API 网关

用户指南（开放 API）

用户指南（开放 API）

概述

API 网关（API Gateway），提供高性能、高可用的API托管服务，帮助您对外开放您部署在ECS、容器服务等阿里云产品上的应用，为您提供完整的API发布、管理、维护生命周期管理。您只需简单操作，即可快速、低成本、低风险的开放数据或服务。

在 API 网关您可以：

- **管理您的 API**
  您可以对API的整个生命周期进行管理，包括API的创建、测试、发布、下线、版本切换等操作。

  便捷转换数据
  支持自定义映射规则，您可以配置映射将调用请求转换成后端需要的格式。

  预设请求校验
  您可以预先设置参数类型、参数值（范围、枚举、正则、Json Schema）校验，由网关帮助您过滤掉非法请求，减少您的后端对非法请求的处理成本。

- **灵活控制流量**
  您可以对API、用户、应用设置按分钟、小时、天的调用量控制。您还可以设置特例用户或者应用，对某个用户或应用单独配置流量控制。

  轻松安全防护

  支持 Appkey 认证，HMAC（SHA-1,SHA-256）算法签名。

  支持 SSL/TSL 加密，并借助阿里云盾防病毒、防攻击。

  全面监控与报警
  为您提供可视化 API 实时监控，包括：调用量、调用方式、响应时间、错误率，并支持历史情况查询，以便统筹分析。您还可以配置预警方式（短信、Email）,订阅预警信息，以便实时掌握 API 运行情况。

- **降低开放成本**
  为您自动生成 API 文档和 SDK（服务端、移动端），降低 API 开放成本。

创建 API

创建 API 即录入 API 的定义，需要录入 API 的基本信息、服务信息、请求信息、和返回信息，然后对创建的 API 进行调试，及进行安全配置。经测试证明 API 可用后，可发布上线供用户使用。

定义 API

在 API 网关控制台中 **API 列表** 页面，单击 **创建 API**，即进入 API 的创建和定义流程。

第一部分：定义请求的基本信息

API 基本信息包括 API 分组、 API 名称、安全认证方式、 API 类型、和描述。

- **API 分组**：分组是 API 的管理单元。创建 API 之前，您需要先创建分组（ API 分组的详细说明见 API 开放）。
- **API 名称**：API 名称标识需在所属分组内具有唯一性。

  安全认证方式：是 API 请求的认证方式，目前支持 **阿里云APP**、**OpenID Connect** 、**OpenID Connect & 阿里云APP** 和 **无认证** 四种认证方式。

  - **阿里云APP**：要求请求者调用该 API 时，需通过对 APP 的身份认证。
  - **OpenID Connect**：是一套基于 OAuth 2.0 协议的轻量级规范，提供通过RESTful APIs 进行身份交互的框架。可以使用 OpenID Connect 和您的自有账号系统无缝对接。详细介绍请参见 OpenID Connect。
  - **OpenID Connect & 阿里云APP**：同时进行OpenID Connect和阿里云APP认证。
  - **无认证**：即任何人知晓该 API 的请求定义后，均可发起请求，网关不对其做身份验证，均会将请求转发至您的后端服务。（强烈不建议使用此模式。）

  API 类型：分为 **公开** 和 **私有** 两种。

  - **公开** 类型的 API：所有用户均可以在 API 网关控制台，**已发布 API** 页面看到 API 的部分信息。
  - **私有** 类型的 API：如果有用户想要调用您的私有类型的 API ，需要您主动操作授权，否则用户无渠道获取 API 信息。
- **描述**： API 功能描述。

第二部分：定义 API 请求

这部分是定义用户如何请求您的 API ，包括协议、请求Path、HTTP Method、入参请求模式、和入参定义。

- **协议**：支持 HTTP 和 HTTPS 协议。

- **请求Path**：Path 指相对于服务 Host， API 的请求路径。请求 Path 可以与后端服务实际 Path 不同，您可以随意撰写合法的有明确语义的 Path 给用户使用。您可以在请求 Path 中配置动态参数，即要求用户在 Path 中传入参数，同时您的后端又可以不在 Path 中接收参数，而是映射为在 Query、Header 等位置接收。在 开放 API 接入 API 网关 中，有详细的举例说明和操作截图展示。
- **HTTP Method**：支持标准的 HTTP Method，可选择 PUT、GET、POST、DELETE、PATCH、或 HEAD。
- **入参请求模式**：API 网关对入参的处理模式，支持 **入参映射** 和 **入参透传** 两种模式。
    - **入参映射**：即 API 网关在接收到您的 API 请求时，通过映射关系将请求转换成您后端需要的格式。使用入参映射模式的特点：
        - 定义方式：定义此类 API 时，需添加前后端参数映射关系。
        - 使用场景：
            - 同一接口，在 API 网关定义不同的 API，以服务差异化的用户。
            - 通过 API 网关规范化陈旧的系统接口。
        - 实现功能：
            - 支持您配置前端和后端的全映射，即参数混排。可以让 API 用户在 **Query** 中传入参数, 后端从 **Header** 里接收等。支持：参数名称转换和参数位置转换。
            - 可以定义参数的校验规则，实现请求参数的预校验，降低后端处理非法请求的压力。支持：参数长度校验、参数数值大小校验、参数正则校验、和参数 JSON Schema 校验。
    - **入参透传**：即 API 网关在接收到 API 请求后，不对请求进行处理，直接转发到后端服务。此模式下：
        - 无法实现参数校验。
        - 无法生成详细的 API 调用文档。
        - 自动生成的 SDK 不包含请求入参。
- **入参定义**：定义您 API 的请求入参，包含参数名、参数位置、类型、是否必填、默认值、示例、描述等信息。**入参透传模式下，不需要录入参数。**
    - **参数名**：展示给用户的参数名称。
    - **参数位置**：参数在请求中的位置，包含 Head、Query、和 Parameter Path。

