ALIBABA CLOUD

Alibaba Cloud

API 网关

历史功能

文档版本：20211126

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**。
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	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	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.流量控制策略 05
2.后端签名说明文档 06
3.经典网络共享实例 09
4.API 网关 OpenID Connect 使用指南 10

1.流量控制策略

流量控制策略和 API 分别是独立管理的，操作两者绑定后，流控策略才会对已绑定的 API 起作用。

在已有的流量控制策略上，可以额外配置特殊用户和特殊应用（APP），这些特例也是针对当前策略已绑定的API生效。

流量控制策略可以配置对 API、用户、应用三个对象的流控值，流控的单位可以是秒、分钟、小时、天。使用流量控制策略您需要了解以下几点：

- 流量控制策略可以涵盖下表中的维度：

API 流量限制	该策略绑定的API在单位时间内被调用的次数不能超过设定值，单位时间可选秒、分钟、小时、天，如5000次/分钟。
APP 流量限制	每个APP对该策略绑定的任何一个API在单位时间内的调用次数不能超过设定值。如50000次/小时。
用户流量限制	每个阿里云账号对该策略绑定的任何一个 API 在单位时间内的调用次数不能超过设定值。一个阿里云账号可能有多个 APP，所以对阿里云账号的流量限制就是对该账号下所有 APP 的流量总和的限制。如 50 万次/天。

  在一个流控策略里面，这三个值可以同时设置。请注意，用户流量限制应不大于 API 流量限制，APP 流量限制应不大于用户流量限制。即 APP 流量限制 <= 用户流量限制 <= API 流量限制。

  此外，您可以在流控策略下添加特殊应用（APP）和特殊用户。对于特例，流控策略基础的 **API 流量限制** 依然有效，您需要额外设定一个阈值作为该 APP 或者该用户的流量限制值，该值不能超过策略的 **API流量限制** 值，同时流控策略基础的 **APP流量限制** 和 **用户流量限制** 对该 APP 或用户失效。

- 与签名密钥相似，当您创建流量控制策略时，需要选择 **Region**，**Region** 一旦选定不可更改，且仅能被应用于同一个 Region 下的 API。
- 由于 API 网关限制，当您设置 **API 流量限制** 值时，考虑每个 API 分组的默认流控上限是500QPS（该值可以通过提交工单申请提高）。
- 绑定 API。您可以将策略绑定于多个 API，流控策略的限制值和特例将对该策略绑定的每一个 API 单独生效。当您绑定 API 时，如果该 API 已经与某个策略绑定，您的此次操作将替换之前的策略，实时生效。
- 特殊对象。如果您想要添加特殊应用或者特殊用户，您需要获得应用 ID 即 AppID 或者用户的阿里云邮箱账号。
- 在 API 网关控制台，您可以完成对流量控制策略的创建、修改、删除、查看等基本操作。还有流控策略与 API 的绑定解绑，流量控制策略特殊对象的添加删除等操作。

2.后端签名说明文档

后端签名密钥是由您创建的一对 Key 和 Secret，相当于您给网关颁发了一个账号密码。开启后端签名后，API 网关向您后端服务请求时会使用这一对Key和Secret对请求内容进行加签处理，您后端服务可以对网关发送过来的请求做对称加签计算，对比网关的签名和服务器端计算的签名是否一致就可以对网关做身份验证。

概述

- 创建签名密钥并将签名密钥绑定到 API 即可开启后端签名（请妥善保管此密钥，API 网关会对密钥进行加密存储来保障密钥的安全性）。
- 创建密钥时需要选择 **Region**，**Region** 一旦选定不能更改，而且密钥只能被绑定到同一个 **Region** 下的API上。
- 一个 API 仅能绑定一个密钥，密钥可以被替换和修改。
- 所有您定义的参数都会参与签名，包括您录入的业务参数、您定义的常量系统参数和 API 网关系统参数（如 CaClientIp 等）。
- 后端对 API 网关的签名字符串校验后，如果校验失败，建议返回 errorcode = 403；errormessage = “InvalidSignature”。
- 若您的后端服务为VPC环境，且通过内网对接（VPC内网访问API网关），您无需使用后端签名，通道自身是安全的。

读取 API 网关签名方法

网关计算的签名保存在 Request 的 Header 中，Header 名称：X-Ca-Proxy-Signature。

后端 HTTP 服务加签方法

签名计算的详细 demo（JAVA）请参照链接：https://github.com/aliyun/api-gateway-demo-sign-backend-java。

签名计算方法步骤如下：

1. **组织参与加签的数据：**

```
String stringToSign=
HTTPMethod + "\n" +      //Method全大写
Content-MD5 + "\n" +     //Content-MD5 需要判断是否为空，如果为空则跳过，但是为空也需要添加换行符 "\n"
Headers +            //Headers 如果为空不需要添加"\n"，不为空的Headers中包含了"\n"，详见下面组织Headers的描述
Url
```

