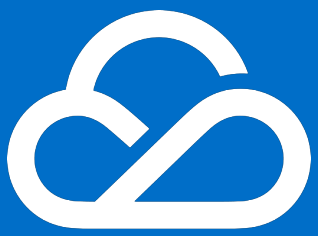
腾讯云消息队列 IoT MQ

接入示例

产品文档



腾讯云

【版权声明】

【商标声明】

【服务声明】

本文档意在向客户介绍腾讯云全部或部分产品、服务的当时的整体概况，部分产品、服务的内容可能有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或模式的承诺或保证。

文档目录

接入示例

Java 接入示例

本示例使用 Eclipse Paho Java SDK 接入 消息队列 IoT MQ。

环境准备

本示例使用 Maven 构建。如果您使用其他方式构建，可前往 使用入门 查看对应的 SDK 下载。

pom.xml 配置

在项目的 pom.xml 文件中配置 mqttv3 依赖，配置如下：

```
<dependencies>
  <dependency>
  <groupId>org.eclipse.paho</groupId>
  <artifactId>org.eclipse.paho.client.mqttv3</artifactId>
  <version>1.1.0</version>
  </dependency>
</dependencies>

<repositories>
  <repository>
  <id>Eclipse Paho Repo</id>
  <url>https://repo.eclipse.org/content/repositories/paho-releases/</url>
  </repository>
  <repository>
  <id>snapshots-repo</id>
  <url>https://oss.sonatype.org/content/repositories/snapshots</url>
  <releases>
  <enabled>false</enabled>
  </releases>
  <snapshots>
  <enabled>true</enabled>
```

```
  </snapshots>
  </repository>
</repositories>
```

计算签名

客户端签名计算方法详见 客户端签名计算。签名计算辅助类代码如下：

```
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.io.UnsupportedEncodingException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;


public class Signature {

  private static char[] b64c = new char[]{'A', 'B', 'C', 'D',
  'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q',
  'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', 'a', 'b', 'c', 'd',
  'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q',
  'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', '0', '1', '2', '3',
  '4', '5', '6', '7', '8', '9', '+', '/'};

  private static final String CONTENT_CHARSET = "UTF-8";

  // base64 编码计算，如果使用 java8，可直接使用 java.util.Base64 计算
  private static String base64_encode(byte[] data) {
  StringBuilder sb = new StringBuilder();
  int len = data.length;
  int i = 0;
  int b1, b2, b3;
  while (i < len) {
```

```
b1 = data[i++] & 0xff;
if (i == len) {
sb.append(b64c[b1 >>> 2]);
sb.append(b64c[(b1 & 0x3) << 4]);
sb.append("==");
break;
}
b2 = data[i++] & 0xff;
if (i == len) {
sb.append(b64c[b1 >>> 2]);
sb.append(b64c[((b1 & 0x03) << 4)
| ((b2 & 0xf0) >>> 4)]);
sb.append(b64c[(b2 & 0x0f) << 2]);
sb.append("=");
break;
}
b3 = data[i++] & 0xff;
sb.append(b64c[b1 >>> 2]);
sb.append(b64c[((b1 & 0x03) << 4)
| ((b2 & 0xf0) >>> 4)]);
sb.append(b64c[((b2 & 0x0f) << 2)
| ((b3 & 0xc0) >>> 6)]);
sb.append(b64c[b3 & 0x3f]);
}
return sb.toString();
}

// 计算签名
public static String signature(String src, String key, String method)
throws NoSuchAlgorithmException, UnsupportedEncodingException, InvalidKeyException {
Mac mac = Mac.getInstance(method);
SecretKeySpec secretKey = new SecretKeySpec(key.getBytes(CONTENT_CHARSET),
mac.getAlgorithm());
mac.init(secretKey);
```

```
  byte[] digest = mac.doFinal(src.getBytes(CONTENT_CHARSET));
  return base64_encode(digest);
  }

  // 以下配置项通过控制台获得
  final static String secretKey ="Gu5t9xGARNpq86cd98joQYCN3Cozk1qA";
  final static String appId = "1251762227";
  final static String instanceId = "mqtt-fludu2t6";

  // 计算发起连接使得签名
  public static String signature()
  throws NoSuchAlgorithmException, InvalidKeyException, UnsupportedEncodingException {
  String src = "Appid=" + appId + "&Instanceid=" + instanceId + "&Action=Connect";
  return signature(src, secretKey, "HmacSHA256");
  }
}
```

