消息队列 CMQ 版

开发指南

产品文档

腾讯云

文档目录

开发指南
HTTP Endpoint 订阅

最近更新时间：2024-01-03 10:20:35

投递描述

CMQ 通过发送 POST 请求将主题消息推送到订阅的 HTTP Endpoint 端，消息格式支持两种：JSON 格式和 SIMPLIFIED 精简格式。
JSON 格式：推送的 HTTP 请求 Body 包含消息的正文和消息的属性信息。**Content-type 为 text/plain**。
SIMPLIFIED 格式：推送的 HTTP 请求 Body 即为消息正文，而 msgId 等信息会在 HTTP 请求 Header 中传递给订阅方。
订阅方的 HTTP 服务返回标准的 2xx 响应（如200），代表投递成功，否则为投递失败，并触发重试投递策略；超时不响应，CMQ 也会认为失败，同样触发重试投递策略，检测超时时间约为15秒。

投递的 HTTP 请求 Header

参数名称	描述
x-cmq-request-id	此次推送消息的 requestId
x-cmq-message-id	此次推送消息的 msgId
x-cmq-message-tag	此次推送消息的消息标签

投递的 HTTP 请求 Body

在 JSON 格式下，HTTP Body 包含消息的正文和消息的属性信息。

参数名称	类型	描述
TopicOwner	String	被订阅的主题拥有者的 APPID
topicName	String	主题名称
subscriptionName	String	订阅名称
msgId	String	消息 ID

msgBody	String	消息正文
publishTime	Int	消息的发布时间

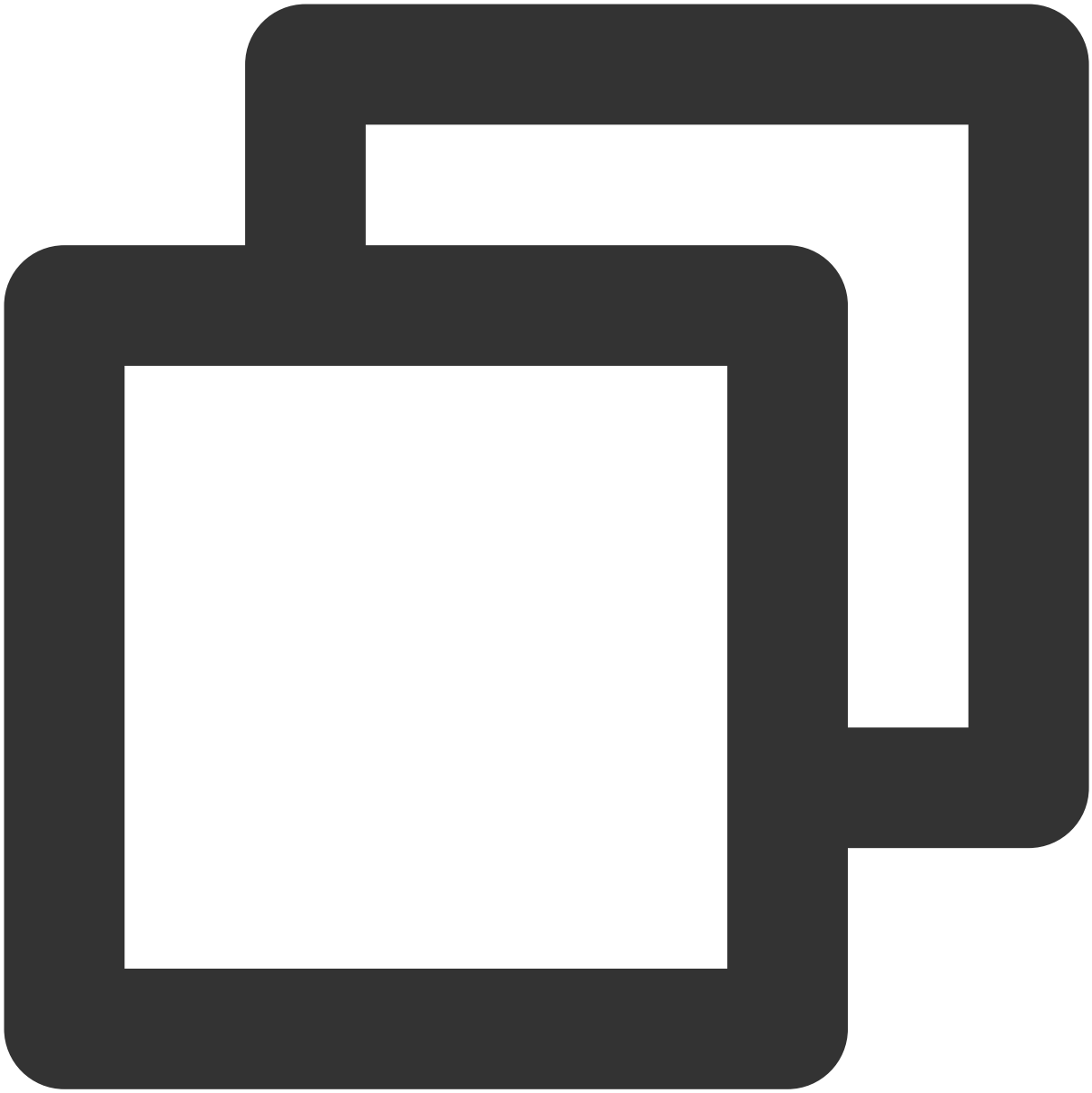
在 SIMPLIFIED 格式下，HTTP Body 即为发布者发布的消息正文。