      注意：如果您在 Path 中配置了动态参数，存在参数位置为 Parameter Path 的同名参数。

    - **类型**：字段的类型，支持：String、Int、Long、Float、Double、和 Boolean。
    - **必填**：指此参数是否为必填。当选择为 **是** 时，API 网关会校验用户的请求中是否包含了此参数。若不包含此参数，则拒绝该请求。
    - **默认值**：当 **必填** 为 **否** 时生效。在用户请求中不包含此参数时，API 网关自动添加默认值发送给后端服务。
    - **示例**：指参数的填写示例。
    - **描述**：参数的用途描述及使用注意事项等。
    - **参数校验规则**：单击 **编辑更多** 配置校验规则，如参数值的长度、取值大小、枚举、正则、JSON Schema等。API 网关会参照校验规则对请求做初步校验。如果入参不合法，则不会发送给您的后端服务，以降低后端服务的压力。

第三部分：定义后端服务信息

这部分主要是定义一些参数的前后端映射，即 API 后端服务的配置，包括后端服务地址、后端Path、后端超时时间、参数映射、常量参数、系统参数。用户请求到达 API 网关后，API 网关会根据您的后端配置，映射为对应的后端服务的请求形式，请求后端服务。

- **后端服务类型**：目前支持 HTTP/HTTPS和函数计算两种类型。
  - HTTP/HTTPS：若您的服务为 HTTP/HTTPS 服务，则选择此项。

    注意：若是 HTTPS 服务，后端服务必须有 SSL 证书。

  - 函数计算：若选择函数计算为后端服务，需先在 **函数计算控制台** 中创建函数，并需填入函数服务名称和函数名称，和获取函数计算角色 Arn。
- **VPC 通道**：当您的后端服务在 VPC 中时，需要选择 **使用 VPC 通道**。使用方法，请参见 **专属网络 VPC 环境开放 API**。
- **后端服务地址**：后端服务的 Host，可以是一个域名，也可以是 http(s)://host:port 的形式。填写时，必须包含 http:// 或 https:// 。
- **后端请求 Path**：Path 是您的 API 服务在您后端服务器上的实际请求路径。若您后端 Path 需要接收动态参数，则需要声明调用者需传入参数的具体位置和参数名，即声明映射关系。
- **后端超时时间**：指 API 请求到达 API 网关后，API 网关调用 API 后端服务的响应时间，即由 API 网关请求后端服务开始直至 API 网关收到后端返回结果的时长。单位为毫秒。设置值不能超过 30 秒。如果响应时间超过该值，API 网关会放弃请求后端服务，并给用户返回相应的错误信息。
- **常量参数**：您可以配置常量参数。您配置常量参数对您的用户不可见，但是 API 网关会在中转请求时，将这些参数加入到请求中的指定位置，再传递至后端服务，实现您的后端的一些业务需求。比如，您需要 API 网关每次向后端服务发送请求都带有一个关键词 **aligateway**，您就可以把 **aligateway** 配置为常量参数，并指定接收的位置。

系统参数：指 API 网关的系统参数。默认系统参数不会传递给您，但是如果您需要获取系统参数，您可以在 API 里配置接收位置和名称。具体内容如下表：

参数名称	参数含义
CaClientIp	发送请求的客户端 IP
CaDomain	发送请求的域名
CaRequestHandleTime	请求时间（格林威治时间）
CaAppId	请求的 APP 的 ID
CaRequestId	RequestId
CaApiName	API 名称
CaHttpSchema	用户调用 API 使用的协议，http 或者 https
CaProxy	代理（AliCloudApiGateway）

注意：您需确保您录入的所有参数的参数名称全局唯一，包括 Path 中的动态参数、Headers 参

数、Query 参数、Body 参数（非二进制）、常量参数、系统参数。如果您同时在 Headers 和 Query 里各有一个名为 **name** 的参数，将会导致错误。

第四部分 定义返回结果

录入返回 ContentType、返回结果示例、失败返回结果示例、和错误码定义。

API调试

API 定义录入完成后，您可以在 API 调试页面调试 API，以确定 API 的可用性。

API 创建、定义完成后，页面自动跳转到 **API 列表** 页。您可以通过此页面按钮，测试创建的 API 是否可用，请求链路是否正确。

1. 单击 API 名称或 **管理** 按钮，进入 **API 定义** 页面。
2. 单击左侧导航栏中 **调试 API**。
3. 输入请求参数，单击 **发送请求**。返回结果将显示在右侧页面。如果调试返回成功结果，则说明该 API 可以使用。如果返回代码为 4XX 或 5XX，则表示存在错误。请参见 如何获取错误信息 和 错误代码表。

后续步骤

完成以上定义后和初步调试后，您就完成了 API 的创建。您可以发布 API 到测试、预发、线上环境，继续调试或供用户使用。还可以为 API 绑定 签名密钥 和 流量控制 等安全配置。

API 开放

API 创建完成后，您就可以开放 API 服务了。要开放 API 服务您需要绑定一个在阿里云系统备案成功的独立域名，且该域名要完成 CNAME 解析。而独立域名是绑定在 API 分组上面的，所以在这个部分为您详细说明一下开放 API 服务需要了解的 API 分组和域名。

第一部分：API 分组

API 分组是 API 的管理单元。您创建 API 之前，需要先创建分组，然后在某个分组下创建 API。分组包含名称、描述、区域（Region）、域名几大属性。

- 分组的区域（Region）在分组创建时选定不可更改。创建 API 时，如果选定分组那么 Region 也一同选定，不可更改。

- 每个账号 API 分组个数上限为50个，每个分组 API 个数上限为200个。
- 域名。分组创建时，系统会为分组分配一个二级域名。如果需要开放 API 服务，您需要为分组绑定一个在阿里云系统备案成功的独立域名，且将独立域名 CNAME 到相应的二级域名上。每个分组最多只能绑定5个独立域名。具体请看下文——域名及证书。

第二部分：环境管理

关于环境需要理解两个概念，**环境**和**环境变量**。

> **环境**是 API 分组上的一个配置，每个分组有若干个环境。 API 录入后，未经发布时，就只是 API 定义。发布到某个环境后才是能够对外提供服务的 API 。
>
> **环境变量**是在环境上用户可创建可管理的一种变量，该配置是固定于环境上的。如在线上环境创建变量，变量名为 **Path**，变量值为 **/stage/release**.

