

ALIBABA CLOUD

# 阿里云

## 微消息队列MQTT版 MQTT 客户端权限验证

文档版本：20210202

 阿里云

## 法律声明

阿里云提醒您,在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档,您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档,且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息,您应当严格遵守保密义务;未经阿里云事先书面同意,您不得向任何第三方披露本手册内容或提供给任何第三方使用。
2. 未经阿里云事先书面许可,任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部,不得以任何方式或途径进行传播和宣传。
3. 由于产品版本升级、调整或其他原因,本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利,并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。
4. 本文档仅作为用户使用阿里云产品及服务的参考性指引,阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引,但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的,阿里云不承担任何法律责任。在任何情况下,阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害,包括用户使用或信赖本文档而遭受的利润损失,承担责任(即使阿里云已被告知该等损失的可能性)。
5. 阿里云网站上所有内容,包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计,均由阿里云和/或其关联公司依法拥有其知识产权,包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意,任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外,未经阿里云事先书面同意,任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称(包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌,上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司)。
6. 如若发现本文档存在任何错误,请与阿里云取得直接联系。

# 通用约定

格式	说明	样例
 危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	 危险 重置操作将丢失用户配置数据。
 警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	 警告 重启操作将导致业务中断，恢复业务时间约十分钟。
 注意	用于警示信息、补充说明等，是用户必须了解的内容。	 注意 权重设置为0，该服务器不会再接受新请求。
 说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	 说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击设置> 网络> 设置网络类型。
<b>粗体</b>	表示按键、菜单、页面名称等UI元素。	在结果确认页面，单击 <b>确定</b> 。
Courier字体	命令或代码。	执行 <code>cd /d C:/window</code> 命令，进入Windows系统文件夹。
<i>斜体</i>	表示参数、变量。	<code>bae log list --instanceid</code> <i>Instance_ID</i>
[ ] 或者 [a b]	表示可选项，至多选择一个。	<code>ipconfig [-all -t]</code>
{ } 或者 {a b}	表示必选项，至多选择一个。	<code>switch {active stand}</code>

# 目录

1.鉴权概述	05
2.签名鉴权模式	08
3.Token 鉴权模式	10
3.1. Token 鉴权概述	10
3.2. Token 应用服务器接口	11
3.3. Token 客户端接口	12
4.一机一密鉴权模式	15
4.1. 一机一密概述	15
4.2. 一机一密应用服务器接口	16

# 1. 鉴权概述

本文主要介绍微消息队列MQTT版客户端鉴权的原理和分类，以便您使用相应的鉴权功能。

## 鉴权原理

使用微消息队列MQTT版的客户端收发消息时，服务端会根据MQTT客户端设置的UserName和Password参数来进行鉴权。针对不同的权限验证场景，UserName和Password参数具备不同的含义。MQTT客户端开发应该根据实际场景选择合适的鉴权模式，按照对应的规范计算正确的UserName和Password。

各鉴权模式下UserName和Password的计算方法和应用场景如下表所示。

权限验证模式	模式名称	UserName	Password	适用场景
签名验证	Signature	由鉴权模式名称、AccessKey ID、Instance ID三部分组成，以竖线 ( ) 分隔。  举例：一个客户端的ClientId是GID_Test@@@0001，使用了实例ID是mqtt-xxxxx，使用了AccessKey ID是YYYYY，则签名模式的UserName应该设置成“Signature YYYYY mqtt-xxxxx”。	对Client ID签名的Base64编码结果，具体设置方法，请参见 <a href="#">签名鉴权模式</a> 。	永久授权，适用于客户端安全受信任的场景。
临时Token权限验证	Token		上传的Token内容，具体设置方法，请参见 <a href="#">Token鉴权概述</a> 。	临时授权，适用于客户端不安全的场景。