投递的 HTTP 请求响应

订阅方 HTTP 服务正常处理投递消息，返回 2xx 响应；其他响应码或者超时不响应，当作错误，并触发重试投递策略。

请求示例

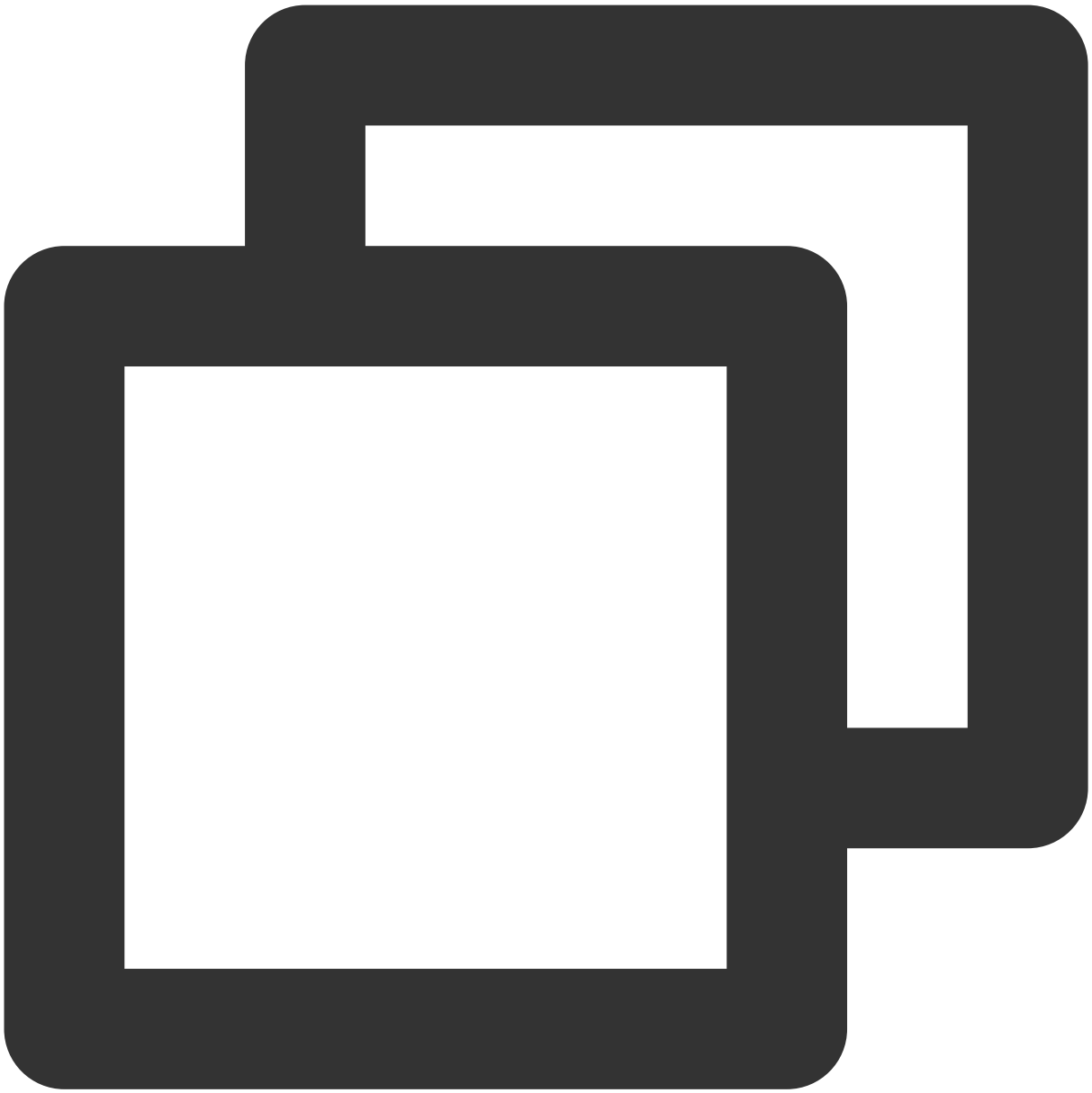
本示例中，订阅的 HTTP Endpoint 为 `http://test.com/cgi` 。
JSON 格式：



```
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","
```