在 API 定义中的 **Path** 位置，写作 **#Path#**，即配置为变量标识，变量名为 **Path**。

那么将该 API 发布到线上环境时，该 API 在线上环境的运行定义，Path 处的 **#Path#**，会取值为 **/stage/release**。

而将该 API 发布到其他环境时，若环境上没有环境变量 **#Path#**，则无法取值，那么 API 就无法调用。

使用环境变量可以解决后端服务需要区分环境的问题，通过不同的环境上配置不同的服务地址和Path，来调用不同的后端服务，同时 API 的定义配置又是一套。使用时需要注意以下几点：

- 在 API 定义中配置了变量标识后，在 API **列表—管理—调试** 页面将无法调试。
- 变量名严格区分大小写。
- 如果在 API 定义中设置了变量，那么一定要在要发布的环境上配置 **变量名**和**变量值**，否则变量无赋值， API 将无法正常调用。

第三部分：域名及证书

API 网关通过域名来定位到一个唯一的 API 分组，再通过 **Path+HTTPMethod** 确定唯一的 API 。如果要开放 API 服务，您需要了解 **二级域名** 和 **独立域名**。

- **二级域名**是分组创建时系统分配的，唯一且不可更改。在您还没有独立域名之前，您可以通过访问二级域名来测试调用您的 API 。二级域名仅能用于测试，默认每天只能请求1000次。

  独立域名即自定义域名，是您开放 API 服务需要绑定的，用户通过访问您的独立域名来调用您开放的 API 服务。您可以为一个分组绑定多个独立域名，上限为5个。对于独立域名的配置您需要注意以下几点：

  > 独立域名不必须是根域名，可以是二级、三级域名。
  >
  > 独立域名如果尚未备案，则可以在阿里云做 首次备案。

独立域名若已在其他系统备案，则需要在阿里云做 **备案接入**。

独立域名需要 CNAME 解析到分组的二级域名上。

满足上述的备案和解析两个要求，域名才能成功绑定。

当您的 API 服务支持 HTTPS 协议时，需要为该域名上传 SSL 证书，在 **分组详情** 页面进行添加即可。SSL 证书上传不支持文件上传，需要填写 **证书名称**、**内容** 和 **私钥**。

第四部分：测试、线上、授权

通过上述操作您已经完成 API 的创建和域名绑定，接下来就可以将 API 发布到测试或者线上环境，进行调试和开放了。其中一个重要的环节是授权，授权即授予某个 APP 可以调用某个 API 的权限。

- 当您完成 API 创建之后，您就可以将 API 发布到测试或者线上，并给自己创建的 APP 授权，通过访问二级域名来调用指定环境中的 API，进行测试。
- 成功绑定独立域名之后，您的 API 就可以在市场上架，供客户购买、调用。您还可以不经过购买将 API 授权给合作伙伴的 APP，供其调用。

至此，您完成 API 服务的开放。在 API 创建到开放的整个过程中，您还可以随时操作 API 的创建、修改、删除、查看、测试、发布、下线、授权、解除授权、发布历史及版本切换等操作。

API 管理

API 定义就是指您创建 API 时对 API 的请求结构的各方面定义。您可以在控制台完成 API 定义的查看、编辑、删除、创建、复制。您需要注意以下几点：

- 当您需要编辑某个 API 的定义时，如果该 API 已经发布，对定义的修改不会对线上产生影响，定义修改后需要再次发布才能把修改后的定义同步到线上环境。
- 当您想要删除某个 API，如果该 API 已经发布，则不允许直接删除 API 定义，需要先将 API 下线，然后删除。
- API 网关为您提供了复制定义的功能。您可以从测试环境/线上环境复制线上的定义覆盖当前的最新定义，然后重新点击编辑进行修改。

API 发布管理

当您完成 API 的创建后，您可以将 API 发布到测试或者线上。也可以将测试或者线上的 API 下线。您需要注意以下几点：

- API 创建完成后，发布到某环境，通过二级域名或者独立域名访问时，需要在请求的Header指定要请

求的环境，参见 **请求示例**。
- 当您要发布某个 API 时，如果该 API 在测试或者线上已经有版本在运行，您的此次发布将使测试或者线上的该 API 被覆盖，实时生效。但是历史版本及定义会有记录，您可以快速回滚。
- 您可以将测试或者线上的某个 API 下线，下线之后，与策略、密钥、 APP 的绑定或者授权关系依然存在，再次上线时会自动生效。如果要解除关系，需要专门操作删除。

API 授权管理

您的 API 如果上架到市场，那么购买者有权利操作给自己的某个 APP 授权。

如果不经过购买行为，您需要在线下跟合作伙伴建立使用关系，那么您需要通过授权来建立 API 和 APP 的权限关系。您将 API 发布到线上环境后，需要给客户的 APP 授权，客户才能用该 APP 进行调用。您有权对此类授权操作建立或者解除某个 API 与某个 APP 的授权关系， API 网关会对权限关系进行验证。操作授权时，您需要注意以下几点：

- 您可以将一个或者多个 API 授权给一个或者多个 APP 。批量操作时，建议不要同时操作多个分组下的 API 。
- 批量操作时，先选择 API 后选择环境。比如一个 API 在测试和线上均有发布，最后选择了测试，就只会将测试下的该 API 授权。
- 您可以通过客户提供给您的AppID或者阿里云邮箱账号来定位 APP 。
- 当您需要解除某个 API 下某个 APP 的授权时，您可以查看 API 的授权列表，在列表页进行解除。

历史与版本切换

您可以查看您每个 API 的发布历史记录，包括每次发布的版本号、说明、环境、时间和具体定义内容。

查看历史时，您可以选定某个版本然后操作切换到此版本，该操作会使该版本直接在指定环境中替换之前的版本，实时生效。

插件

签名密钥

什么是签名密钥

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，网关向您后端服务请求时会将密钥计算后一起传过去，您后端获取相应的字符串做对称计算，就可以对网关做身份验证。使用时您需要了解以下几点：

- 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，可以与 API 绑定或者解绑。
- 您将密钥绑定到 API 之后，由网关抛向您服务后端的该 API 的请求均会加上签名信息。您需要在后端做对称计算来解析签名信息，从而验证网关的身份。具体 HTTP 加签说明请查看文档——后端签名密钥说明文档