权限验证模式	模式名称	UserName	Password	适用场景
一机一密验证	DeviceCredential	<p>由鉴权模式名称、DeviceAccessKeyId、Instance ID三部分组成，以竖线（ ）分隔。</p> <p>举例：一个客户端的ClientId是GID_Test@@@0001，使用了实例ID是mqtt-xxxxx，DeviceAccessKeyId是YYYYY，则一机一密模式的UserName应该设置成“DeviceCredential YYYYY mqtt-xxxxx”。</p> <p>DeviceAccessKeyId为MQTT服务端为设备颁发的访问凭证中的参数，详细信息，请参见<a href="#">一机一密概述</a>。</p>	对Client ID签名的Base64编码结果，具体设置方法，请参见 <a href="#">一机一密概述</a>	永久授权，适用于客户端安全受信的场景区，并且服务端可以随时更新或禁用已发放的设备访问凭证。

## 鉴权模式分类和比较

按照上文分类，微消息队列MQTT版目前支持了签名校验、Token校验及一机一密校验三种验证方式，这三种方式覆盖了固定权限以及非固定的临时权限的场景，具体细分和适用场景分析如下所述：

- 签名模式（固定权限）

签名模式是微消息队列MQTT版推荐使用的默认鉴权模式，该模式下约定同一类型的MQTT客户端拥有同样的权限范围，这些客户端使用相同的账号来做签名计算，服务端通过签名比对即可识别出客户端对应的账号以及拥有的权限范围。该模式理解成本低，使用简单。

- 适用场景

使用的MQTT客户端逻辑上可以划分为同一类，逻辑意义上都属于一个账号，具备同样的权限范围，且每个MQTT客户端运行的环境相对可控，不需要考虑设备被破解盗用等恶意攻击的场景。

因为在业务层面上，对一类的MQTT客户端都划分到一个账号（可以是主账号或者子账号），所以客户端只需要根据对应的账号AccessKey Secret计算签名即可，权限范围由账号的管理员在产品的控制台进行统一管理。

- 签名计算

为防止签名被盗用，需要取每个客户端独立的信息进行签名计算，微消息队列MQTT版约定每个客户端使用自己独一无二的ClientId来作为待签名内容。关于该类型签名的具体计算方法，请参见[签名鉴权模式](#)。

- Token模式（临时权限）

如果业务上需要对每个MQTT客户端的权限进行细致划分，或者仅需要对客户端授予临时的有时间期限的权限，则可以通过Token模式这种临时凭证访问方式实现。通过Token服务，您可以设置单一客户端访问的资源内容、权限级别和权限过期时间。

关于Token鉴权的流程和注意事项，请参见[Token鉴权概述](#)。

#### ○ 适用场景

业务方拥有自己独立的本地账号系统，需要对阿里云账号的身份做二次拆分，甚至需要对每个MQTT客户端分配独立的个体账号和权限范围。在这种情况下MQTT客户端的细分使用阿里云的RAM账号系统无法满足。

因为运行的MQTT客户端的业务虽然隶属于同一个阿里云账号，但是还需要有本地账号（业务部门或者单一客户端）的角色区分。此时签名模式下划定的阿里云账号维度就无法满足。特别是在移动端场景，客户端如果需要考虑破解劫持的风险，固定的权限模式管理相对比较受限，如果权限控制需要细化到单一客户端，且权限粒度需要细化到单一资源，所有的权限都是临时的非固定的，则更加灵活。

#### ○ Token使用

使用Token模式相对比较复杂，按照上文描述，需要业务方自己拥有账号（设备）管理能力，业务方需要自己管理每个设备的权限范围和时效性，由业务方在安全可控的管理节点来申请Token并发放给MQTT客户端使用。具体的使用方法，请参见[Token鉴权概述](#)。

### ● 一机一密模式

一机一密给予MQTT客户端能够以独立的用户名、密码来进行身份识别的能力，帮助您解决客户端在不安全场景下，存在Token冒认的问题。且在一机一密模式下，用户名、密码与ClientId绑定，您可独立管理。

#### ○ 适用场景

业务方对安全性要求较高，且类似于IoT这种不便实时更新Token的场景，一机一密能够提供持久化验证能力，不需要频繁的被动更新访问凭证。

#### ○ 设备访问凭证使用

一机一密模式下，客户端应用服务器为每个客户端申请独一无二的设备访问凭证，客户端向微消息队列MQTT版发起认证请求时，按照约定的规范对访问凭证中的信息进行计算并作为连接参数，具体的计算方法，请参见[一机一密概述](#)。

