : "enable",
      "max_consume_count" : 3,
```

```
      "group_count" : 0
    }, {
      "id" : "1aaf34d0-7bb0-43be-9b71-f4b719d7ca47",
      "name" : "queue-normal",
      "description" : null,
      "queue_mode" : "NORMAL",
      "reservation" : 4320,
      "created" : 1495447342000,
      "max_msg_size_byte" : 524288,
      "produced_messages" : 2,
      "redrive_policy" : "enable",
      "max_consume_count" : 3,
      "group_count" : 0
    }
  ],
  "total" : 3
}
```


Status Code

Table 2-8 illustrates the status code of successful operations. For details about the status code of failed operations, see 1.8 Status Code.

Table 2-8 Status code

Status code	Description
200	The information is obtained successfully.

2.3 Viewing a Specified Queue

Function

This API is used to view a specified queue.

URI

GET /v1.0/{project_id}/queues/{queue_id}?include_deadletter={include_deadletter}

Table 2-9 describes the parameters of this API.

Table 2-9 Parameter description

Parameter	Type	Mandatory or Not	Description
project_id	String	Yes	Indicates the ID of a tenant.
queue_id	String	Yes	Indicates the ID of the queue to be queried.
include_deadletter	Boolean	No	Indicates whether to list dead letter parameters in the response message. Default value: false.

Example

```
GET v1.0/b78a90ae2a134b4b8b2ba30acab4e23a/queues/
075ae7da-6ce5-4966-940c-17c19fb5175e?include_deadletter=true
```

Request

Request parameters:

None.

Example request:

None.

Response

Response parameters:

Table 2-10 describes the response parameters.

Table 2-10 Parameter description

Parameter	Type	Description
id	String	Indicates the ID of a queue.
name	String	Indicates the name of a queue.
created	Timestamp	Indicates the time when a queue is created.
description	String	Indicates the basic information about a queue.
queue_mode	String	Indicates the queue type.
reservation	Integer	Indicates the retention period (unit: min) of a message in a queue.
max_msg_size_byte	Integer	Indicates the maximum message size (unit: byte) that is allowed in queue.
produced_messages	Integer	Indicates the total number of messages (not including the messages that have expired and been deleted) in a queue.
redrive_policy	String	Indicates whether to enable dead letter messages. This parameter is displayed only when **include_deadletter** is set to **true**. Options: ● enable ● disable

Parameter	Type	Description
max_consume_count	Integer	This parameter indicates the maximum number of allowed message consumption failures. When a message fails to be consumed after the number of consumption attempts of this message reaches this value, DMS stores this message into the dead letter queue. This parameter is displayed only when **include_deadletter** is set to **true**.
group_count	Integer	Indicates the total number of consumer groups in a queue.

Example response:

```
{
   "id": "9bf46390-38a2-462d-b392-4d5b2d519c55",
   "name": "queue_001",
   "description": "test1",
   "queue_mode" : "NORMAL",
   "created": 1470063965218,
   "reservation": 4320,
   "max_msg_size_byte": 524288,
   "produced_messages": 5,
   "redrive_policy" : "disable",
   "max_consume_count" : 0,
   "group_count": 0
}
```

Status Code

Table 2-11 illustrates the status code of successful operations. For details about the status code of failed operations, see 1.8 Status Code.

Table 2-11 Status code

Status code	Description
200	The information is obtained successfully.

2.4 Deleting a Specified Queue

Function

This API is used to delete a specified queue.

URI

DELETE /v1.0/{project_id}/queues/{queue_id}

Table 2-12 describes the parameters of this API.

Table 2-12 Parameter description

Parameter	Type	Mandatory or Not	Description
project_id	String	Yes	Indicates the ID of a project.
queue_id	String	Yes	Indicates the ID of the queue to be deleted.

Request

Request parameters:

None.

Example request:

None.

Response

Response parameters:

None

Example response:

None.

Status Code

Table 2-13 illustrates the status code of successful operations. For details about the status code of failed operations, see **1.8 Status Code**.

Table 2-13 Status code

Status code	Description
204	The queue is deleted successfully.

2.5 Creating a Consumer Group

Function

This API is used to create a consumer group.

Multiple consumer groups can be created for a queue at one time. A queue supports a maximum of three consumer groups.

NOTE

After you create a consumer group, it takes 1s to 3s to initialize the system. If you operate the queue immediately after creating the consumer group, the consumption may fail. You are advised to perform operations 3s later.

URI

POST /v1.0/{project_id}/queues/{queue_id}/groups

Table 2-14 describes the parameters of this API.

Table 2-14 Parameter description

Parameter	Type	Mandatory or Not	Description
project_id	String	Yes	Indicates the ID of a project.
queue_id	String	Yes	Indicates the ID of a specified queue.

Request

Request parameters:

Table 2-15 describes the parameters of this API.

Table 2-15 Parameter description

Parameter	Type	Mandatory or Not	Description
groups	Array	Yes	Indicates the information of a consumer group.

Table 2-16 groups parameter

Parameter	Type	Mandatory or Not	Description
name	String	Yes	Indicates the name of a consumer group. A string of 1 to 32 characters that contain a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Example request:

```
{
  "groups" : [{
      "name" : "group-aa"
    }
  ]
}
```

Response

Response parameters:

Table 2-17 describes the response parameters.

Table 2-17 Parameter description

Parameter	Type	Description
id	String	Indicates the ID of a consumer group.
name	String	Indicates the name of a consumer group.

Example response:

```
{
  "groups" : [{
      "id" : "g-02fb1974-9be1-4eee-8448-ed2d3e89884a",
      "name" : "group-aa"
    }
  ]
}
```

Status Code

Table 2-18 illustrates the status code of successful operations. For details about the status code of failed operations, see **1.8 Status Code**.

Table 2-18 Status code

Status code	Description
201	The consumer group is created successfully.

2.6 Viewing All Consumer Groups of a Specified Queue

Function

This API is used to view all consumer groups of a specified queue.

URI

GET /v1.0/{project_id}/queues/{queue_id}/groups?include_deadletter={include_deadletter}

Table 2-19 describes the parameters of this API.

Table 2-19 Parameter description

Parameter	Type	Mandatory or Not	Description
project_id	String	Yes	Indicates the ID of a project.
queue_id	String	Yes	Indicates the ID of a specified queue.
include_deadletter	Boolean	No	Indicates whether to list dead letter parameters in the response message. Default value: **false**.

Example

```
GET v1.0/b78a90ae2a134b4b8b2ba30acab4e23a/queues/
075ae7da-6ce5-4966-940c-17c19fb5175e/groups?include_deadletter=true
```

Request

Request parameters:

None.

Example request:

None.

Response

Response parameters:

Table 2-20 and Table 2-21 describe the response parameters.

Table 2-20 Parameter description

Parameter	Type	Description
queue_id	String	Indicates the ID of a queue.
queue_name	String	Indicates the name of a queue.
groups	Array	Indicates the consumer group list.
redrive_policy	String	Indicates whether to enable dead letter messages. This parameter is displayed only when **include_deadletter** is set to **true**. Options: ● enable ● disable

Table 2-21 groups parameter

Parameter	Type	Description
id	String	Indicates the ID of a consumer group.
name	String	Indicates the name of a consumer group.
produced_messages	Integer	Indicates the total number of messages (not including the messages that have expired and been deleted) in a queue.
consumed_messages	Integer	Indicates the total number of messages that are successfully consumed.
available_messages	Integer	Indicates the accumulated number of messages that can be consumed.
produced_deadletters	Integer	Indicates the total number of dead letter messages generated by the consumer group. This parameter is displayed only when **include_deadletter** is set to **true**.
available_deadletters	Integer	Indicates the accumulated number of dead letter messages that have not been consumed. This parameter is displayed only when **include_deadletter** is set to **true**.

Example response:

```
{
    "queue_name" : "queue-772289871",
    "groups" : [{
            "name" : "group-1690260950",
            "id" : "g-eb9305bb-5bec-4712-84ab-0a36fbe9c2c0",
            "consumed_messages" : 0,
            "available_messages" : 8,
            "produced_messages" : 10,
            "produced_deadletters" : 2,
            "available_deadletters" : 2
        }
    ],
    "redrive_policy" : "enable",
    "queue_id" : "f5b6dd28-08dd-4f0f-866c-2eadf6788163"
}
```

Status Code

Table 2-22 illustrates the status code of successful operations. For details about the status code of failed operations, see **1.8 Status Code**.

Table 2-22 Status code

Status code	Description
200	The information is obtained successfully.

2.7 Deleting a Specified Consumer Group

Function

This API is used to delete a specified consumer group.

URI

DELETE /v1.0/{project_id}/queues/{queue_id}/groups/{group_id}

Table 2-23 describes the parameters of this API.

Table 2-23 Parameter description

Parameter	Type	Mandatory or Not	Description
project_id	String	Yes	Indicates the ID of a project.
queue_id	String	Yes	Indicates the ID of a queue.
group_id	String	Yes	Indicates the ID of the consumer group to be deleted.

Request

Request parameters:

None.

Example request:

None.

Response

Response parameters:

None.

Example response:

None.

Status Code

Table 2-24 illustrates the status code of successful operations. For details about the status code of failed operations, see **1.8 Status Code**.

Table 2-24 Status code

Status code	Description
204	The consumer group is deleted successfully.

2.8 Sending Messages to a Specified Queue

Function

This API is used to send messages to a specified queue. Multiple messages can be sent at the same time. The following requirements must be met:

- A maximum of 10 messages can be sent at a time.
- The aggregated size of messages sent at a time cannot exceed 512 KB.

To improve the message sending and consumption efficiency, you are advised to use the batch message sending and consumption mode, which can effectively lower the number of API calls and minimize service fees.

URI

POST /v1.0/{project_id}/queues/{queue_id}/messages

Table 2-25 describes the parameters of this API.

Table 2-25 Parameter description

Parameter	Type	Mandatory or Not	Description
project_id	String	Yes	Indicates the ID of a project.
queue_id	String	Yes	Indicates the ID of a specified queue.

Request

Request parameters:

Table 2-26 and Table 2-27 describe the request parameters.

Table 2-26 Parameter description

Parameter	Type	Mandatory or Not	Description
messages	Array	Yes	Indicates the list of messages.

Table 2-27 messages parameter

Parameter	Type	Mandatory or Not	Description
body	JSON	Yes	Indicates the message body. **NOTE** "\" is defined as an escape character in a message body. If you need to enter a backward slash (\) or a double quotation mark (") in a message body, enter \\ or \".
attributes	JSON	No	Indicates the list of attributes, including attribute names and values. The attribute name must be unique for a message.

Example request:

```
{
 "messages" : [{
   "body" : "TEST11",
   "attributes" : {
    "attribute1" : "value1",
    "attribute2" : "value2"
   }
  }, {
   "body" : {
    "foo" : "test02"
   },
   "attributes" : {
    "attribute1" : "value1",
    "attribute2" : "value2"
   }
  }
 ]
}
```

Response

Response parameter:

Table 2-28 and Table 2-29 describe the response parameters.

Table 2-28 Parameter description

Parameter	Type	Description
message	JSON object	Message list.

Table 2-29 messages parameter

Parameter	Type	Description
error	String	Indicates error information.

Parameter	Type	Description
error_code	Integer	Indicates the error code (if any).
state	Integer	Indicates message sending status. **0**: Messages are successfully sent. **1**: Messages failed to be sent. The **error** and **error_code** parameters indicate the cause of failure.

Example response:

```
{
 "messages" : [
  {
   "error" : null,
   "error_code" : null,
   "state" : 0
  },
  {
   "error_code" : null,
   "state" : 0,
   "error" : null
  }
 ]
}
```

Status Code

Table 2-30 illustrates the status code of successful operations. For details about the status code of failed operations, see 1.8 Status Code.

Table 2-30 Status code

Status code	Description
201	Messages are sent successfully.

2.9 Consuming Messages

Function

This API is used to consume messages in a specified queue.

Multiple messages can be consumed at the same time. The load of messages consumed each time cannot exceed 512 KB.

To improve the message sending and consumption efficiency, you are advised to use the batch message sending and consumption mode, which can effectively lower the number of API calls and minimize service fees.

When there are only a few messages in a queue, the number of messages actually consumed at a time may be less than the message quantity specified in the consumption request. However,

all messages in the queue will be eventually obtained by the message consumer after multiple rounds of consumption. If the queue is empty, no message will be returned to the consumer.

URI

GET /v1.0/{project_id}/queues/{queue_id}/groups/{consumer_group_id}/messages?max_msgs={max_msgs}&time_wait={time_wait}&ack_wait={ack_wait}

Table 2-31 describes the parameters of this API.

Table 2-31 Parameter description

Parameter	Type	Mandatory or Not	Description	Value Range
project_id	String	Yes	Indicates the ID of a project.	N/A
queue_id	String	Yes	Indicates the ID of a specified queue.	N/A
consumer_group_id	String	Yes	Indicates the ID of a consumer group. You can obtain the consumer group ID from the response message returned after you call the API to view all consumer groups of a specified queue. For details, see section **3.6 Viewing All Consumer Groups of a Specified Queue**.	N/A
max_msgs	Integer	No	Indicates the number of consumable messages that can be obtained per time.	Value range: 1 to 10. Default value: 10.

Parameter	Type	Mandatory or Not	Description	Value Range
time_wait	Integer	No	Indicates the amount of time that the API call can wait for a message to arrive in the empty queue before returning an empty response. If a message is available during the wait period, the call will return the message consumption result immediately. If no message is available until the wait period expires, the call will return an empty response after the wait period expires.	Value range: 1 to 60 seconds. Default value: 3. The wait period is 3 seconds even if the API request does not carry the time_wait parameter or the time_wait parameter in the API request is left unspecified.
ack_wait	Integer	No	Indicates the amount of time that the API call can wait for a message retrieval acknowledgement. The client needs to submit the message retrieval acknowledgement within the specified time. If the message retrieval is not acknowledged within this period of time, the system displays a message, indicating that message retrieval acknowledgement has timed out or the handler is invalid. In this case, the system determines that the message fails to be retrieved.	Value range: 15 to 300 seconds. Default value: 30. If this parameter is left unspecified, the default value 30s is used.

Example