2. **计算签名：**

```
Mac hmacSha256 = Mac.getInstance("HmacSHA256");
byte[] keyBytes = secret.getBytes("UTF-8"); //secret 为绑定到 API 上的签名密钥
hmacSha256.init(new SecretKeySpec(keyBytes, 0, keyBytes.length, "HmacSHA256"));
String sign = new String(Base64.encodeBase64(Sha256.doFinal(stringToSign.getBytes("UTF-8")),"UTF-8"));
```

补充说明

- **Content-MD5**

Content-MD5 是指 Body 的 MD5 值，只有 HttpMethod 为 PUT 或者 POST 且 Body 为非 Form 表单时才计算 MD5，计算方式为：

```
String content-MD5 = Base64.encodeBase64(MD5(bodyStream.getbytes("UTF-8")));
```

- **Headers**

  Headers 指所有参与签名计算的 Header的Key、Value。参与签名计算的 Header 的 Key 从 Request Header 中读取，Key为："X-Ca-Proxy-Signature-Headers"，多个 Key 用英文逗号分割。

- **Headers 组织方法**：

  先对所有参与签名计算的 Header 的 Key 按照字典排序，然后将 Header 的 Key 转换成小写后按照如下方式拼接：

```
String headers = HeaderKey1.toLowerCase() + ":" + HeaderValue1 +"\n"+
 HeaderKey2.toLowerCase() + ":" + HeaderValue2 +"\n"+
 ... +
HeaderKeyN.toLowerCase() + ":" + HeaderValueN + "\n"
```

- **Url**

  Url指 Path+Query+Body 中 Form 参数，组织方法： 如果有 Query 或 Form 参数则加 **?**，然后对 Query+Form 参数按照字典对 Key 进行排序后按照如下方法拼接，如果没有 Query 或 Form 参数，则 Url = Path。

```
String url =
 Path +
 "?" +
 Key1 + "=" + Value1
+ "&" + Key2 + "=" + Value2 +
 ...
"&" + KeyN + "=" + ValueN
```

> **说明**
>
> 1. 这里 Query 或 Form 参数的 Value 可能有多个，多个的时候只取第一个 Value 参与签名计算；2. 只要传递了的参数，签名过程中的等号 “=” 无论什么情况都需要保留，比如这两个query参数传递时的形式："path?a=&b"，签名时需要写成："path?a=&b="。

- **调试模式**

  为了方便后端签名接入与调试，可以开启 Debug 模式进行调试，具体方法如下：

  i. 请求API网关的 Header 中添加 X-Ca-Request-Mode = debug。

  ii. 后端服务在 Header 中读取 X-Ca-Proxy-Signature-String-To-Sign 即可，因为 HTTP Header 中值不允许有换行符，因此换行符被替换成了 "|"。

  注意：X-Ca-Proxy-Signature-String-To-Sign 不参与后端签名计算。

- **时间戳校验**

  如果后端需要对请求进行时间戳校验，可以在 API 定义中选择系统参数 "CaRequestHandleTime"，值为网关收到请求的格林威治时间。

密钥泄露 修改替换

当您遇到如下情况：

- 您的某一个密钥发生了泄露，您可能想要保留该密钥与 API 的绑定关系，但是想要修改密钥的 Key 和 Secret。
- 当您操作将密钥应用于 API 时，可能该 API 已经绑定了某个密钥，需要替换密钥。

以上两种情况都建议按照下面的流程来操作：

1. 先在后端同时支持两个密钥：原来的密钥和即将修改或替换的密钥，确保切换过程中的请求能够通过签名验证，不受修改或替换的影响。
2. 后端配置完备后，完成修改，确定新 Key 和 Secret 生效后再将之前已泄露或废弃的密钥删除。

3.经典网络共享实例

经典网络共享实例不需要支付资源的时租费用，按照API的使用次数和产生的公网流量计费。但服务器资源池、IP地址、带宽等资源为当前Region下的一组用户共有，**共享实例API分组的RPS默认上限为500，如果您需要更大的RPS，请直接购买专享实例。**共享实例（经典网络）是API网关最早推出的实例类型，支持公网访问、VPC内网访问、支持VPC后端地址、公网后端地址及经典网络内网后端地址，但暂时并不支持新推出的插件体系。

> **说明**
>
> 经典网络共享实例是API网关早期实例类型，功能有限制，如果没有对接经典网络后端地址的需求，不建议使用。

技术规格	共享实例（经典网络）
插件系统	不支持
后端地址	支持经典网络内网后端地址
函数计算	同Region内网连接
https后端tls版本	TLS1.0
CORS	默认返回全部允许
Flash跨域	内置crossdomain.xml