## 2. 签名鉴权模式

如果您需使用微消息队列MQTT版的签名鉴权模式，可根据本文提供的签名计算方式获取签名，再使用控制台来验证计算得出的签名是否正确。

### 签名计算方式

按照[鉴权概述](#)的描述，如果选择签名校验模式，MQTT 客户端实际连接 MQTT 消息服务器时，connect 报文中的 Username 和 Password 需要按照本文约定的规范设置，具体设置和计算方法如下。

#### • Username

由鉴权模式名称、AccessKeyId 和 InstanceId 三部分组成，以 “|” 分隔。签名模式下鉴权模式设置为 Signature。

举例：一个客户端的 Client ID 是 GID\_Test@@@0001，使用的实例 ID 是 mqtt-xxxxx，使用的 AccessKeyId 是 YYYYY，则签名模式的 Username 应该设置成 Signature|YYYYY|mqtt-xxxxx。

Client ID 的详细说明请参见[名词解释](#)。

#### • Password

对 Client ID 签名的结果。具体计算方法如下：

举例：一个客户端的 Client ID 是 GID\_Test@@@0001，使用的 AccessKeySecret 是 XXXXX。

用 XXXXX 作为密钥，使用 HMAC-SHA1 方法对待签名字符串 GID\_Test@@@0001 做签名计算得到一个二进制数组，再对该二进制数组做 Base64 编码得到最终的 Password 签名字符串。

HMAC-SHA1 的算法实现，各个语言都有现成的函数库，请自行搜索，或者参见[微消息队列 MQTT 使用签名模式收发消息](#)的示例代码中设置 Username 和 Password 的部分。

### 使用控制台验证签名

[微消息队列MQTT版控制台](#)提供了签名计算工具，以方便您比对验证自己的签名计算是否正确。

1. 登录[微消息队列MQTT版控制台](#)，在顶部菜单栏，选择地域（Region）。
2. 在左侧导航栏，单击[签名校验](#)。
3. 在[签名校验](#)页面，输入程序使用的账号的 AccessKeyId、AccessKeySecret 以及 Client ID，单击[计算签名](#)即可得到程序中需要设置的 Username 和 Password 参数。

 说明

此工具仅仅使用浏览器前端 JavaScript 完成计算，并不会传输 AccessKeySecret 到消息队列RocketMQ版

后端，因此不用担心 AccessKeySecret 泄漏的风险。实际业务中控制台仅仅是用来排查问题比对数据。签名计算可以放在客户端或者更安全的服务端计算完成后下发给 MQTT 客户端。

## 3.Token 鉴权模式

### 3.1. Token 鉴权概述

微消息队列MQTT版通过 Token 鉴权模式向 MQTT 客户端提供临时访问权限。您可通过 MQTT Token 服务来给本地账号系统管理的用户颁发临时的访问凭证，并限制访问权限，以实现单一客户端、单一资源的细粒度权限控制。本文介绍使用流程以及相关注意事项。

