阿里云
微消息队列 MQTT

权限验证

文档版本：20191029

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. **阿里云文档中所有内容，包括但不限于图片、架构设计、页面布局、文字描述，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
	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。	 **禁止:** 重置操作将丢失用户配置数据。
	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。
	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。
	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。
粗体	表示按键、菜单、页面名称等UI元素。	在结果确认页面，单击**确定**。
`Courier字体`	命令。	执行`cd /d C:/window`命令，进入Windows系统文件夹。
*## *	表示参数、变量。	`bae log list --instanceid` *`Instance_ID`*
[]或者[a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{}或者{a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1 鉴权概述

本文主要介绍微消息队列 MQTT 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 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。

鉴权模式

模式名称	模式说明	适用场景
Signature	签名验证	永久授权，适用于客户端安全受信任的场景
Token	临时 Token 权限验证	临时授权，适用于客户端不安全的场景

Username

由鉴权模式名称、AccessKeyId、InstanceId 三部分组成，以“|”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 AccessKeyId 是 YYYYY，则签名模式的 UserName 应该设置成“Signature|YYYYY|mqtt-xxxxx”，Token 模式的 UserName 应该设置为 “ Token|YYYYY|mqtt-xxxxx”。

Password

权限验证模式	Password	适用场景
签名验证	对 clientId 签名的 Base64 编码结果，具体设置方法参考签名计算文档	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	上传的 Token 内容，具体设置方法参考 Token 计算文档	临时授权，适用于客户端不安全的场景

鉴权模式分类和比较

按照上文分类，微消息队列 MQTT 目前支持了签名校验以及 Token 校验两种验证方式，这两种方式分别对应了固定权限以及非固定的临时权限两种场景，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 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 AccessKeySecret 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 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子账号系统无法满足。

  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。具体的使用方法参考 Token 服务章节的文档。

2 签名鉴权模式

如果您需使用微消息队列 MQTT 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*#unique_6*。

- **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 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。

在微消息队列 MQTT 控制台左侧导航栏，单击签名校验，然后输入程序使用的账号的 AccessKeyId、AccessKeySecret 以及 Client ID，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 MQ 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 通过 Token 鉴权模式向 MQTT 客户端提供有时效性的临时访问权限。通过 MQTT Token 服务可以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，以实现对单一客户端，单一资源的细粒度权限控制。

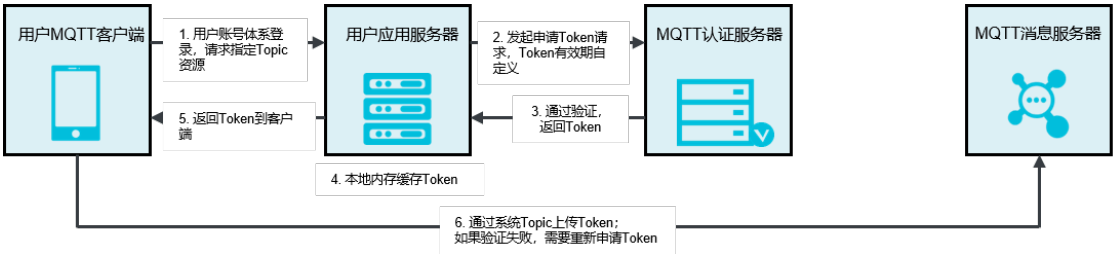
用户作为本地账号管理者，为实际客户端申请临时访问凭证（Token），客户端使用临时凭证来做实际的业务访问。Token 代表单一客户端的临时身份，由业务管理者指定该 Token 所拥有的权限、资源列表和访问类型，以实现对单一客户端，单一资源的细粒度权限控制。

术语	中文名称	说明
Token	临时凭证	微消息队列 MQTT 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
AppServer	应用服务器	用户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
AuthServer	认证服务器	微消息队列 MQTT 权限认证服务器，用来处理 AppServer 发起的 Token 相关的请求

使用流程

使用 Token 鉴权模式相比签名模式更加复杂，业务方需要按照下图所示的流程，部署自己的应用服务器，且 MQTT 客户端的初始化需要具备和业务方的应用服务器交互（获取和更新 Token）的能力。

图 3-1: 鉴权流程



具体步骤如下：

1. 客户端启动时，需要先连接自己的应用服务器验证身份。
2. 客户端向应用服务器申请自己所需的所有 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 的权限，如果验证通过则向微消息队列 MQTT 认证服务器申请对应资源的 Token。
4. 微消息队列 MQTT 认证服务器验证申请 Token 的请求，判断合法之后返回对应的 Token。
5. MQTT 客户端将应用服务器返回的 Token 持久化到本地，对每个客户端需要的权限以及 Token 信息做一个映射。缓存 Token 有两个用处：
   - 客户端重新启动进行访问时，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，如果认证服务器异常，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 消息服务器，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

- 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 可以选择断开连接重新初始化连接，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，避免因访问认证服务器短暂不可用导致的业务阻塞。

相关 API

Token 鉴权流程通过相关的 API 来完成。

Token 的申请和吊销管理交由应用服务器负责，和微消息队列 MQTT 认证服务器之间通过 HTTPS 的 OpenAPI 进行交互。每个接口都要求通过 AccessKey 和请求签名做来做身份验证。目前开放申请 Token、吊销 Token 以及校验 Token 接口。

微消息队列 MQTT 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。

Token 服务地址

目前 Token 服务支持中国公共云地域和新加坡地域，暂不支持金融云。具体的服务地址如下：

服务地域	Token 服务器地址
华北2	mqauth.aliyuncs.com
公网	mqauth.aliyuncs.com
华东1	mqauth.aliyuncs.com
华东2	mqauth.aliyuncs.com
华南1	mqauth.aliyuncs.com
新加坡	mqauth.ap-southeast-1.aliyuncs.com