SIMPLIFIED 格式：

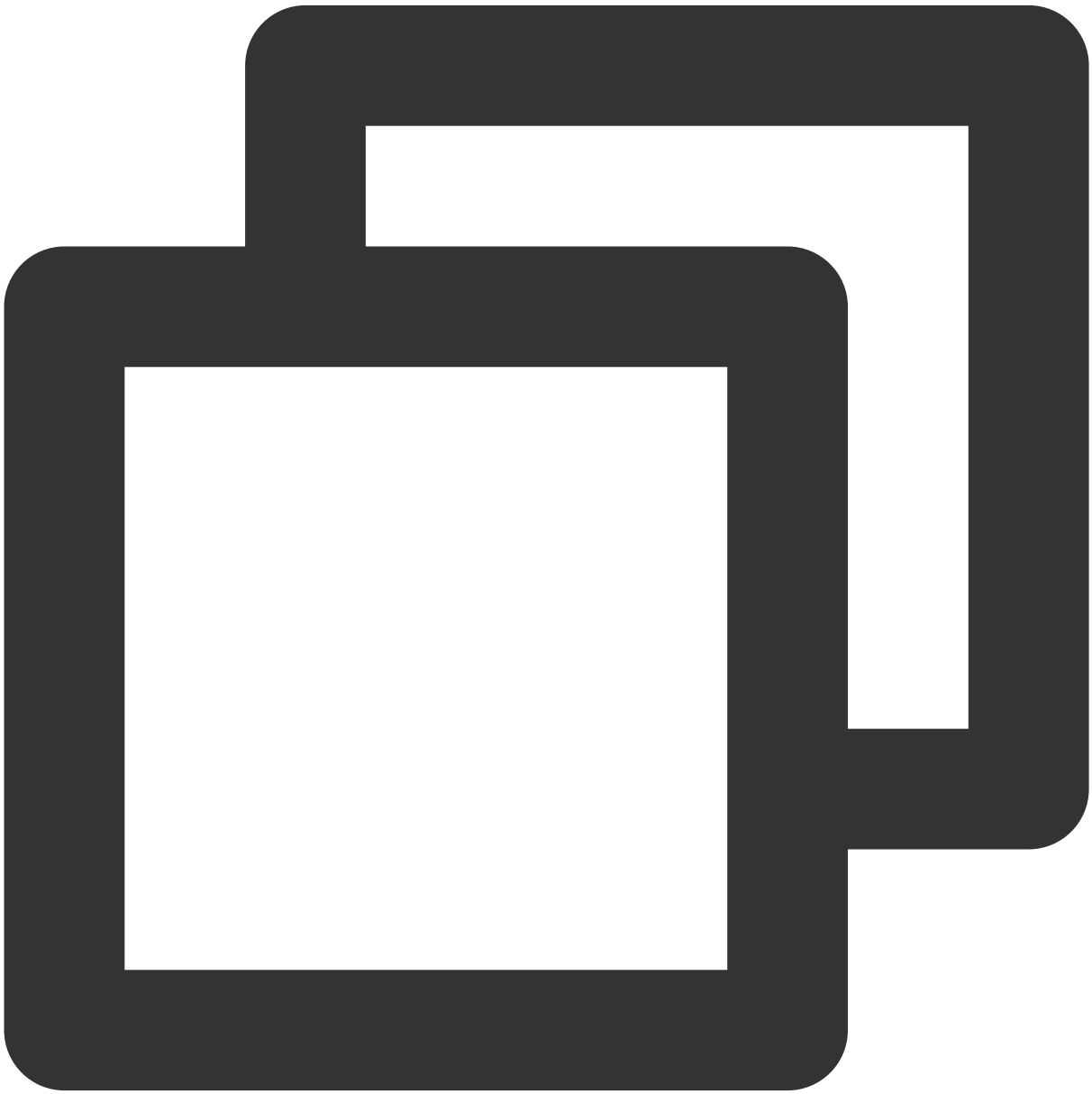


```
POST /cgi HTTP/1.1
Host: test.com
Content-Length: 123
Content-Type: text/plain
User-Agent: Qcloud Notification Service Agent
x-cmq-request-id: 2394928734
x-cmq-message-id: 6942316962
x-cmq-message-tag: a, b
```

```
test message
```