4.API 网关 OpenID Connect 使用指南

阿里云API网关在OpenID Connect协议的基础上实现了一套基于用户体系对用户的API进行授权访问的机制，满足用户个性化安全设置的需求。

1. OpenID Connect简介

OpenID Connect是2014年初发布的开放标准，定义了一种基于OAuth2的可互操作的方式来来提供用户身份认证。它使用简单的REST/JSON消息流来实现，和之前任何一种身份认证协议相比，开发者可以轻松集成。OpenID Connect使用 API 进行身份交互的框架，允许客户端根据授权服务器的认证结果最终确认用户的身份，以及获取基本的用户信息；支持包括Web、移动、JavaScript在内的所有客户端类型；它是可扩展的协议，允许你使用某些可选功能，如身份数据加密、OpenID提供商发现、会话管理等。

1.1 OpenID Connect 角色介绍

1. 客户端：直接为终端用户提供服务
2. 认证服务：OpenID 提供方，通常是一个 OpenID 认证服务器，它能为第三方颁发用于认证的ID Token
3. 业务服务：提供业务服务，比如查询用户资料
4. 终端用户：指持有资源拥有人

1.2 OpenID Connect 流程描述

1. 客户端发送认证请求给认证服务；
2. 终端用户在认证页面进行授权确认（可选）；
3. 认证服务对认证请求进行验证，发送ID Token给客户端；
4. 客户端向业务请求，请求中携带ID Token；
5. 业务服务验证ID Token是否合法后返回业务应答；

这个流程的第二步是可选的，如果终端用户在客户端输入了用户名和密码，第一步中的认证请求中携带了用户名和密码，那么认证服务在验证了用户名和密码后，省去第二步，可以直接在应答中返ID Token。这种模式更加简洁，阿里云的API网关的OpenID Connect验证模式就是这种，本文将在第二节仔细描述。

1.3 JWT(JSON Web Token)格式数据

认证服务返回的ID Token需要严格遵守JWT(JSON Web Token)的定义，下面是JWT(JSON Web Token)的定义细节：

1. iss：必须。Issuer Identifier，认证服务的唯一标识，一个区分大小写的https URL，不包含query和fragment组件。
2. sub：必须。Subject Identifier，iss提供的终端用户的标识，在iss范围内唯一，最长为255个ASCII个字符，区分大小写。
3. aud：必须。Audience(s)，标识ID Token的受众，必须包含OAuth2的client_id，分大小写的字符串数组。
4. exp：必须。Expiration time，超过此时间的ID Token会作废。
5. iat：必须。Issued At Time，JWT的构建的时间。
6. auth_time：AuthenticationTime，终端用户完成认证的时间。
7. nonce：发送认证请求的时候提供的随机字符串，用来减缓重放攻击，也可以用来关联客户端Session。如果nonce存在，客户端必须验证nonce。
8. acr：可选。Authentication Context Class Reference，表示一个认证上下文引用值，可以用来标识认证上下文类。
9. amr：可选。Authentication Methods References，表示一组认证方法。
10. azp：可选。Authorized party，结合aud使用。只有在被认证的一方和受众（aud）不一致时才使用此值，一般情况下很少使用。

下面是一个典型ID Token的示例：

```
{
    "iss": "https://1.2.3.4:8443/auth/realms/kubernetes",
    "sub": "547cea22-fc8a-4315-bdf2-6c92592a6e7c",
    "aud": "kubernetes",
    "exp": 1525158711,
    "iat": 1525158411,
    "auth_time": 0,
    "nonce": "n-0S6_WzA2Mj",
    "acr": "1",
    "azp": "kubernetes",
    "nbf": 0,
    "typ": "ID",
    "session_state": "150df80e-92a1-4b0c-a5c5-8c858eb5a848",
    "userId": "123456",
    "preferred_username": "theone",
    "given_name": "the",
    "family_name": "one",
    "email": "theone@mycorp.com"
}
```


关于ID Token的更详细的定义请参见OpenID Connect Core 1.0。

2. API网关OpenID Connect业务流程

阿里云API网关将OpenID Connect作为用户自主授权API的一种认证模式。作为处于客户端和后端服务中间的API网关，可以为后端服务去验证ID Token的合法性，并将没有拿到后端服务颁发的合法ID Token的请求在API网关层面给拒绝了。