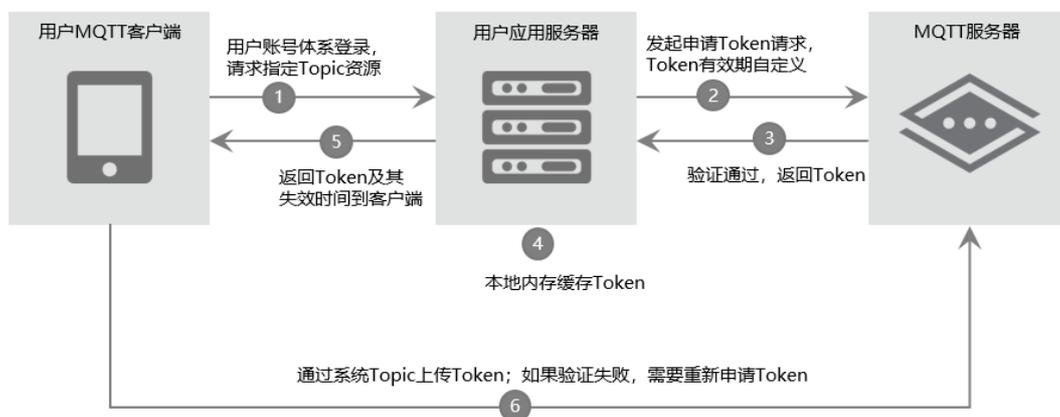
#### 名词解释

术语	说明
Token（临时凭证）	微消息队列MQTT版颁发的临时访问凭证，代表单一客户端对特定资源的访问权限。
应用服务器	您管理本地账号的服务器，用来替客户端申请和管理 Token 服务的应用。
MQTT 服务器	微消息队列MQTT版权限认证和消息收发服务器，用来处理应用服务器发起的 Token 相关的请求以及消息收发业务。

#### 使用流程

Token 鉴权模式相比签名模式更加复杂，您需要按照下图所示的流程，部署您的应用服务器。而且在初始化时，MQTT 客户端需要具备与您的应用服务器交互（获取和更新 Token）的能力。

##### 鉴权流程



具体流程如下：

1. 客户端启动时，需要先连接应用服务器验证身份。
2. 客户端向应用服务器申请所需的所有 Topic 的权限。
3. 应用服务器验证客户端是否有必要操作所需的 Topic 权限，如果验证通过则向微消息队列MQTT版服务器申请对应资源的 Token。
4. 微消息队列MQTT版服务器验证申请 Token 的请求，判断合法之后返回对应的 Token。
5. 应用服务器将返回的 Token 持久化到本地，对每个客户端需要的权限以及 Token 信息进行映射。缓存 Token 有以下优势：

- 客户端重新启动后再访问时，只要权限范围没有变化，应用服务器可以返回缓存的 Token，避免重复申请。
  - 客户端重新申请 Token 时，如果 MQTT 服务器异常，应用服务器可以尝试返回客户端之前申请的 Token 以实现本地容灾。
6. MQTT 客户端按照规范将 Token 作为参数设置，连接 MQTT 服务器，服务端验证通过后客户端即可正常收发消息。
  7. 客户端正常收发消息。如果服务端判断 Token 失效，即会触发鉴权失败，断开连接，客户端应该重新发起申请 Token 的请求。

## 客户端行为约束

- 客户端从应用服务器拿到的信息除了 Token 本身，还需要获取 Token 的失效时间，以便本地计算何时提前刷新 Token。
- 必须按照约定形式将 Token 作为连接参数设置到 Password 中，每次连接时上传。
- 客户端应该知晓自己使用的 Token 的时效性，确保不要使用过期的 Token，否则服务端会断开连接。
- 客户端可以监听服务端下推的 Token 失效的通知消息，但服务端不保证通知一定触发，该通知仅供排查问题使用。
- 客户端应该对应用服务器返回的 Token 做持久化，避免每次重连都申请一样的 Token，防止大批量客户端同时连接压垮应用服务器。
- 客户端更新 Token 可以选择关闭旧的客户端连接，然后重新使用新的 Token 来连接，也可以选择使用 MQTT 提供的系统 Topic 动态上传更新 Token。如果选择动态更新 Token，需要保证本地的配置也同步更新，以供下次连接初始化使用。

## 应用服务器行为约束

- 应用服务器必须验证自己的客户端是否合法，避免客户端冒用身份申请 Token。
- 应用服务器应该对 Token 和客户端的关系、已分配的 Token、对应权限内容以及生效时间进行管理，避免同一个客户端重复调用。
- 应用服务器将 Token 返回给客户端的同时，也应该告知客户端当前 Token 的操作权限以及过期时间，以便客户端提前刷新 Token。
- 应用服务器需要做本地容灾，避免因 MQTT 服务器访问短暂不可用而导致业务阻塞的情况。

## 相关 API

Token 鉴权流程通过相关的 API 来完成。

- 应用服务器负责 Token 的申请和吊销管理，和微消息队列MQTT版服务器之间通过 HTTPS 的 OpenAPI 进行交互。

每个接口都要求通过 AccessKey 和请求签名来做身份验证。目前开放申请 Token、吊销 Token 以及校验 Token 接口。详情请参见 [Token 应用服务器接口](#)。