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。

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。

监控预警

- API网关为您提供可视化的实时监控和预警。您可以获得您开放的API被调用数据，包括调用量、流量、响应时间、错误分布等多个维度、多种时间单位的数据统计结果。同时支持历史数据查询，以便您统筹分析。
- 后续我们会陆续支持报表输出，给您提供最周到的数据支持。
- 您还可以配置预警，以便实时掌握API运行情况。

API 网关使用限制

限制项	限制描述
使用API网关服务的用户限制	用户需实名认证
用户创建API分组数量限制	每个账号下，API分组的个数上限是100个
用户创建API数量限制	每个API分组下，最多可以创建1000个API。即每个账号最多可创建100*1000 = 100000个API。
用户的API分组绑定独立域名个数限制	每个分组最多绑定5个独立域名
API的TPS限制	每个API分组接受访问的TPS上限为500。如需要将该限制调高，请提工单申请。对高TPS配置，网关会收取一定费用。
API分组官方二级域名限制	API分组创建成功后，API网关会为分组颁发二级域名。该域名用于测试该分组下的API，访问次数限制为1000次/天。请不要直接使用该二级域名对外提供API服务。
参数大小限制	Body位置的参数（包括Form和非Form形式）总体不能超过 2 Mb，其他位置的参数（包括Header和Query）总体不能超过 128 Kb。

后端签名密钥说明文档

概述

- API 网关提供后端 HTTP 服务签名验证功能，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）。
- 开启后端签名后 API 网关到后端HTTP服务的请求将会添加签名信息，后端 HTTP 服务读取 API 网关的签名字符串，然后对收到的请求进行本地签名计算，比对网关与本地签名结果是否一致。
- 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（专属网络VPC环境开放API），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：

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
```

计算签名

```
Mac hmacSha256 = Mac.getInstance("HmacSHA256");
byte[] keyBytes = secret.getBytes("UTF-8");
hmacSha256.init(new SecretKeySpec(keyBytes, 0, keyBytes.length, "HmacSHA256"));
String sign = new String(Base64.encodeBase64(Sha256.doFinal(stringToSign.getBytes("UTF-8")),"UTF-8"));
```

secret 为绑定到 API 上的签名密钥

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
注意: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
```

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：

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

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。

OpenID Connect 认证

OpenID Connect 是一套基于 OAuth 2.0 协议的轻量认证级规范，提供通过 API 进行身份交互的框架。较 OAuth 而言， OpenID Connect 方式除了认证请求之外，还标明请求的用户身份。

API 网关依据 OpenID Connect 的标准，提供两种认证方式：

OpenID Connect

标准的OpenID Connect模式，调用API时先通过用户名密码获取Token，之后的每次API请求都通过Token来验证。

OpenID Connect & 阿里云APP

在OpenID Connect基础上增加Appkey认证， 会同时验证API请求中的 Appkey和Token（Token 由 API 提供者的系统颁发，网关颁发 Appkey）。

两种OpenID Connect认证区别

OpenID Connect & 阿里云APP需要认证APPkey，OpenID Connect则不需要。无论哪种模式，都推荐具有登陆态的APP、服务端或者前端程序来使用。

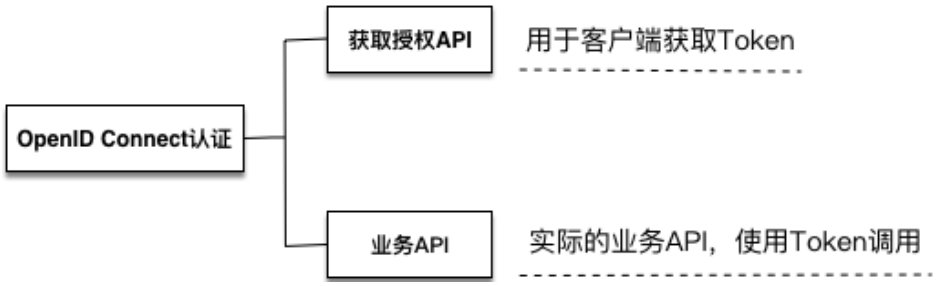
使用这两种认证方式建议：配置好流量控制，避免恶意用户进行暴力破解用户名/密码。

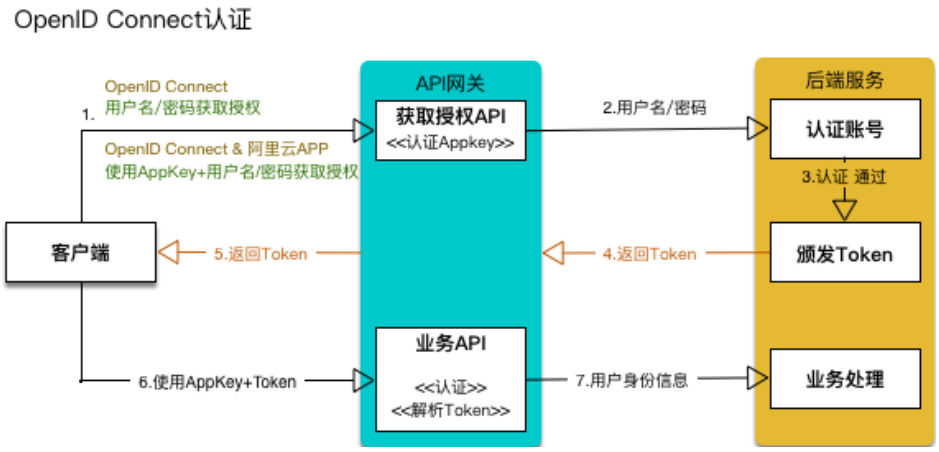
使用标准OpenID Connect将无法使用的功能

- APP鉴权，无法使用。
- APP级流量控制，无法配置/无法生效。
- 阿里云用户级流量控制，无法配置/无法生效。

实现原理

使用 OpenID Connect认证，需要您在 API 网关配置**获取授权 API** 和 **业务 API** 两种类型的API 。





- 获取授权API：用于您的客户端获取Token 。配置这个API时，您需要告知API网关您Token对应的Key和解析Token使用的公钥。
- 业务类接口，是您实际的业务接口，比如获取用户息、进行某个操作等。配置这类API时，你需要告知API网关你请求中表示Token的参数名称。当客户端调用这类API的请求到达API网关后，API网关自动验证这个请求的Appkey和Token是否合法。