2.1 准备条件

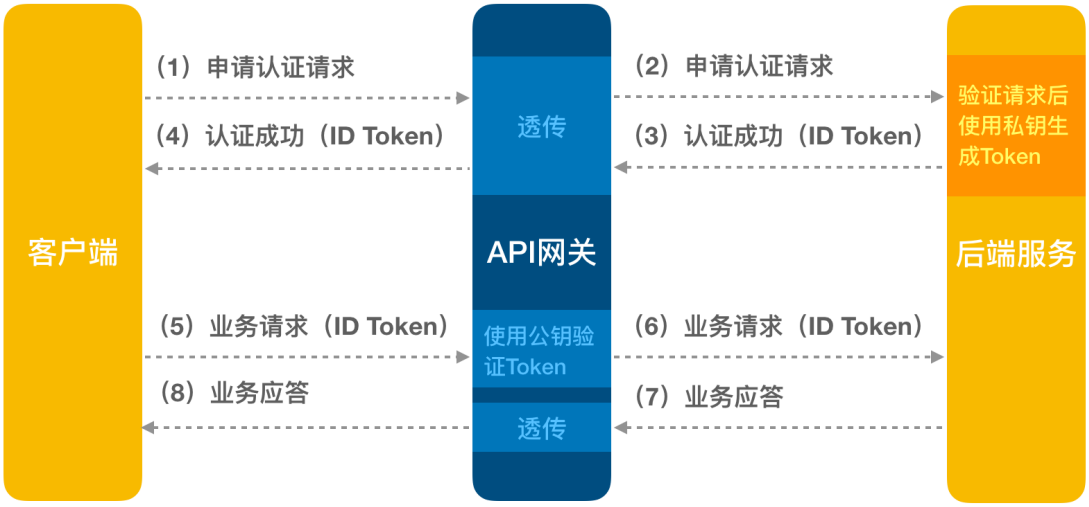
用户需要在API网关上配置以下两点，才能正常使用API网关的OpenID Connect认证模式：

1. 配置一个获取授权API，并且在这个API上指定一个公钥；

2. 所有需要保护的业务API设置成为OpenID Connect的业务API;

具体的配置方法参见第三章。

2.2 业务流程



上图是API网关处理OpenID Connect的整个业务流程，下面我们用文字来详细描述下：

1. 客户端向API网关发起认证请求，请求中一般会携带终端用户的用户名和密码；

2. API网关将请求直接转发给后端服务；

3. 后端服务读取请求中的验证信息（比如用户名、密码）进行验证，验证通过后使用私钥生成标准的ID Token，返回给API网关；

4. API网关将携带ID Token的应答返回给客户端，客户端需要将这个ID Token缓存到本地；

5. 客户端向API网关发送业务请求，请求中携带ID Token；

6. API网关使用用户设定的公钥对请求中的ID Token进行验证，验证通过后，将请求透传给后端服务；

7. 后端服务进行业务处理后应答；

8. API网关将业务应答返回给客户端。

在这个整个过程中，API网关利用OpenID Connect的认证机制，实现了用户使用自己的用户体系对自己API进行授权的能力。

2.3 授权范围与时效

API网关会认为用户颁发的ID Token有权利访问整个分组下的所有OpenID Connect的业务API。如果需要更细力度的权限管理，还需要后端服务自己解开ID Token进行权限认证。 API网关会验证ID Token中的exp字段，一旦这个字段过期了，API网关会认为这个ID Token无效而将请求直接打回。过期时间这个值必须设置，并且过期时间一定要小于7天。