- 微消息队列MQTT版客户端则包括动态更新 Token、监听 Token 失效信息，以及监听 Token 非法信息三个接口。详情请参见 [Token 客户端接口](#)。

# 3.2. Token 应用服务器接口

您的应用服务器访问认证服务器的 OpenAPI 接口时，需要遵循一定的参数和返回值获取规范，否则会无法调用接口。本文介绍微消息队列MQTT版提供的应用服务器 Token OpenAPI 接口。

API	描述
ApplyToken	调用ApplyToken申请临时访问的Token。
QueryToken	调用QueryToken校验Token的有效性。
RevokeToken	调用RevokeToken吊销Token。

### 3.3. Token 客户端接口

本文介绍微消息队列MQTT版客户端在Token模式下如何上传Token、更新Token、监听Token失效信息、监听Token非法信息。

#### Token 模式 MQTT 客户端连接参数设置

MQTT 支持三种模式的Token，每个客户端每个类型至多申请一个Token，根据实际需要可能申请其中的一种或者多种，并使用。具体的类型如下表所示。

类型标志	说明
R	只读类型的Token，只拥有指定资源的读权限
W	只写类型的Token，只拥有指定资源的写权限
RW	读写类型的Token，对指定资源既可以读也可以写

Token 模式下 MQTT 客户端的连接参数设置如下。

- Username

由鉴权模式名称、AccessKeyId、InstanceId 三部分组成，以“|”分隔。Token 模式下鉴权模式设置为“Token”。

- 举例

一个客户端的 ClientId 是 GID\_Test@@@0001，使用了实例 ID 是 mqtt-xxxxx，使用了 AccessKeyId 是 YYYYY，则 Token 模式的 Username 应该设置成 “Token|YYYYY|mqtt-xxxxx”。

- Password:

客户端需要使用的Token内容。具体设置方法是将所有的Token按照Token类型和Token内容依次使用“|”连接符拼接成一个完整的字符串，不同类型之间的Token拼接顺序无要求。

举例 1：客户端只有一个读类型的Token，Token字符串为“123”，则Password为“R|123”。

举例 2：客户端拥有 2 个类型的Token，读类型的Token是“123”，写类型的Token是“abcd”，则Password为“R|123|W|abcd”。

 **说明** 客户端设置Token参数时需要保证严格按照约定规则，且需要保证所有Token有效，如果仅用部分Token合法，服务端仍然会判定非法。

#### 客户端更新Token凭证

正常情况下客户端更换 Token 时需要断开连接重新使用新的 Token 来连接，如果业务场景中不希望由于更换 Token 中断客户端的连接，可以使用动态更新 Token 的接口来动态替换服务端 session 内的 Token 数据。

动态更新 Token 本质上是由 MQTT 客户端以约定的系统 Topic 发送一个特殊的消息，将新的 Token 内容更新到服务端，客户端需要保证在动态替换的同时，本地配置也要替换，防止下次连接初始化又使用了旧的 Token 数据。

- 更新 Token 发送 Topic: \$SYS/uploadToken
- 内容: JSONObject
- 内容信息:

名称	类型	说明
token	String	如果客户端选用 Token 模式，则需要上传 Token 字符串。
type	String	Token 类型，分为 W、R、RW 共三种，对应三种权限类型的 Token。一个客户端最多拥有这 3 个 Token，设置错误的类型会导致权限校验错误。

- 返回值

普通的 PubAck 报文。客户端必须等到该响应才能进行下一步 Pub 或者 Sub 操作，否则服务端仍然有可能使用旧的 Token 数据来做鉴权，可能会鉴权失败导致连接断开。

### 客户端监听即将失效的 Token 信息（无需订阅）

服务端为方便业务调试和监控，会在 Token 即将失效的时候以系统 Topic 的形式推送通知消息到 MQTT 客户端，客户端可以监控该消息来判断是否有出现过 Token 即将到期的情况。

- 接收 Topic: \$SYS/tokenExpireNotice
- 内容: JSONObject
- 内容信息:

名称	类型	说明
expireTime	Long	该 Token 即将于什么时候失效，格式为毫秒时间戳，一般提前 5 分钟通知，只通知一次，但服务端不保证一定会有通知。
type	String	Token 类型，分为 W、R、RW 共三种，对应客户端上传的三种权限类型的 Token。

- 响应

客户端收到 Token 即将失效的消息后，需要尽快处理重新申请 Token 的动作，以免造成收发消息失败。

### 客户端监听 Token 非法的通知（无需订阅）

服务端为方便业务调试和监控，会在 Token 鉴权错误时以系统 Topic 的形式推送通知消息到 MQTT 客户端，客户端可以监控该消息来判断是否有出现过 Token 不匹配等错误权限的情况。

- 接收 Topic: \$SYS/tokenInvalidNotice
- 内容: JSONString
- 内容信息:

名称	类型	说明
code	int	Token 校验失败的类型。
type	String	Token 类型，分为 W、R、RW 共三种，对应客户端上传的三种权限类型的 Token。

- 响应:

服务端校验 Token 失效，会导致鉴权失败，服务端会主动断开链接。断开链接之前，服务端会给客户端推送失败的 Code，客户端根据 Code 即可判断原因。

type code	错误类型
1	伪造 Token，不可解析
2	Token 已经过期
3	Token 已经被吊销
4	资源和 Token 不匹配
5	权限类型和 Token 不匹配
8	签名不合法
-1	帐号权限不合法

## 4. 一机一密鉴权模式

### 4.1. 一机一密概述

一机一密认证方法，即预先为每个设备申请唯一的访问凭证（Client Id、DeviceAccessKeyId和DeviceAccessKeySecret）。当设备与微消息队列MQTT版建立连接时，需要将设备访问凭证中的信息按照约定的形式设置到Username和Password中，向微消息队列MQTT版发起认证请求，微消息队列MQTT版认证通过后激活设备，设备与微消息队列MQTT版间才可传输数据。

#### 名词解释

术语	说明
设备访问凭证	MQTT服务器为用户客户端颁发的访问凭证，包含DeviceAccessKeyId和DeviceAccessKeySecret，与Client ID绑定并全局唯一。客户端与微消息队列MQTT版建立连接时，需要按照约定形式将设备访问凭证中的DeviceAccessKeyId和DeviceAccessKeySecret设置到Username和Password中进行认证。
应用服务器	您管理本地账号的服务器，用来替客户端申请和管理设备访问凭证服务的应用。
MQTT服务器	微消息队列MQTT版权限认证和消息收发服务器，用来处理应用服务器发起的设备访问凭证相关的请求以及消息收发业务。

#### 计算方式

按照[鉴权概述](#)的描述，如果选择一机一密鉴权模式，MQTT客户端实际连接MQTT消息服务器时，connect报文中的Username和Password需要按照本文约定的规范设置，具体设置和计算方法如下：

- Username

由鉴权模式名称、DeviceAccessKeyId和InstanceId三部分组成，以竖线（|）分隔。一机一密模式下鉴权模式设置为DeviceCredential。

举例：一个客户端的Client ID是GID\_Test@@@0001，使用的实例ID是mqtt-xxxxx，使用的DeviceAccessKeyId是YYYYY，则签名模式的UserName应该设置成 DeviceCredential|YYYYY|mqtt-xxxxx。

