

TDMQ for CMQ

Development Guide

Product Documentation



Copyright Notice

©2013–2026 Tencent Cloud. All rights reserved.

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

Trademark Notice



All trademarks associated with Tencent Cloud and its services are owned by the Tencent corporate group, including its parent, subsidiaries and affiliated companies, as the case may be. Trademarks of third parties referred to in this document are owned by their respective proprietors.

Service Statement

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

Contents

Development Guide

HTTP Endpoint Subscription

General References

Parameter Differences

Development Guide

HTTP Endpoint Subscription

Last updated: 2024-01-03 10:20:35

Delivery Description

CMQ can push topic messages to the subscribed HTTP endpoint by sending POST requests. The message can be in JSON or SIMPLIFIED format.

- JSON format: The body of the pushed HTTP request contains the message body and attribute information. The `Content-type` is `text/plain`.
- SIMPLIFIED format: The body of the pushed HTTP request is the message body, while information such as `msgId` will be delivered to the subscriber in the HTTP request header.

If the HTTP service of the subscriber returns a standard 2xx response code (such as 200), the delivery succeeded; otherwise, the delivery failed, and the delivery retry policy has been triggered. If no response is returned after the timeout period elapses, CMQ will also consider the request as a failure and trigger the delivery retry policy. The detection timeout period is about 15 seconds.

HTTP Delivery Request Headers

Parameter	Description
<code>x-cmq-request-id</code>	<code>requestId</code> of the current push message
<code>x-cmq-message-id</code>	<code>msgId</code> of the current push message
<code>x-cmq-message-tag</code>	Tag of the current push message

HTTP Delivery Request Body

The body of an HTTP request in JSON format contains the message body and attribute information.

Parameter	Type	Description
<code>TopicOwner</code>	String	<code>APPID</code> of the owner of the subscribed topic
<code>topicName</code>	String	Topic name
<code>subscriptionName</code>	String	Subscription name
<code>msgId</code>	String	Message ID
<code>msgBody</code>	String	Message body

publishTime	Int	Message published time
-------------	-----	------------------------

The body of an HTTP request in SIMPLIFIED format is the message body published by the publisher.

HTTP Delivery Request Response

If a 2xx response code is returned, the HTTP service of the subscriber has normally processed the delivered message; if another response code is returned or no response is returned after the timeout period elapses, an error will be reported, and the delivery retry policy will be triggered.

Sample Request

In this sample, the subscribed HTTP endpoint is `http://test.com/cgi`.

JSON format:

```
POST /cgi HTTP/1.1
Host: test.com
Content-Length: 761
Content-Type: text/plain
User-Agent: Qcloud Notification Service Agent
x-cmq-request-id: 2394928734
x-cmq-message-id: 6942316962
x-cmq-message-tag: a, b

{"TopicOwner":100015036,"topicName":"MyTopic","subscriptionName":"mysubscription","msgId":"6942316962","msgBody":"test message","publishTime":11203432}
```

SIMPLIFIED format:

```
POST /cgi HTTP/1.1
Host: test.com
Content-Length: 123
Content-Type: text/plain
User-Agent: Qcloud Notification Service Agent
x-cmq-request-id: 2394928734
x-cmq-message-id: 6942316962
x-cmq-message-tag: a, b

test message
```

Sample Message Subscription

The following is a message subscription demo, where all messages are delivered in the POST method; therefore, you only need to rewrite the `do_POST` method.