订阅消息

```
import org.eclipse.paho.client.mqttv3.*;
import org.eclipse.paho.client.mqttv3.persist.MemoryPersistence;

import java.io.IOException;

public class Subscribe {
  public static void main(String[] args) throws IOException {
  // SecretID 和 SecretKey 通过云 API 密钥获取
  final String secretId ="AKIDz8krbsJ5yKBZQpn74WFkmLPx3gnPhESA";
  // endpoint 通过 IoT MQ 控制台获取
  final String endpoint = "tcp://mqtt-fludu2t6.bj.mqtt.myqcloud.com:1883";
  // Topic 需先在 IoT MQ 控制台创建
  final String topic = "test/mobile/007";
  // clientId 需以“InstanceID@”开头，且需保证全局唯一
```

```
final String clientId = "mqtt-fludu2t6@579863256";

MemoryPersistence persistence = new MemoryPersistence();
try {
final MqttClient sampleClient = new MqttClient(endpoint, clientId, persistence);
final MqttConnectOptions connOpts = new MqttConnectOptions();
System.out.println("Connecting to : " + endpoint);

String sign = Signature.signature();
final String[] topicList = new String[]{topic};
final int[] qos = {0};
connOpts.setUserName(secretId);
connOpts.setServerURIs(new String[]{endpoint});
connOpts.setPassword(sign.toCharArray());
connOpts.setCleanSession(true);
connOpts.setKeepAliveInterval(30);
sampleClient.setCallback(new MqttCallback() {
public void connectionLost(Throwable throwable) {
System.out.println("connection lost");
throwable.printStackTrace();
while (!sampleClient.isConnected()) {
try {
Thread.sleep(1000);
sampleClient.connect(connOpts);
sampleClient.subscribe(topicList, qos);
} catch (Exception e) {
e.printStackTrace();
}
}
}

public void messageArrived(String topic, MqttMessage mqttMessage) throws Exception {
System.out.println("messageArrived:" + topic + "---" + new String(mqttMessage.getPayload()));
}
```

```
  public void deliveryComplete(IMqttDeliveryToken iMqttDeliveryToken) {
  System.out.println("deliveryComplete:" + iMqttDeliveryToken.getMessageId());
  }
  });
  sampleClient.connect(connOpts);
  sampleClient.subscribe(topicList, qos);
  Thread.sleep(Integer.MAX_VALUE);
  } catch (Exception me) {
  me.printStackTrace();
  }
  }
}
```

发送消息

```
import org.eclipse.paho.client.mqttv3.*;
import org.eclipse.paho.client.mqttv3.persist.MemoryPersistence;

import java.io.IOException;

public class Publish {

  public static void main(String[] args) throws IOException {
  // SecretID 和 SecretKey 通过云 API 密钥获取
  final String secretId ="AKIDz8krbsJ5yKBZQpn74WFkmLPx3gnPhESA";
  // endpoint 通过 IoT MQ 控制台获取
  final String endpoint = "tcp://mqtt-fludu2t6.bj.mqtt.myqcloud.com:1883";
  // Topic 需先在 IoT MQ 控制台创建
  final String topic = "test/mobile/007";
  // clientId 需以“InstanceID@”开头，且需保证全局唯一
  final String clientId = "mqtt-fludu2t6@579863256";
```

```
MemoryPersistence persistence = new MemoryPersistence();
try {
final MqttClient sampleClient = new MqttClient(endpoint, clientId, persistence);
final MqttConnectOptions connOpts = new MqttConnectOptions();
System.out.println("Connecting to : " + endpoint);

String sign = Signature.signature();
connOpts.setUserName(secretId);
connOpts.setServerURIs(new String[]{endpoint});
connOpts.setPassword(sign.toCharArray());
connOpts.setCleanSession(true);
connOpts.setKeepAliveInterval(30);
sampleClient.setCallback(new MqttCallback() {
public void connectionLost(Throwable throwable) {
System.out.println("connection lost");
while (!sampleClient.isConnected()) {
try {
Thread.sleep(1000);
sampleClient.connect(connOpts);
} catch (Exception e) {
e.printStackTrace();
}
}
}

public void messageArrived(String topic, MqttMessage mqttMessage) throws Exception {
System.out.println("messageArrived:" + topic + "------" + new String(mqttMessage.getPayload()));
}

public void deliveryComplete(IMqttDeliveryToken iMqttDeliveryToken) {
System.out.println("deliveryComplete:" + iMqttDeliveryToken.getMessageId());
}
});
sampleClient.connect(connOpts);
```

```
  for (int i = 0; i < 10; i++) {
  try {
  String scontent = "IoT MQ demo " + i;
  final MqttMessage message = new MqttMessage(scontent.getBytes());
  message.setQos(0);
  System.out.println("publish msg: " + scontent);
  sampleClient.publish(topic, message);
  } catch (Exception e) {
  e.printStackTrace();
  }
  }
  } catch (Exception me) {
  me.printStackTrace();
  }
  }
}
```