订阅消息示例

如下为订阅消息 Demo，投递消息均为 POST 方法，因此只需重写 do_POST 方法即可。
本示例会打印接收到的 HTTP POST 请求内容，并将 post data json 反序列化，实现遍历打印请求数据。



```
#!/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)
```

通用参考
参数差异说明

最近更新时间：2024-01-03 10:20:35

新版 CMQ 与原 CMQ 参数差异说明

新版 CMQ 在数据流（消息收发）SDK 的用法和语法上与原 CMQ 一致，但有些参数与特性会和原 CMQ 有一定的差异。这些差异新版 CMQ 会通过特殊设置这些参数来保证在您迁移之后不会改变原有的生产消费逻辑，但如果是新建的队列或主题则尽可能参考新 CMQ 的逻辑进行设置。

消息生命周期

新版 CMQ 采用了业界通用的消息生命周期模型，即通过“最长未确认时间”(TTL, Time to Live) 来避免产生过量的堆积导致消息队列负载过高（消息堆积容量达到100%以上时会触发不可写入）。
相较于原版 CMQ，新版 CMQ 增设了消息最大未确认时间，范围30秒到12小时，如果消息在发送成功后，超过此时间消息仍然未被确认（ack），则服务端会自动确认该消息。
未确认的消息将持续保存在 MQ 中不会删除，已确认的消息受到消息可回溯时间大小和可回溯磁盘空间的作用（如果消息所在的队列没有打开消息回溯开关，则消息会在确认后直接删除）。
新版 CMQ 取消了消息生命周期的限制。默认不再支持消息长时间在队列中堆积，如有特殊需要可以开启消息回溯并设置可回溯的时间范围，只有开启了消息回溯的消息才允许在消息队列中保存超过12小时，开启后会产生一定的存储费用，具体计费请参考新版 CMQ 计费说明。
说明：
从原 CMQ 迁移到新 CMQ 的队列会将消息生命周期继承到消息最大未确认时间，以确保原有业务正常，下次手动调整不得超出30秒 - 12小时范围，调整时请留意客户端相关的处理逻辑。

消息堆积上限

新版 CMQ 取消了原版 CMQ 关于消息堆积条数的限制，理论上只要存储资源满足，可以无限堆积。但实际从硬件层面出发，我们会给每个队列分配10GB的最大堆积存储资源，并支持您通过该值配置对应的腾讯云可观测平台告警。一般平均1KB大小的消息可以堆积大约1千万条，可以依据此进行简单换算，如发现迁移后此处可能存在堆积超过上限的风险，可以及时通过 工单 与我们联系。

消息大小

新版 CMQ 不再支持设置消息大小上限，为不影响业务从原 CMQ 迁移，新增队列统一设定为原 CMQ 的消息大小上限，即1024KB。
说明：
从原 CMQ 迁移过来的队列不再支持设置消息大小，如需增加限制，请重新创建队列使用。

消息接收长轮询等待时间

参数意义完全相同，但是其作用效果在新版 CMQ 和原 CMQ 上有所不同，新版中该参数推荐设置到3s以下。
在新版 CMQ 中，如果消息接收长轮询等待时间设置的过大，由于底层需要保证“至少一次”的语义，可能导致消息投递的重复率显著增高，从而对于一些未做消息去重的下游业务系统产生较大影响。因此，如果希望减少消息重复的概率，可以尽量设置的小一些，推荐3秒，3秒以下基本不会产生重复投递。

未确认消息容量

新版 CMQ 增设了未确认消息容量的限制，这样可以保障 MQ 服务端的内存消耗得到控制从而确保稳定性。不可见消息过多一般是客户端未及时 ACK 导致的，该指标有对应的监控图表进行监控，如有明显突增，请尽快检查消费者的确认删除逻辑是否正常运行，如突发性容量不足请尽快 提交工单 申请。