认证方式

客户端调用 “获取授权 API” 获取Token的流程

i. 客户端使用认证信息来获取Token，
   i. OpenID Connect：需要使用用户的**“用户名/密码”**调用**“获取授权” API** 获取Token。。
   ii. OpenID Connect & 阿里云APP： **“Appkey 签名” + “用户名/密码”** 调用 **“获取授权” API** 获取Token。
   iii. 调用API请参照：**调用API**

API 网关收到请求后，OpenID Connect & 阿里云APP模式（OpenID Connect不需要）认证您的 **Appkey**后，调用后端服务的账号系统认证您传递的 **“用户名/密码”** 。

后端服务认证认证用户名/密码，通过后返回 **Token** 给您，您可凭 **Token** 来调用 **“业务API”** 。

客户端调用业务类 API ，来实现业务功能

客户端使用 **“获取授权 API”** 得到的 **Token** 和 **签名后的 Appkey** 来调用 **“业务API”** 。（调用API请参照：**调用API**）

API 网关认证、解析 **Token** 的内容，并将 **Token** 中包含的用户信息传递给后端。

在此阶段的操作中，API 的提供者需要事先进行如下操作：

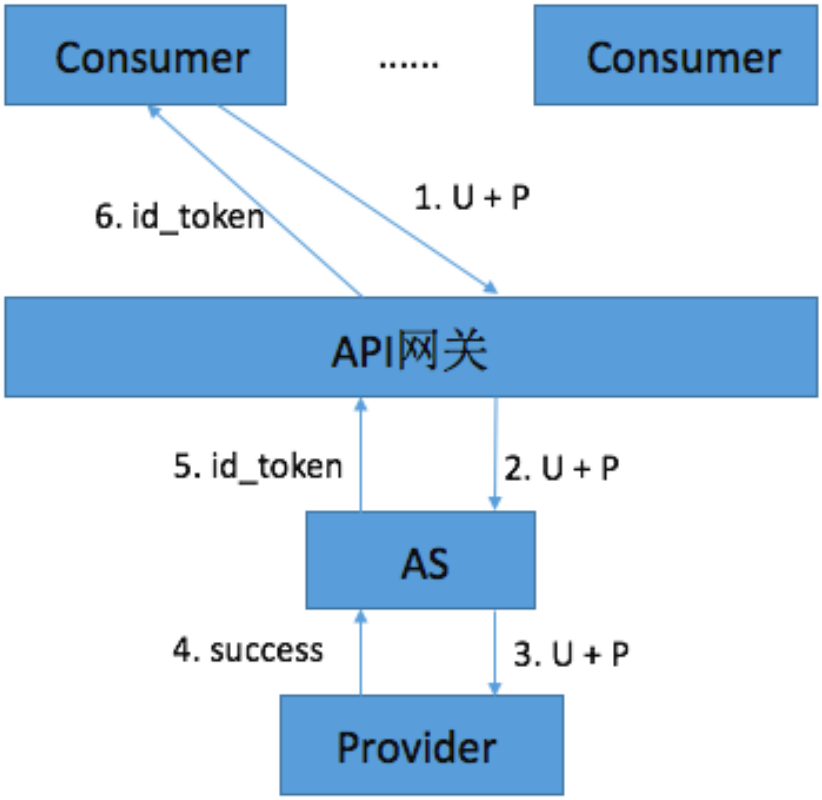
a. 开放账号系统，允许 API 网关对请求中 **用户名/密码** 进行验证，并依据网关提供的加密方式，颁发 Token。详细内容请参照下文 **如何实现 AS 模块**。
b. 在 API 网关定义 API。详细内容参照下文 **在 API 网关配置 API**。
注意：**用户名/密码** 是极为敏感的信息，在网络中明文传输存在风险，建议在传输前对用户名密码再次加密，并使用 HTTPS 协议传输。

实现方案简介

实现方案分为两个重要的部分：

1. Authorization server（AS）：认证服务器，负责生成 id_Token 并管理公钥私钥对。

这一步需要您自行实现。实现方法，请参照下文 **在 API 网关配置 API**。



参考上图，流程简述如下：

1. Consumer（调用者）向API网关发送获取 id_token 认证请求，比如：通过用户名和密码（U+P）的方式。
2. API 网关透传该请求到 AS。
3. AS 向 Provider（服务提供方）发送认证用户信息请求。

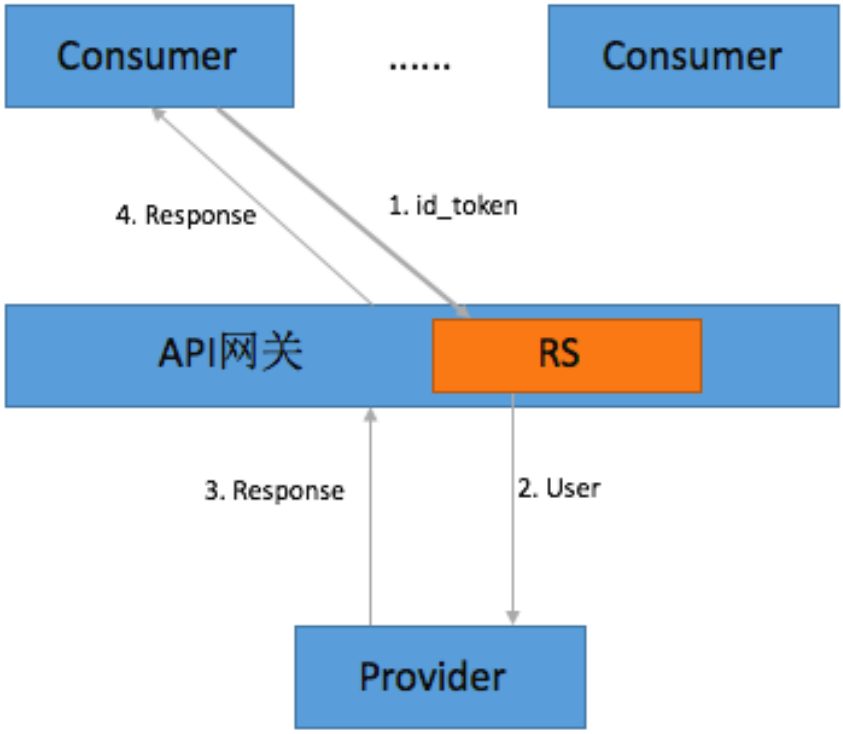
4. Provider 响应认证结果，若失败则直接响应错误信息。
5. 认证结果成功，AS 生成 id_token，id_token 中包含了 User 信息（可扩展，也可包含其他必要信息）。