3. 在API配置OpenID Connect

本章主要教大家如何在API网关配置OpenID Connect，主要有三步：

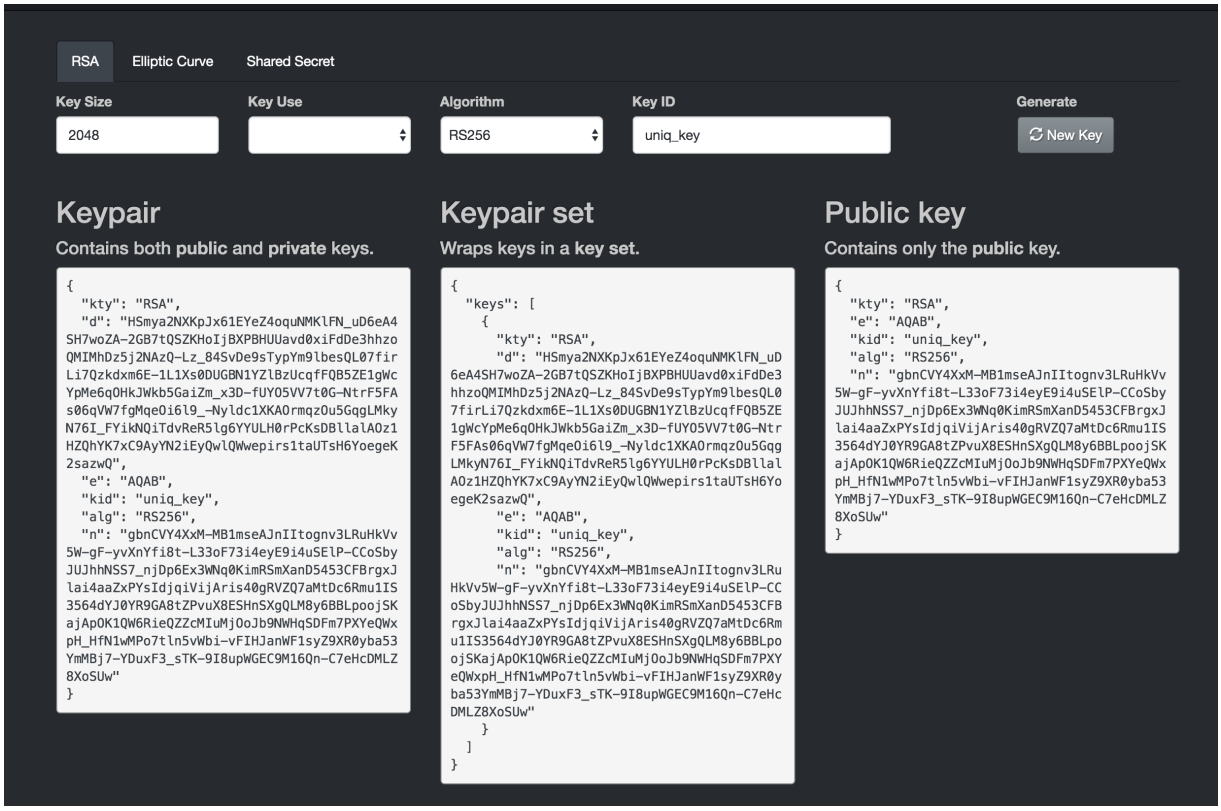
1. 生成一个公私秘钥对；

2. 配置一个授权API，在这个API上将公钥配置好；

3. 所有业务API安全认证类型设置为OpenID Connect模式。下面是具体细节。

3.1 准备公钥与私钥

用户可以在这个站点https://mkjwk.org生成用于ID Token生成与验证的公钥与私钥，目前API网关支持的秘钥对的加密算法为RSA SHA256，秘钥对的加密的位数为2048。



在这个网站生成的公钥用于本文3.1中设置授权API:

```
{
 "kty": "RSA",
 "e": "AQAB",
 "kid": "uniq_key",
 "alg": "RS256",
 "n": "gbnCVY4XxM-MB1mseAJnIItognv3LRuHkVv5W-gF-yvXnYfi8t-L33oF73i4eyE9i4uSElP-CCoSbyJUJhhNSS7_njDp6Ex3WNq0KimRSmXanD5453CFBrgxJlai4aaZxPYsIdjqiVijAris40gRVZQ7aMtDc6Rmu1IS3564dYJ0YR9GA8tZPvuX8ESHnSXgQLM8y6BBLpoojSKajApOK1QW6RieQZZcMIuMjOoJb9NWHqSDFm7PXYeQWxpH_HfN1wMPo7tln5vWbi-vFIHJanWF1syZ9XR0yba53YmMBj7-YDuxF3_sTK-9I8upWGEC9M16Qn-C7eHcDMLZ8XoSUw"
}
```