```
GET v1.0/b78a90ae2a134b4b8b2ba30acab4e23a/queues/
075ae7da-6ce5-4966-940c-17c19fb5175e/groups/g-5ec247fd-d4a2-4d4f-9876-
e4ff3280c461/messages?max_msgs=10&ack_wait=30
```

Request

Request parameters:

None.

Example request:

None.

Response

Response parameters:

Table 2-32 and Table 2-33, describe the response parameters.

Table 2-32 Parameter description

Parameter	Type	Description
message	JSON	Indicates the message content.
handler	String	Indicates the message handler.

Table 2-33 messages parameter

Parameter	Type	Description
body	JSON	Indicates the message body.
attributes	JSON	Indicates the list of attributes.

Example response:

```
[{
  "message" : {
   "body" : {
    "foo" : "123="
   },
   "attributes": {
       "attribute1": "value1",
       "attribute2": "value2"
        }
  },
  "handler" :
"eyJjZyI6Im15X2pzb25fZ3JvdXAiLCJjaSI6InJlc3QtY29uc3VtZXItYzNlNThiNjEtYzA0NC00NGJkL
TkxM2ItZDgzNjljNmJhYTQxIiwiY291bnQiOjAsIm9mZnNldCI6MCwicCI6MCwidCI6InRlc3QyIn0="
 }, {
  "message" : {
   "body" : {
    "foo" : "123="
   },
   "attributes": {
       "attribute1": "value1",
       "attribute2": "value2"
        }
  },
  "handler" :
```

```
"eyJjZyI6Im15X2pzb25fZ3JvdXAiLCJjaSI6InJlc3QtY29uc3VtZXItYzNlNThiNjEtYzA0NC00NGJkL
TkxM2ItZDgzNjljNmJhYTQxIiwiY291bnQiOjAsIm9mZnNldCI6MSwicCI6MCwidCI6InRlc3QyIn0="
 }
]
```

Status Code

Table 2-34 illustrates the status code of successful operations. For details about the status code of failed operations, see 1.8 Status Code.

Table 2-34 Status code

Status code	Description
200	The information is obtained successfully.

2.10 Acknowledging the Consumption of Specified Messages

Function

This API is used to acknowledge the consumption of specified messages.

While a message is being consumed, it remains in the queue and cannot be consumed again within 30s since the consumption started. If the consumption is not acknowledged within this period, message can be consumed again.

If a message is successfully acknowledged by a consumer group, the message cannot be re-consumed by the same group. However, message can still be consumed by other consumer groups. It is retained in the queue for 72 hours (unless the queue is deleted) and will be deleted after this period.

After a batch of messages is consumed, consumers must submit their consumption acknowledgement by strictly following the message consumption sequence. DMS sequentially checks whether messages are successfully consumed. If DMS finds that a message is not acknowledged as a consumed message or fails to be consumed, DMS stops checking but directly determines that all the subsequent messages fail to be consumed. Therefore, when a consumer fails to acknowledge the consumption of a message (in a batch of messages), you are advised to stop the consumer from consuming the rest of the messages, and directly submit acknowledgement of the successfully consumed messages to DMS.

If the consumption of a message fails to be acknowledged, this message can be re-consumed and its consumption can also be acknowledged again. If dead letter messages are enabled and a message fails to be consumed for a preset number of times, the message will be sent to the dead letter queue and retained in the dead letter queue for a maximum of 72 hours. You can then consume the message from the dead letter queue.

URI

POST /v1.0/{project_id}/queues/{queue_id}/groups/{consumer_group_id}/ack

Table 2-35 describes the parameters of this API.

Table 2-35 Parameter description

Parameter	Type	Mandatory or Not	Description
project_id	String	Yes	Indicates the ID of a project.
queue_id	String	Yes	Indicates the ID of a queue.
consumer_group_id	String	Yes	Indicates the ID of a consumer group.

Request

Request parameters:

Table 2-36 and Table 2-37 describe the request parameters.

Table 2-36 Parameter description

Parameter	Type	Mandatory or Not	Description
message	Array	Yes	Indicates the acknowledgement of message arrays.

Table 2-37 message parameter

Parameter	Type	Mandatory or Not	Description
handler	String	Yes	Indicates the ID returned during the consumption.
status	String	Yes	Message consumption status, which can be **success** or **fail**.

Example request:

```
{
  "message": [
    {
      "handler":
"eyJjb25zdW1lckdyb3VwIjoibXFzX2NvbnN1bWVyXzMiLCJjb25zdW1lckluc3RhbmNlIjoicmVzdC1jb25zdW1lci1hMWM5YTRlMy1mNTY5LTQyYTgtOTQ1Ni1hYmU0NDVmZjUxYzkiLCJjb3VudCI6MSwib2Zmc2V0IjowLCJvZmZzZXRJbmRleCI6LTEsInBhcnRpdGlvbiI6MiwidG9waWMiOiJxLWI3OGE5MGFlMmExMzRiNGI4YjJiYTMwYWNhYjRlMjNhLTA3NWFlN2RhLTZjZTUtNDk2Ni05NDBjLTE3YzE5ZmI1MTc1ZSJ9",
      "status": "success"
    }
  ]
}
```

Response

Response parameters:

Table 2-38 describes the response parameters.

Table 2-38 Parameter description

Parameter	Type	Description
success	Integer	Indicates the number of messages that are successfully acknowledged. The value **N** indicates that the first *N* messages are successfully acknowledged.
fail	Integer	Indicates the number of messages that failed to be acknowledged. The value **N** indicates that the last *N* messages failed to be acknowledged.

Example response:

```
{
  "success": 1,
  "fail": 2
}
```

Status Code

Table 2-39 illustrates the status code of successful operations. For details about the status code of failed operations, see **1.8 Status Code**.

Table 2-39 Status code

Status code	Description
200	The messages are successfully consumed.

2.11 Viewing the Quotas

Function

This API is used to view the quotas of the current project.

URI

GET /v1.0/{project_id}/quotas/dms

Table 2-40 describes the parameters of this API.

Table 2-40 Parameter description

Parameter	Type	Mandatory or Not	Description
project_id	String	Yes	Indicates the ID of a project.

Request

Request parameters:

None.

Example request:

None.

Response

Response parameters:

Table 2-41, **Table 2-42**, and **Table 2-43** describe the response parameters.

Table 2-41 Parameter description

Parameter	Type	Description
quotas	JSON	Indicates the quotas of a tenant.

Table 2-42 quotas parameter

Parameter	Type	Description
resources	Array	Indicates the list of quotas.

Table 2-43 resources parameter

Parameter	Type	Description
type	String	Indicates the name of a quota.
quota	Integer	Indicates the total quota.
used	Integer	Indicates the consumed quota.
min	Integer	Indicates the minimum limit that a quota must reach.
max	Integer	Indicates the maximum limit that a quota cannot exceed.

Example response:

```
{
  "quotas" : {
    "resources" : [{
        "type" : "queue",
        "quota" : 5,
        "used" : 3,
        "min" : 1,
        "max" : 20
      }
    ]
  }
}
```

Status Code

Table 2-44 illustrates the status code of successful operations. For details about the status code of failed operations, see 1.8 Status Code.

Table 2-44 Status code

Status code	Description
200	The information is obtained successfully.

2.12 Consuming Dead Letter Messages

Function

This API is used to consume the dead letter messages generated by a specified consumer group.

Multiple messages can be consumed at the same time. The load of messages consumed each time cannot exceed 512 KB.

URI

GET /v1.0/{project_id}/queues/{queue_id}/groups/{consumer_group_id}/deadletters?max_msgs={max_msgs}&time_wait={time_wait}&ack_wait={ack_wait}

Table 2-45 describes the parameters of this API.

Table 2-45 Parameter description

Parameter	Type	Mandatory or Not	Description	Value Range
project_id	String	Yes	Indicates the ID of a project.	N/A
queue_id	String	Yes	Indicates the ID of a specified queue.	N/A

Parameter	Type	Mandatory or Not	Description	Value Range
consumer_group_id	String	Yes	Indicates the ID of a consumer group. You can obtain the consumer group ID from the response message returned after you call the API to view all consumer groups of a specified queue. For details, see section **3.6 Viewing All Consumer Groups of a Specified Queue**.	N/A
max_msgs	Integer	No	Indicates the number of consumable dead letter messages that can be obtained per time. **NOTE** The number of dead letter messages actually consumed at a time may be less than the message quantity specified in the consumption request. However, all dead letter messages in the queue will be eventually obtained by the consumer after multiple rounds of consumption.	Value range: 1 to 10. Default value: 10.
time_wait	Integer	No	Indicates the amount of time that the API call can wait for a dead letter message to arrive in the empty queue before returning an empty response. If a dead letter message is available during the wait period, the call will return the message consumption result immediately. If no dead letter message is available until the wait period expires, the call will return an empty response after the wait period expires.	Value range: 1 to 60 seconds. Default value: 3. The wait period is 3 seconds even if the API request does not carry the **time_wait** parameter or the **time_wait** parameter in the API request is left unspecified.

Parameter	Type	Mandatory or Not	Description	Value Range
ack_wait	Integer	No	Indicates the amount of time that the API call can wait for a message retrieval acknowledgement. The client needs to submit the message retrieval acknowledgement within the specified time. If the message retrieval is not acknowledged within this period of time, the system displays a message, indicating that message retrieval acknowledgement has timed out or the handler is invalid. In this case, the system determines that the message fails to be retrieved.	Value range: 15 to 300 seconds. Default value: 30. If this parameter is left unspecified, the default value 30s is used.

Example

```
GET v1.0/b78a90ae2a134b4b8b2ba30acab4e23a/queues/
075ae7da-6ce5-4966-940c-17c19fb5175e/groups/g-5ec247fd-d4a2-4d4f-9876-
e4ff3280c461/deadletters?max_msgs=10&ack_wait=30
```

Request

Request parameters:

None.

Example request:

None.

Response

Response parameters:

Table 2-46 and Table 2-47, describe the response parameters.

Table 2-46 Parameter description

Parameter	Type	Description
message	JSON	Indicates the message content.
handler	String	Indicates the message handler.

Table 2-47 message parameter

Parameter	Type	Description
body	JSON	Indicates the message body.
attributes	JSON	Indicates the list of attributes.

Example response:

```
[{
  "message" : {
   "body" : {
    "foo" : "123="
   },
   "attributes": {
       "attribute1": "value1",
       "attribute2": "value2"
        }
  },
  "handler" :
"eyJjZyI6Im15X2pzb25fZ3JvdXAiLCJjaSI6InJlc3QtY29uc3VtZXItYzNlNThiNjEtYzA0NC00NGJkL
TkxM2ItZDgzNjljNmJhYTQxIiwiY291bnQiOjAsIm9mZnNldCI6MCwicCI6MCwidCI6InRlc3QyIn0="
 }, {
  "message" : {
   "body" : {
    "foo" : "123="
   },
   "attributes": {
       "attribute1": "value1",
       "attribute2": "value2"
        }
  },
  "handler" :
"eyJjZyI6Im15X2pzb25fZ3JvdXAiLCJjaSI6InJlc3QtY29uc3VtZXItYzNlNThiNjEtYzA0NC00NGJkL
TkxM2ItZDgzNjljNmJhYTQxIiwiY291bnQiOjAsIm9mZnNldCI6MSwicCI6MCwidCI6InRlc3QyIn0="
 }
]
```

Status Code

Table 2-48 illustrates the status code of successful operations. For details about the status code of failed operations, see **1.8 Status Code**.

Table 2-48 Status code

Status code	Description
200	The dead letter message is consumed successfully.

2.13 Acknowledging the Consumption of Specified Dead Letter Messages

Function

This API is used to acknowledge the consumption of specified dead letter messages.

When a dead letter message is being consumed, it remains in the queue. It cannot be consumed again by the same consumer group within 30s since the start of the consumption. If consumption is not acknowledged within this period, DMS determines that this dead letter message fails to be consumed, and this dead letter message can be consumed again.

Once consumption is acknowledged, this dead letter message can no longer be consumed by the same consumer group. Dead letter messages remain in the queue for 72 hours (unless the consumer group is deleted) and will be deleted after this period.

After a batch of messages is consumed, consumers must submit their consumption acknowledgement by strictly following the message consumption sequence. DMS sequentially checks whether messages are successfully consumed. If DMS finds that a message is not acknowledged as a consumed message or fails to be consumed, DMS stops checking but directly determines that all the subsequent messages fail to be consumed. Therefore, when a consumer fails to consume a message (in a batch of messages), you are advised to stop the consumer from consuming the rest messages, and directly submit acknowledgement of the successfully consumed messages to DMS.

URI

POST /v1.0/{project_id}/queues/{queue_id}/groups/{consumer_group_id}/deadletters/ack

Table 2-49 describes the parameters of this API.

Table 2-49 Parameter description

Parameter	Type	Mandatory or Not	Description
project_id	String	Yes	Indicates the ID of a project.
queue_id	String	Yes	Indicates the ID of a queue.
consumer_group_id	String	Yes	Indicates the ID of a consumer group.

Request

Request parameters:

Table 2-50 and Table 2-51 describe the request parameters.

Table 2-50 Parameter description

Parameter	Type	Mandatory or Not	Description
message	Array	Yes	Indicates the array of the message consumption acknowledgement.

Table 2-51 message parameter

Parameter	Type	Mandatory or Not	Description
handler	String	Yes	Indicates the ID returned during the consumption.
status	String	Yes	Indicates the message consumption status, which can be **success** or **fail**.

Example request:

```
{
  "message": [
    {
      "handler":
"eyJjb25zdW1lckdyb3VwIjoibXFzX2NvbnN1bWVyXzMiLCJjb25zdW1lckluc3RhbmNlIjoicmVzdC1jb
25zdW1lci1hMWM5YTRlMy1mNTY5LTQyYTgtOTQ1Ni1hYmU0NDVmZjUxYzkiLCJjb3VudCI6MSwib2Zmc2V
0IjowLCJvZmZzZXRJbmRleCI6LTEsInBhcnRpdGlvbiI6MiwidG9waWMiOiJxLWI3OGE5MGFlMmExMzRiN
GI4YjJiYTMwYWNhYjRlMjNhLTA3NWFlN2RhLTZjZTUtNDk2Ni05NDBjLTE3YzE5ZmI1MTc1ZSJ9",
      "status": "success"
    }
  ]
}
```

Response

Response parameters:

Table 2-52 describes the response parameters.

Table 2-52 Parameter description

Parameter	Type	Description
success	Integer	Indicates the number of dead letter messages that are successfully acknowledged. The value **N** indicates that the first *N* dead letter messages are successfully acknowledged.
fail	Integer	Indicates the number of dead letter messages that failed to be acknowledged. The value **N** indicates that the last *N* dead letter messages failed to be acknowledged.

Example response:

```
{
  "success": 1,
  "fail": 2
}
```

Status Code

Table 2-53 illustrates the status code of successful operations. For details about the status code of failed operations, see **1.8 Status Code**.

Table 2-53 Status code

Status code	Description
200	The consumption of the dead letter message is successfully acknowledged.

3 FAQ

3.1 Why Is the Message "Connect IAM Timeout" Displayed When I Attempt to Access DMS?

Symptom

An error message "Connect IAM Timeout" is displayed when I attempt to use the API to access DMS.