   API 网关将 AS 返回的 id_token 响应给 Consumer。

   说明：AS 不用必须是单独部署的应用，完全可以集成在 Provider 中，在整个体系中担任 id_token 生产者角色，所生成的 id_token 必须符合 OIDC（1.0版本）协议中的 规范。

2. Resource server（RS）：资源服务器，负责校验 id_token，并解析出相应的信息。

此部分由网关来完成。因为 API 网关目前已经集成了 RS 功能，服务提供方只需要按照相应的加密规则生成 id_token 即可。



参考上图，流程简述如下：

1. Consumer 用带有 id_token 的参数去请求 API 网关。
2. API 网关会保存校验所使用的公钥，验证并解析 id_token 获取其中的 User 信息传给 Provider，若验证失败则直接返回错误信息。
3. Provider 处理请求并返回结果给 API 网关。
4. API 网关透传 Provider 响应的结果给 Consumer。

说明：RS 在整个体系中担任 id_token 消费者角色，只有 id_token 校验通过，才能将请求转发给 Provider。

如何实现 AS 模块

AS 中使用 OIDC 生成 id_token 的说明

- id_token，也叫 ID Token，是在 OIDC 协议中定义的一种令牌，详细内容参见 OpenID Connect Core 1.0。
- id_token 生成需要 KeyPair, keyId 与 Claims (有关Claims更多信息请访问 ID_Token）。

KeyId 说明

KeyId 必须保证唯一，比如使用 UUID 生成的长度至少32位的随机字符串，可以全为数字或数字+字母。
参考示例（JAVA）

```
String keyId = UUID.randomUUID().toString().replaceAll("-", "");
```

或

```
String keyId = String.valueOf(UUID.randomUUID().getMostSignificantBits()) +
String.valueOf(UUID.randomUUID().getMostSignificantBits());
```

KeyPair 说明

KeyPair 是一个基于 PKI 体系的非对称算法的公私钥组合，每一对包括公钥（publicKey）与私钥（privateKey）； 公钥放置在 RS 中，在校验（verify）时使用，私钥放置在 AS 中，在生成 id_token 时做数字签名使用；
KeyPair 使用 RSA SHA256 加密算法，为保证足够安全其加密的位数为2048；
AS 中使用的 KeyPair 均为 JSON 格式的数据，一个示例如下：
publicKey:

```
{"kty":"RSA","kid":"671741829679797099139504717892261817 21","alg":"ES256","n":"oH5WunqaqIopfOFBz9RfBVVII
cmk0WDJagAcROKFiLJScQ8N\_nrexgbCMlu-dSCUWq7XMnp1ZSqw-XBS2-XEy4W4l2Q7rx3qDWY0cP8pY83hqxTZ6-
8GErJm\_0yOzR4WO4plIVVWt96-
mxn3ZgK8kmaeotkS0zS0pYMb4EEOxFFnGFqjCThuO2pimF0imxiEWw5WCdREz1v8RW72WdEfLpTLJEOpP1FsFyG3OI
DbTYOqowD1YQEf5Nk2TqN\_7pYrGRKsK3BPpw4s9aXHbGrpwsCRwYbKYbmeJst8MQ4AgcorE3NPmp-
E6RxA5jLQ4axXrwC0T458LIVhypWhDqejUw","e":"AQAB"}
```

privateKey:

```
{"kty":"RSA","kid":"671741829679797099139504717892261817 21","alg":"ES256","n":"oH5WunqaqIopfOFBz9RfBVVII
cmk0WDJagAcROKFiLJScQ8N\_nrexgbCMlu-dSCUWq7XMnp1ZSqw-XBS2-XEy4W4l2Q7rx3qDWY0cP8pY83hqxTZ6-
8GErJm\_0yOzR4WO4plIVVWt96-
mxn3ZgK8kmaeotkS0zS0pYMb4EEOxFFnGFqjCThuO2pimF0imxiEWw5WCdREz1v8RW72WdEfLpTLJEOpP1FsFyG3OI
```