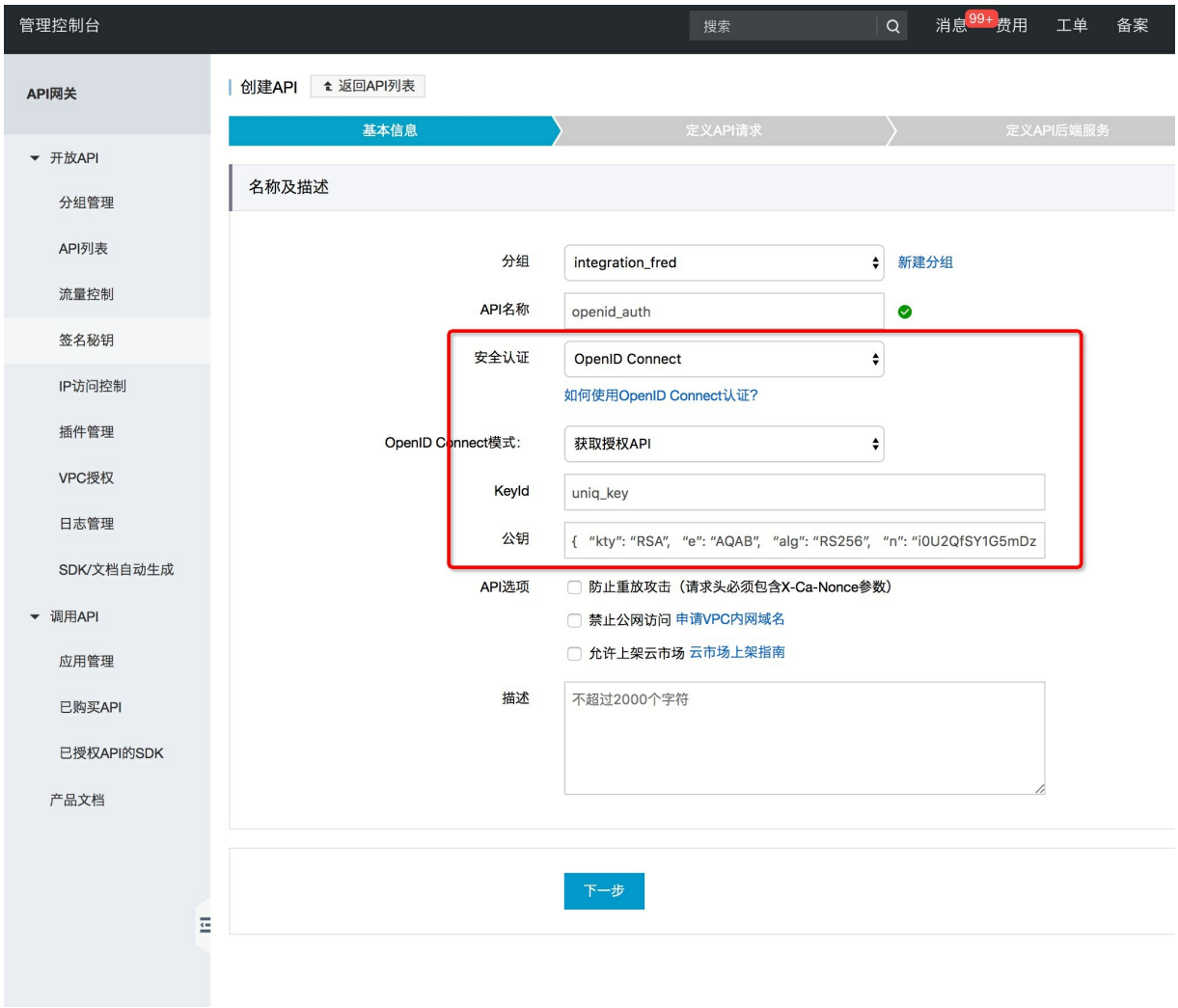
在这个网站生成的秘钥对用于本文4章中生成IdToken编程时使用:

{
"kty": "RSA",
"d": "HSmya2NXKpJx61EYeZ4oquNMKlFN_uD6eA4SH7woZA-2GB7tQSZKHoIjBXPBHUUavd0xiFdDe3hhzoQMIMhDz5j2NAzQ-Lz_84SvDe9sTypYm9lbesQL07firLi7QzkdxM6E-1L1Xs0DUGBN1YZlBzUcqfFQB5ZE1gWcYpMe6qOHkJWkb5GaiZm_x3D-fUYO5VV7t0G-NtrF5FAs06qVW7fgMqeOi6l9_-Nyldc1XKAOrmqzOu5GqgLMkyN76I_FYikNQiTdvReR5lg6YYULH0rPcKsDBllalAOz1HZQhYK7xC9AyYN2iEyQwlQWwepirs1taUTsH6YoegeK2sazwQ",
"e": "AQAB",
"kid": "uniq_key",
"alg": "RS256",
"n": "gbnCVY4XxM-MB1mseAJnIltognv3LRuHkVv5W-gF-yvXnYfi8t-L33oF73i4eyE9i4uSElP-CCoSbyJUJhhNSS7_njDp6Ex3WNq0KimRSmXanD5453CFBrgxJlai4aaZxPYsIdjqiVijAris40gRVZQ7aMtDc6Rmu1IS3564dYJ0YR9GA8tZPvuX8ESHnSXgQLM8y6BBLpoojSKajApOK1QW6RieQZZcMIuMjOoJb9NWHqSDFm7PXYeQWxpH_HfN1wMPo7tln5vWbi-vFIHJanWF1syZ9XR0yba53YmMBj7-YDuxF3_sTK-9I8upWGEC9M16Qn-C7eHcDMLZ8XoSUw"
}