This sample will print the received HTTP POST request content and deserialize `post_data json` to traverse the printed request data.

```
#!/usr/bin/python
from BaseHTTPServer import HTTPServer, BaseHTTPRequestHandler
import json
class TestHTTPHandle(BaseHTTPRequestHandler):
    def do_POST(self):
        content_len = int(self.headers.getheader('content-length',0))
        post_body = self.rfile.read(content_len)
        print "receive cmq topic publisher request:"
        print self.headers
        print post_body
        post_data = json.loads(post_body)
        for k,v in post_data.iteritems():
            print "key:%s value:%s" % (k,v)
        #response http status 200
        self.send_response(200)
        self.end_headers()
        self.wfile.write('ok')
def start_server(port):
    http_server = HTTPServer(('0.0.0.0', int(port)),TestHTTPHandle)
    http_server.serve_forever()
if __name__ == '__main__':
    start_server(80)
```

General References

Parameter Differences

Last updated: 2024-01-03 10:20:36

Parameter Differences Between TDMQ for CMQ and CMQ

The usage and syntax of the data flow (messaging) SDK of TDMQ for CMQ remain unchanged, but some parameters and features are different from those of CMQ. Given these differences, TDMQ for CMQ has pertinent parameters configured specially to ensure that the original production and consumption logic will stay the same after migration. However, we recommend you configure new queues and topics based on the logic of TDMQ for CMQ.

Message lifecycle

TDMQ for CMQ adopts the message lifecycle model commonly used in the industry, where "Max Message Unack Time" (time to live, TTL) is used together to avoid high message queue load caused by excessive heap of messages. When the utilization of the capacity for heaped messages reaches 100%, writes will be impossible.

Compared with CMQ, TDMQ for CMQ has a new configuration item of maximum message unacknowledged time ranging from 30 seconds to 12 hours. If the consumer client fails to acknowledge a received message within this time period, the server will automatically acknowledge the message.

Unacknowledged messages will be stored in the message queue and will not be deleted. Acknowledged messages are subject to the message rewindable time and rewindable disk space. If message rewind is not enabled for the queue, messages will be deleted directly once acknowledged.

TDMQ for CMQ removes the limit on message lifecycle. By default, messages can no longer be heaped in the queue for a long time. If there are special needs, you can enable message rewind and set the rewindable time range. Only messages with message rewind enabled can be stored in the message queue for more than 12 hours. Once enabled, this feature incurs storage fees as detailed in TDMQ for CMQ's billing overview.

Note:

Queues migrated from CMQ to TDMQ for CMQ will inherit the message lifecycle within the maximum message unacknowledged time to ensure that existing businesses run properly. The period can range from 30 seconds to 12 hours when you adjust it later. Pay attention to the client processing logic when adjusting it.

Message heap limit

TDMQ for CMQ removes the limit on the number of heaped messages. In theory, an unlimited number of messages can be heaped as long as the storage capacity is sufficient. However, from the perspective of

hardware, we allocate a maximum capacity for heaped message of 10 GB to each queue. You can configure alarms in TCOP based on this value.

Generally, about 10 million messages with an average size of 1 KB each can be heaped. If you expect that the heap may exceed the upper limit after migration, [submit a ticket](#) for assistance.

Message size

TDMQ for CMQ no longer supports setting the maximum message size. In order not to affect the business migration from CMQ, the original maximum message size of 1,024 KB is set for all new queues.

Note:

You cannot set the message size for queues migrated from CMQ. If you need to add a limit, create new queues in TDMQ for CMQ.

Long polling wait time for message receipt

The meaning of this parameter remains the same, but its effect is different in TDMQ for CMQ, where we recommend you keep it below 3s.

In TDMQ for CMQ, if the long polling wait time for message receipt is too long, as the underlying layer needs to ensure the semantics of "consumption at least once", the repetition rate of message delivery may increase significantly, causing a severe impact on some downstream services without message deduplication implemented. Therefore, if you want to reduce the probability of message duplication, set this parameter as small as possible (3 seconds is recommended, as basically no repeated delivery will occur under this value).

Capacity for unacknowledged messages

TDMQ for CMQ has a limit on the capacity for unacknowledged messages, ensuring that the memory usage of the message queue server is controlled for guaranteed stability. If the client fails to acknowledge messages in a timely manner, an excessive number of invisible messages are generated. This metric has a monitoring chart. If it surges, check whether the consumer's acknowledgment and deletion logic works properly. If the capacity is insufficient under the surge, [submit a ticket](#) for assistance.