DbTYOqowD1YQEf5Nk2TqN_7pYrGRKsK3BPpw4s9aXHbGrpwsCRwYbKYbmeJst8MQ4AgcorE3NPmp-E6RxA5jLQ4axXrwC0T458LIVhypWhDqejUw","e":"AQAB","d":"aQsHnLnOK-1xxghw2KP5JTZyJZsiwt-ENFqqJfPUzmlYSCNAV4T39chKpkch2utd7hRtSN6Zo4NTnY8EzGQQb9yvunaiEbWUkPyJ6kM3RdlkkGLvVtp0sRwPCZ2EAYBlsMad9jkyrtmdC0rtf9jerzt3LMLC7XWbnpC3WAl8rsRDR1CGs_-u4sfZfttsaUbJDD9hD0q4NfLDCVOZoQ_8wkZxyWDAQGCe6GcCbu6N81fTp2CSVbiBj7DST_4x2NYUA2KG8vyZYcwviNTxQzk4iPfdN2YQz_9aMTZmmhVUGlmTvAjE5ebBqcqKAS0NfhOQHg2uR46eBKBy_OyVOLohsQ","p":"8Tdo3DCs-0t9JMtM0lYqPRP4wYJs37Rv6S-ygRui2MI_hadTY9I2A199JMYw7Fjke_wa3gqJLa98pbybdLWkrOxXbKEkwE4uc4-fuNjLbUTC5tqdM5-nXmpL887uREVYnk8FUzvWeXYTCNCb7OLw5l8yPJ1tR8aNcd0fJNDKh98","q":"qlRrGSTsZzBkDgDi1xlCoYvoM76cbmxrCUK-mc_kBRHfMjlHosxFUnAbxqIBE4eAJEKVfIJLQrHFvIDjQb3kM9ylmwMCu9f8u9DHrT8J7LSDlLqDaXuiM2oiKtW3bAaBPuiR7sVMFcuB5baCebHU487YymJCBTfeCZtFdi6c4w0","dp":"gVCROKonsjiQCG-s6X4j-saAL016jJsw-7QEYE6uiMHqR_6iJ_uD1V8Vuec-RxaItyc6SBsh24oeqsNoG7Ndaw7w912UVDwVjwJKQFCJDjU0v4oniItosKcPvM8M0TDUB1qZojuMCWWRYsJjNSWcvAQA7JoBAd-h6I8AqT39tcU","dq":"BckMQjRg2zhnjZo2Gjw_aSFJZ8iHo7CHCi98LdlD03BB9oC_kCYEDMLGDr8d7j3h-llQnoQGbmN_ZeGy1l7Oy3wpG9TEWQEDEpYK0jWb7rBK79hN8l1CqyBlvLK5oi-uYCaiHkwRQ4RACz9huyRxKLOz5VvlBixZnFXrzBHVPlk","qi":"M5NCVjSegf_KP8kQLAudXUZi_6X8T-owtsG_gB9xYVGnCsbHW8gccRocOY1Xa0KMotTWJl1AskCu-TZhOJmrdeGpvkdulwmbIcnjA_FgfIp4lAj4TCWmtRI6982hnC3XP2e-nf_z2XsPNiuOactY7W042D_cajyyX_tBEJaGOXM"}

生成 KeyPair 参考示例（JAVA）

```
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;



String keyId = UUID.randomUUID().toString().replaceAll("-", "");
RsaJsonWebKey jwk = RsaJwkGenerator.generateJwk(2048);
jwk.setKeyId(keyId);
jwk.setAlgorithm(AlgorithmIdentifiers.ECDSA_USING_P256_CURVE_AND_SHA256);
String publicKey = jwk.toJson(RsaJsonWebKey.OutputControlLevel.PUBLIC_ONLY);
String privateKey = jwk.toJson(RsaJsonWebKey.OutputControlLevel.INCLUDE_PRIVATE);
```

生成 id_token 参考步骤

通过 OIDC 协议中定义的 Claims 属性(aud, sub, exp, iat, iss)与其属性值，生成 Claims(全称 JwtClaims)

示例代码(JAVA)

```
JwtClaims claims = new JwtClaims();
```

```
claims.setGeneratedJwtId();
claims.setIssuedAtToNow();
//expire time
NumericDate date = NumericDate.now();
date.addSeconds(120);
claims.setExpirationTime(date);
claims.setNotBeforeMinutesInThePast(1);
claims.setSubject("YOUR_SUBJECT");
claims.setAudience("YOUR_AUDIENCE");
//添加自定义参数

claims.setClaim(key, value);
```

通过 keyId, Claims, privateKey 与使用的数字签名算法 (RSA SHA256)生成 JWS(Json Web Signature)

示例代码(JAVA)

```
JsonWebSignature jws = new JsonWebSignature();
jws.setAlgorithmHeaderValue(AlgorithmIdentifiers.RSA_USING_SHA256);
jws.setKeyIdHeaderValue(keyId);
jws.setPayload(claims.toJson());
PrivateKey privateKey = new RsaJsonWebKey(JsonUtil.parseJson(privateKeyText)).getPrivateKey();
jws.setKey(privateKey);
```

通过 JWS 获取 id_token 值

示例代码(JAVA)

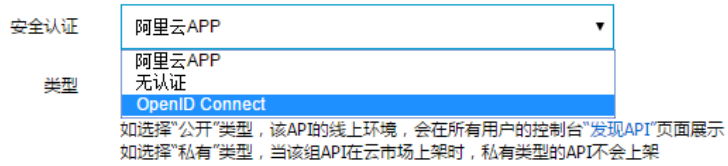
```
String idToken = jws.getCompactSerialization();
```

一个生成的 id_token 示例：

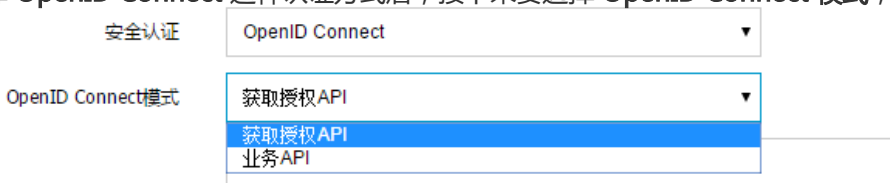
```
eyJhbGciOiJSUzI1NiIsImtpZCI6Ijg4NDgzNzI3NTU2OTI5MzI2NzAzMzA5OTA0MzUxMTg1ODE1NDg5In0.eyJ1c2VySWQiOiIzMzcwMTU0NDA2ODI1OTY4NjI3IiwidGFnTmFtZSI6ImNvbmFuVGVzdCIsImV4cCI6MTQ4MDU5Njg3OSwiYXVkIjoiQWxpX0FQSV9Vc2VyIiwianRpIjoiTm9DMFVVeW5xV0N0RUFEVjNoeEIydyIsImlhdCI6MTQ4MDU5MzI3OSwibmJmIjoxNDgwNTkzMjE5LCJzdWIiOiJ7ZGF0YU1hcD0ne3VzZXJJZD0zMzcwMTU0NDA2ODI1OTY4NjI3fScsIHN0YXR1c0NvZGU9JzAnLCBlcnJvcnM9J1tdJ30ifQ.V3rU2VCziSt6uTgdCktYRsIwkMEMsO_jUHNCCIW_Sp4qQ5ExjtwNt9h9mTGKFRujk2z1E0k36smWf9PbNGTZTWmSYN8rvcQqdsupcC6LU9r8jreA1Rw1CmmeWY4HsfBfeInr1wCFrEfZl6_QOtf3raKSK9AowhzEsnYRKAYuc297gmV8qlQdevAwU75qtg8j8ii3hZpJqTX67EteNCHZfhXn8wJjckl5sHz2xPPyMqj8CGRQ1wrZEHjUmNPw-unrUkt6neM0UrSqcjlrQ25L8PEL2TNs7nGVdl6iS7Nasbj8fsERMKcZbP2RFzOZfKJuaivD306cJIpQwxfS1u2bew
```