3.2 配置授权API和业务API

3.2.1 授权API

下图是配置授权API的第一页，也是最重要的一页，除了这一页以外，授权API的其他配置均和普通的API配置相同。

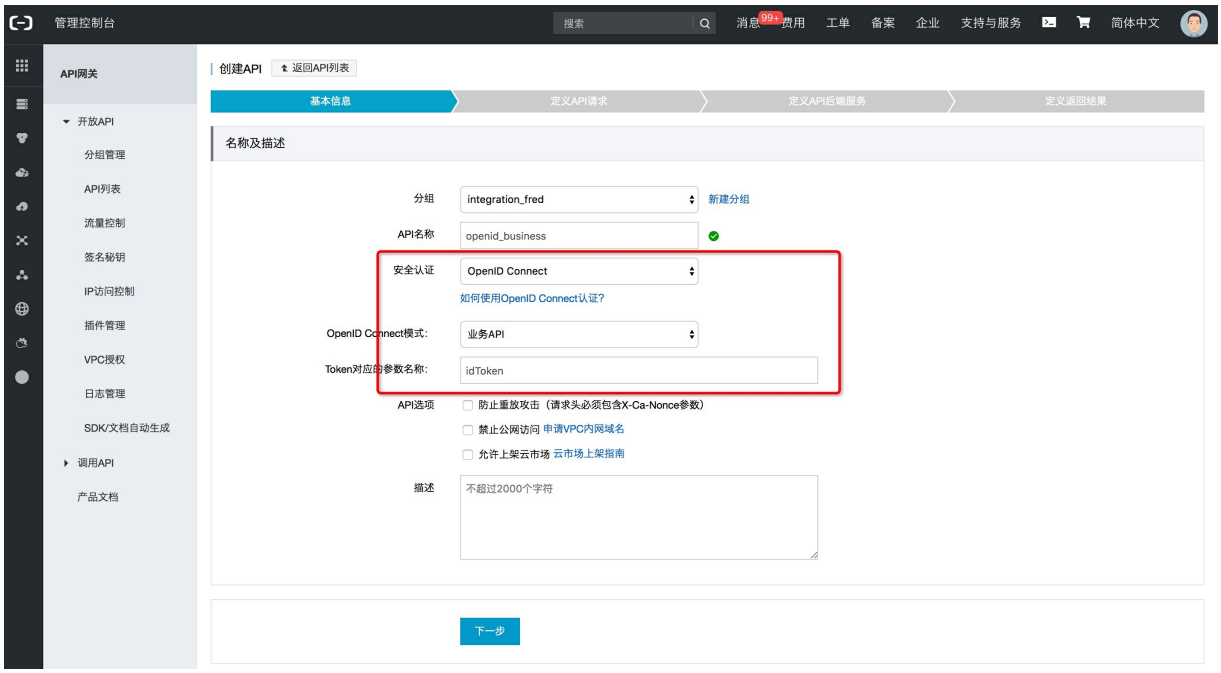


这一页中标注出来的注意项如下：

1. 安全认证：选择OpenID Connect或者 OpenID Connect & 阿里云APP，这两者的区别是，后者除了验证OpenID Connect的ID Token之外，还需要验证阿里云自己的APP，签名。
2. OpenID Connect 模式：选择获取授权API;
3. KeyId，填写一个分组范围内唯一的秘钥名称，尽量使用英文加数字；
4. 公钥，填写刚才生成的公钥，是整个JSON串都需要填入。

3.2.2 业务API

下图是配置OpenID Connect 业务API的第一页，也是最重要的一页，除了这一页以外，OpenID Connect 业务API的其他配置均和普通的API配置相同。



1. 安全认证：选择OpenID Connect或者 OpenID Connect && 阿里云APP，这两者的区别是，后者除了验证OpenID Connect的ID Token之外，还需要验证阿里云自己的APP，签名。
2. OpenID Connect 模式：选择业务API;
3. Token对应的参数名称，填写请求中对应的ID Token的参数名称，比如idToken;