3.2 Token 应用服务器接口

3.2.1 应用服务器 Token 接口规范

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，具体规范如下：

签名计算

应用服务器操作申请 Token 等接口，服务端会对调用参数进行参数校验并过滤非法请求。应用服务器需要按照如下规则将请求参数按照指定方式排序拼接，组织成待签名字符串，然后使用密钥 AccessKeySecret 加密得到签名。

说明：

此处的签名计算规则和 MQTT 客户端的签名校验是不同的逻辑，遵循不同的签名计算规则。

具体约束规则如下：

- 参数对使用 key=value 格式；
- 参数对之间使用“&”分隔；
- 计算签名时，需要将参数对按照 key 的字典序排序。

- 参数对内部，如果同一个 key 有多个 value，即同名参数，也需要使用字典序排序，按序使用“,”连接。

示例

比如原始请求的 HTTP 方法是：

```
https://mqauth.aliyuncs.com/token/apply
```

参数列表有：

```
parama=a
paramc=c2,c1
paramb=b2,b1,b3
```

那么先将参数按照 key 排序，得到：

```
parama=a
paramb=b2,b1,b3
paramc=c2,c1
```

再将同一参数内的值也按照字典序排序，此处“b2,b1,b3”是乱序的，因此排序参数得到：

```
parama=a
paramb=b1,b2,b3
paramc=c1,c2
```

最终拼接得到待签名字符串：

```
parama=a&paramb=b1,b2,b3&paramc=c1,c2
```

然后使用 secretKey 作为密钥，HmacSHA1 算法计算得到签名。

注意：

- HTTP 管理 Token 时，可以选用 HTTP 或者 HTTPS，应用自行选择。
- HTTP 方法可以选择 GET 或者 POST，建议生产环境使用 POST，测试环境可以使用 GET。

返回状态码

Token OpenAPI 请求到达认证服务器后，正常处理后会返回给调用方响应，类型是 JSON 字符串，调用方可根据需求取出 JSON 中特定的值。

JSON 字符串的 Key-Value 映射如下：

Key	类型	说明
success	boolean	是否成功。

Key	类型	说明
message	String	处理结果信息，或者异常描述。
code	int	返回码，可根据 code 判断当前请求处理情况或者错误类型。
tokenData	String	Token 字符串，申请 Token 的接口会返回该值。

其中，错误码列表如下：

Code	说明
200	成功
400	参数校验错误
407	签名计算错误
409	Token 生成处理错误
410	Token 吊销处理失败
411	调用接口频率过高被限流
1	伪造 Token，不可解析
2	Token 已经过期
3	Token 已经被吊销

3.2.2 申请 Token 接口

接口

```
https://{tokenServerUrl}/token/apply
```

使用场景

申请 Token 的接口应该由应用服务器发起，应用服务器验证 MQTT 客户端的权限范围后代替客户端向 MQTT 认证服务器申请 Token。

使用限制

单用户请求频率限制为 1000 次/秒。如有特殊需求，请提交工单申请。

请求参数

名称	类型	说明
actions	String	Token 的权限类型，有“R”（读），“W”（写），“R,W”（读和写）三种类型。如果同时需要读和写的权限，“R”和“W”之间需要用逗号隔开。
resources	String	资源名称，即 MQTT Topic，多个 Topic 以逗号（,）分隔，每个 Token 最多运行操作 100 个资源，当有多个 Topic 时，需要按照字典序排序。 申请 Token 时注册的资源参数支持 MQTT 通配符语法，包含加号单级通配符（+）和井号多级通配符（#）。 例如，如果申请 Token 时指定 `resources` 为“Topic1/+”，则客户端可以操作“Topic1/xxx”的任意 Topic；如果申请 Token 时指定 `resources` 为“Topic1/#”，则客户端可以操作“Topic1/xxx/xxx/xxx”的任意多级 Topic。
accessKey	String	当前请求使用的账号的 AccessKeyId。
expireTime	Long	Token 失效的毫秒时间戳，允许设置的失效最小间隔是 60 秒，最长为 30 天。如果输入的取值超过 30 天，申请接口不会报错，但实际生效时间为 30 天。
proxyType	String	Token 类型，填 MQTT，根据实际产品选择。

名称	类型	说明
serviceName	String	填 mq，其他参数无效 。
instanceId	String	MQTT 实例 ID，一定要和客户端实际使用的实例 ID 匹配。
signature	String	签名字符串，本请求需要计算签名的字段是 actions、resources、expireTime、serviceName、instanceId 。

3.2.3 校验 Token 接口

接口

```
https://{tokenServerUrl}/token/query
```

使用场景

校验 Token 的接口应该由应用服务器发起，应用服务器可以使用本接口确认单个 Token 是否有效。

使用限制

单用户请求频率限制为 1000 次/秒。如有特殊需求，请提交工单申请。

请求参数

名称	类型	说明
token	String	需要查询的 Token
accessKey	String	当前请求使用的账号的 AccessKeyId
signature	String	签名字符串，本请求需要计算签名的字段是 Token

3.2.4 吊销 Token 接口

接口

```
https://{tokenServerUrl}/token/revoke
```

调用场景

吊销 Token 的接口应该由应用服务器发起，在需要提前中止之前的 Token 授权时调用。

使用限制

单用户请求频率限制为 1 次/分钟。

请求参数

名称	类型	说明
token	String	需要做吊销处理的 Token
accessKey	String	当前请求使用的账号的 AccessKeyId
signature	String	签名字符串，本请求需要计算签名的字段是 Token

3.3 Token 客户端接口

本文介绍微消息队列 MQTT 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：JSONString

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：JSONString

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