```
Get quota fail: 401
{"message": "Connect IAM Timeout", "request_id": "5ACB6B21-DAF6-47C8-
B7A4-45A7BDC57FC6"}
```

Possible Cause

The AK/SK pair is deleted on the web-based DMS console.

Troubleshooting Method

Step 1 Log in to the management console.

Step 2 Click the username and select **My Credential** from the drop-down list.

Step 3 Click **Access Keys**.

Step 4 Click **Add Access Key** to switch to the **Add Access Key** page.

Step 5 Enter the password and the short message verification code, and click **OK** to download the access key. Keep the access key secure.

Step 6 Use the new AK/SK pair to access DMS.

----End

4 Appendix

4.1 Error Code Description

Status code	Error Code	Description
400	10240002	The number of query queues is greater than {0}.
400	10240005	Invalid project ID format.
400	10240007	The name contains invalid characters.
400	10240009	The message body contains invalid characters or is not in JSON format.
400	10240010	The description contains invalid characters.
400	10240011	The name must be 1 to 64 characters.
400	10240012	The name must be 1 to 32 characters.
400	10240013	The description must be 0 to 160 characters.
400	10240014	The value of max_msg must be between 1 and 10.
400	10240015	Invalid queue ID format.
400	10240016	Invalid group ID format.
400	10240017	The message queue already exists.
400	10240018	The consumer group already exists.
400	10240019	Exceeded the maximum allowed number of consumer groups.
400	10240020	Insufficient quota.
400	10240021	The value of time_wait must be between 1 and 60.

Status code	Error Code	Description
400	10240022	The value of max Consume Count must be between 1 and 100.
400	10240032	The queue is being created
401	10240101	Invalid token.
401	10240103	Missing token.
401	10240104	The token does not match Project-ID.
403	10240304	Quota is smaller than used.
403	10240306	The tenant is frozen and therefore cannot perform operations on DMS.
403	10240308	Queue quota size must be between 1 and 20.
403	10240309	Access denied. Therefore, operations on DMS are not allowed.
403	10240310	The tenant is a read-only user and therefore cannot perform operations on DMS.
403	10240311	The role is not assigned any permission and therefore cannot perform operations on DMS.
403	10240312	The tenant is restricted.
404	10240401	Wrong or missing Queue-ID
404	10240405	Wrong or missing Group-ID
404	10240406	The URL or endpoint does not exist.
500	10250002	Internal service error.
500	10250003	Internal service error.
500	10250004	Internal service error.
500	10250005	Internal communication error.
500	10250006	Internal service error.
400	10540001	The message body contains invalid fields.
400	10540003	Message ack status must be either 'success' or 'fail'. It should not be '{status}'.
400	10540004	The queue or group name does not match the handler.
400	10540010	Invalid request format: {desc}.
400	10540011	The message size is {message size}, larger than the size limit {max allowed size}.

Status code	Error Code	Description
400	10540012	The message body contains invalid characters or is not in JSON format.
400	10540202	Invalid request format: {desc}.
400	10542204	Failed to consume messages due to the cause {desc}.
400	10542205	The handler does not exist, and the consumer instance failed to be obtained. This may be because the consumer instance is released 1 minute after the message is consumed. In this case, the consumer instance cannot be obtained from the handler. This error code is returned in the following scenarios: ● The consumer submits the handler of another consumer instance after the message is consumed. ● The message consumption is acknowledged 1 minute after the message is consumed. ● A dead letter message consumption API is used to acknowledge the consumption of common messages. ● A common message consumption API is used to acknowledge the consumption of dead letter messages.
400	10542206	The value of ack_wait must be between 15 and 300.
400	10542209	The handler does not exist because the handler fails to be parsed, the message consumption times out, or the message consumption is repeatedly acknowledged. This error code is returned in the following scenarios: ● A wrong handler is submitted after the message is consumed. ● The message is not consumed and a fake handler is directly used to acknowledge the message consumption. ● The message consumption is acknowledged 30s after the message is consumed. ● A used handler is used to repeatedly acknowledge message consumption within 30s after the message is consumed. Due to performance concerns, the system timeout timing mechanism is not very precise, and the system performs periodical timeout detection instead of real-time detection.
400	10542214	Invalid request format: {desc}.
404	10240407	Flow control.., please try later.
404	10540401	Topic not found, topicName={xxx}

5 Change History

Released On	What's New
2018-07-03	This issue is the tenth official release, which incorporates the following changes: ● Revised description in the previous issue.
2018-01-19	This issue is the ninth official release, which incorporates the following changes: ● Revised description in the previous issue.
2017-10-19	This issue is the eighth official release, which incorporates the following changes: ● Revised description in the previous issue.
2017-07-28	This issue is the seventh official release, which incorporates the following changes: ● Added description of the **queue_mode** parameter in section **2.1 Creating a Queue**. ● Added parameters of dead letter queues in sections **2.1 Creating a Queue**, **2.2 Viewing All Queues**, and **2.3 Viewing a Specified Queue**. ● Added sections **2.12 Consuming Dead Letter Messages** and **2.13 Acknowledging the Consumption of Specified Dead Letter Messages**.
2017-04-27	This issue is the sixth official release, which incorporates the following changes: ● Added description of the **time_wait** parameter in section **2.9 Consuming Messages**.
2017-03-30	This issue is the fifth official release, which incorporates the following changes: ● Updated the hyperlink to **Regions and Endpoints** in section **1.1 Service Usage**.

Released On	What's New
2017-02-27	This issue is the fourth official release, which incorporates the following changes: ● Revised description in the previous issue.
2017-02-07	This issue is the third official release, which incorporates the following changes: ● Updated description in section **4.1 Error Code Description**.
2017-01-19	This issue is the second official release, which incorporates the following changes: ● Added section **4.1 Error Code Description**. ● Revised description in the previous issue.
2016-12-30	This issue is the first official release.