3.3 通过API插件配置

API网关还可以通过插件方式配置OpenID Connect

4. 后端服务实现ID Token颁发

```
import java.security.PrivateKey;
import org.jose4j.json.JsonUtil;
import org.jose4j.jwk.RsaJsonWebKey;
import org.jose4j.jwk.RsaJwkGenerator;
import org.jose4j.jws.AlgorithmIdentifiers;
import org.jose4j.jws.JsonWebSignature;
import org.jose4j.jwt.JwtClaims;
import org.jose4j.jwt.NumericDate;
import org.jose4j.lang.JoseException;
```

```
public class GenerateJwtDemo {
  public static void main(String[] args) throws JoseException {
        //使用在API网关设置的keyId
    String keyId = "uniq_key";
        //使用本文3.2节生成的Keypare
    String privateKeyJson = "{\n"
      + "  \"kty\": \"RSA\",\n"
      + "  \"d\": "
      +
      "\"O9MJSOgcjjiVMNJ4jmBAh0mRHF_TlaVva70Imghtlgwxl8BLfcf1S8ueN1PD7xV6Cnq8YenSKsfiNOhC6yZ_fjW1syn5raWfj68eR7cjHWjLOvKjwVY33GBPNOvspNhVAFzeqfWneRTBbga53Agb6jjN0SUcZdJgnelzz5JNdOGaLzhacjH6YPJKpbuzCQYPkWtoZHDqWTzCSb4mJ3n0NRTsWy7Pm8LwG_Fd3pACl7JIY38IanPQDLoighFfo-Lriv5z3IdlhwbPnx0tk9sBwQBTRdZ8JkqqYkxUiB06phwr7mAnKEpQJ6HvhZBQ1cCnYZ_nIlrX9-I7qomrlE1UoQ\",\n"
      + "  \"e\": \"AQAB\",\n"
      + "  \"kid\": \"myJwtKey\",\n"
      + "  \"alg\": \"RS256\",\n"
      + "  \"n\": \"vCuB8MgwPZfziMSytEbBoOEwxsG7XI3MaVMoocziP4SjzU4IuWuE_DodbOHQwb_thUru57_Efe"
      +
      "--sfATHEa0Odv5ny3QbByqsvjyeHk6ZE4mSAV9BsHYa6GWAgEZtnDceeeDc0y76utXK2XHhC1Pysi2KG8KAzqDa099Yh7s31AyoueoMnrYTmWfEyDsQL_OAIiwgXakkS5U8QyXmWicCwXntDzkIMh8MjfPskesyli0XQD1AmCXVV3h2Opm1Amx0ggSOOiINUR5YRD6mKo49_cN-nrJWjtwSouqDdxHYP-4c7epuTcdS6kQHiQERBd1ejdpAxV4c0t0FHF7MOy9kw\"\n"
      + "}";
    JwtClaims claims = new JwtClaims();
    claims.setGeneratedJwtId();
    claims.setIssuedAtToNow();
    //过期时间一定要设置，并且小于7天
    NumericDate date = NumericDate.now();
    date.addSeconds(120*60);
    claims.setExpirationTime(date);
    claims.setNotBeforeMinutesInThePast(1);
    claims.setSubject("YOUR_SUBJECT");
    claims.setAudience("YOUR_AUDIENCE");
    //添加自定义参数，所有值请都使用String类型
    claims.setClaim("userId", "1213234");
    claims.setClaim("email", "userEmail@youapp.com");
    JsonWebSignature jws = new JsonWebSignature();
    jws.setAlgorithmHeaderValue(AlgorithmIdentifiers.RSA_USING_SHA256);
        //必须设置
    jws.setKeyIdHeaderValue(keyId);
    jws.setPayload(claims.toJson());
    PrivateKey privateKey = new RsaJsonWebKey(JsonUtil.parseJson(privateKeyJson)).getPrivateKey();
    jws.setKey(privateKey);
    String jwtResult = jws.getCompactSerialization();
    System.out.println("Generate Json Web token , result is " + jwtResult);
  }
}
```

以上是生成ID Token的Java示例代码，其中有以下几个地方需要重点关注：

1. keyId需要三个环节都一致，且全局唯一：

- 在本文3.1节，https://mkjwk.org 生成秘钥时填写的keyId；
- 在本文3.2.1节设置的keyId；
- 代码中的keyId。

2. JsonWebSignature 对象的KeyIdHeaderValue必须设置；
3. privateKeyJson 使用3.1节中生成的Keypair Json字符串（三个方框内的第一个）；
4. 过期时间一定要设置，并且小于7天；
5. 添加自定义参数，所有值请都使用String类型；

5. 常见问题

5.1 APP签名认证

如果安全认证选择“OpenID Connect & 阿里云APP”时，请求中必须携带使用阿里云AK加密的签名。

5.2 Swagger导入支持OPENID CONNECT吗？

目前暂时不支持。

5.3 API网关错误应答列表

StatusCode	ErrorMessage	含义
401	“OpenId Connect Verify Fail, IdToken Not Exist”	业务APP请求中没有携带定义好的IDToken参数。
401	“234, JWS set idToken exception”	解析IDToken时出现Exception。
401	“234, Not found by keyId”	根据Token中的KeyId，找不到用户设置的公钥。
401	“235, JWS set Public-Key exception”	根据Token中的KeyId，找到的公钥不可用。
401	” 237, Verify signature failed “	验证签名失败。
401	“245, IdToken is out of scope”	请求中的IDToken不属于当前分组，无权访问。
401	“238, JWS get payload exception”	请求中的IDToken找不到payload。

StatusCode	ErrorMessage	含义
401	“239, Parse payload to JwtClaims exception”	转化payload为 JwtClaims 时出错。
401	“239, idToken expired”	请求中的IDToken已经过期。
401	“Invalid OpenId Connect Config”	API配置错误，获取不到OpenId Connect相关配置。