Client ID的详细说明请参见[名词解释](#)。

- Password

对Client ID签名的结果。具体计算方法如下：

举例：一个客户端的Client ID是GID\_Test@@@0001，使用的DeviceAccessKeySecret是XXXXX。

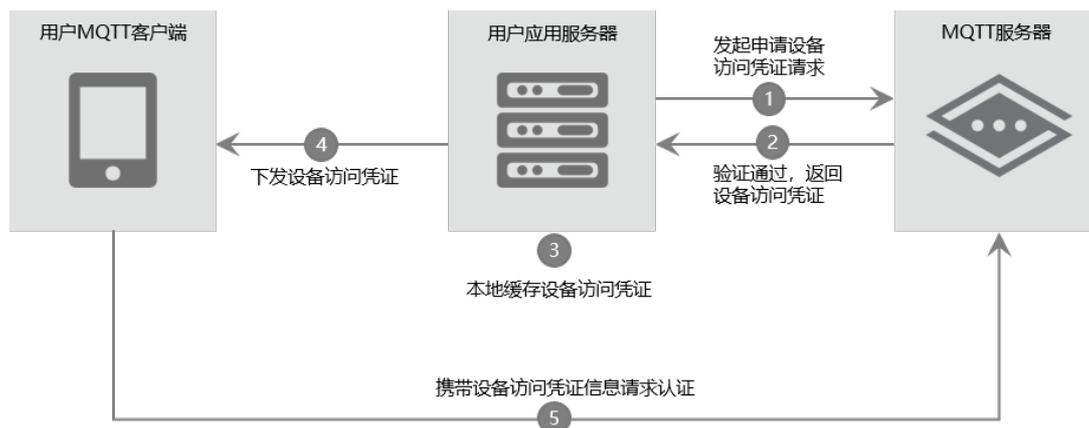
用XXXXX作为密钥，使用HMAC-SHA1方法对待签名字符串GID\_Test@@@0001做签名计算得到一个二进制数组，再对该二进制数组做Base64编码得到最终的Password字符串。

HMAC-SHA1的算法实现，各个语言都有现成的函数库，请自行搜索。

#### 使用流程

使用一机一密鉴权模式，您需要按照下图所示的流程，部署您的应用服务器。而且在初始化时，MQTT客户端需要具备与您的应用服务器交互（获取和更新设备访问凭证）的能力。

鉴权流程



具体流程如下：

1. 用户应用服务器调用OpenAPI接口向MQTT服务器发起请求，预先为用户客户端申请设备访问凭证。
2. MQTT服务器验证申请设备访问凭证的请求，判断合法之后下发对应的设备访问凭证。
3. 应用服务器将返回的设备访问凭证持久化到本地，对每个客户端对应的访问凭证进行映射。缓存设备访问凭证有以下优势：
  - 只要客户端侧没有发生凭证泄露等问题，设备访问凭证就无需更换，此时可以直接返回缓存的设备凭证，避免申请凭证时接口调用的开销。
  - 客户端重新申请设备访问凭证时，如果MQTT服务器异常，应用服务器可以尝试返回客户端之前申请的设备访问凭证以实现本地容灾。
4. 用户应用服务器将申请好的设备访问凭证下发给对应的MQTT客户端。
5. MQTT客户端按照规范将设备访问凭证中的信息作为参数设置，连接MQTT服务器，服务端验证通过后客户端即可正常收发消息。

## 客户端行为约束

- 必须按照约定形式将设备访问凭证中的信息作为连接参数设置到Username和Password中，每次连接时上传。
- 客户端应该对应用服务器返回的设备访问凭证做持久化，避免每次重连都申请一样的设备访问凭证，防止大批量客户端同时连接压垮应用服务器。

## 应用服务器行为约束

- 应用服务器应该对设备访问凭证和客户端的映射关系进行管理，避免同一个客户端重复调用。
- 应用服务器需要做本地容灾，避免因MQTT服务器访问短暂不可用而导致业务阻塞的情况。

## 相关API

一机一密鉴权流程通过相关的API来完成。应用服务器负责设备访问凭证的申请和管理，和微消息队列MQTT版服务器之间通过HTTPS的OpenAPI进行交互。

每个接口都要求通过AccessKey和请求签名来做做身份验证。目前开放注册、查询、注销和更新设备访问凭证的接口。详细接口信息，请参见[一机一密应用服务器接口](#)。

## 4.2. 一机一密应用服务器接口

您的应用服务器访问认证服务器的OpenAPI接口时，需要遵循一定的参数和返回值获取规范，否则会无法调用接口。本文介绍微消息队列MQTT版提供的设备访问凭证相关的OpenAPI接口。

API	描述
RegisterDeviceCredential	调用RegisterDeviceCredential为设备申请访问凭证。
Get DeviceCredential	调用Get DeviceCredential查询设备的访问凭证。
UnRegisterDeviceCredential	调用UnRegisterDeviceCredential注销设备的访问凭证。
RefreshDeviceCredential	调用RefreshDeviceCredential更新设备的访问凭证。