在 API 网关配置 API

API 编辑功能中，基本信息栏目中安全认证增加 **OpenID Connect** 选项，这种方式同时也包括了 **阿里云APP** 认证方式，也就是说只有被授权的 APP 才能调用这个 API。

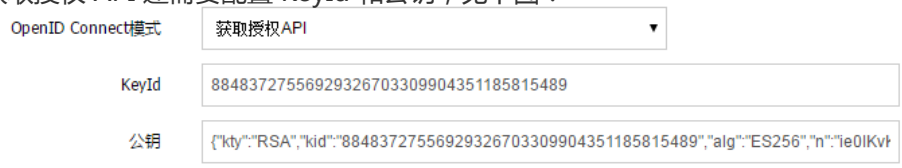


选择 **OpenID Connect** 这种认证方式后，接下来要选择 **OpenID Connect 模式**，有两个选项：



i. 获取授权 API：您用来换取 Token 的 API，比如：通过 U+P 换取 Token。
ii. 业务 API：也就是服务提供商提供服务的 API，调用者会把之前获取到的 Token 作为入参进行调用。
OpenID Connect 认证方式主要包含的就是以上这两种 API，下文将分别说明这两种API是如何配置的。

获取授权 API 还需要配置 KeyId 和公钥，见下图：



KeyId：公钥私钥对 对应的一个唯一 Id，由 As 模块负责生成的；示例：

```
88483727556929326703309904351185815489
```

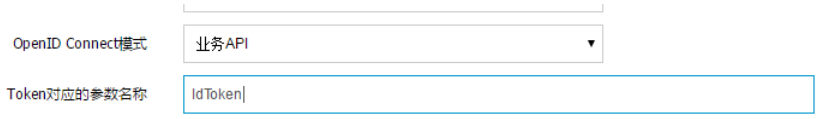
公钥：负责验证和解析 Token，由 As 模块负责生成的，示例：

```
{"kty":"RSA","kid":"88483727556929326703309904351185815489","alg":"ES256","n":"ie0IKvKLd7Y3izHcZemdDsVVXg5QtWtGF7XEkILnn66R2\_3a30DikqV409OVL7Hv0ElACgCaBLEgZeGHTcdLE1xxDTna8MMBnBNuMVghvFERCKh8uzpxlQsfcnFd5IFdJWj1x5Tscetrow6lA3h5zYx0rF5TkZzC4DclxgDmITRam0dsHBxr3uk9m9YYBz2mX0ehjY0px7vIo7hZH2J3gODEPorIZkk3x8GPdlaA4P9OFAO4au9-zcVQop9vLirxdwDedk2p-F9GP6UiQC9V2LTWqkVw\_oPBf9Rlh8Qdi19jA8SeCfzAxJZYlbOTK8dYAFAVEFsvXCFvdaxQefwWFw","e":"AQAB"}
```

设置完这些，后面的设置就和之前普通的 API 一样了，在此就不赘述了。

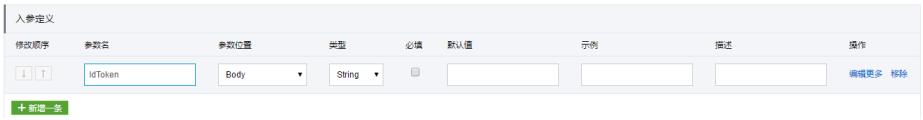
不管是创建 API 还是修改 API，所设置的 KeyId 和公钥都是在 API 发布之后才会生效。

业务 API 需要配置 **Token所对应的参数名称。**



如上图，Token 所对应的参数名称：就是调用者调用 API 传入 id_token 所使用的参数名，API 网关会识别这个参数值，校验并解析。

然后在入参定义中，必须要定义一个对应参数，否则系统会出现错误提示，见下图。



iii. 设置自定义系统参数：业务 API在 **定义 API 后端服务** 标签中会开放配置自定义系统参数，举个例子，见下图：

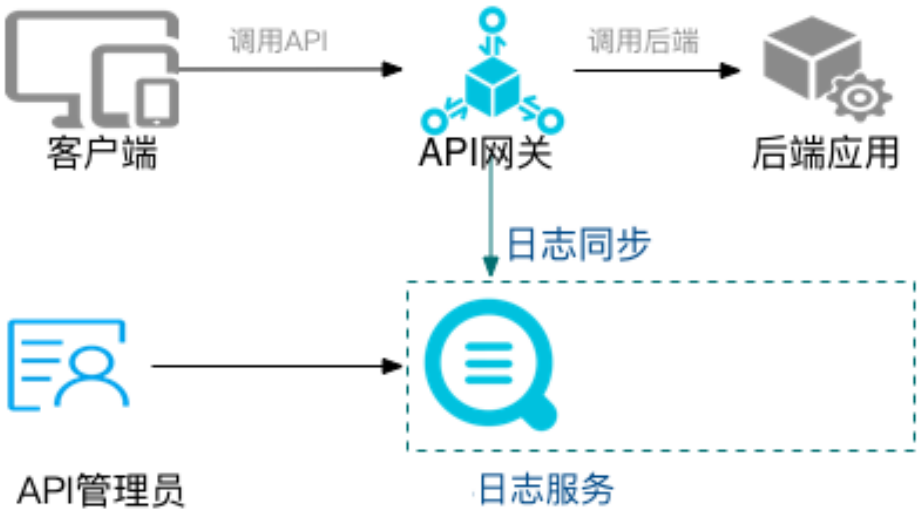
比如：AS 生成的 id_token 中包含了调用者的 userId，那么如果按照上图这样配置的话，就会把从调用者传过来的 id_token 中解析出来的 userId 传给服务提供方，配置方式和系统参数类似。

除了以上3处，定义 API 其他配置和前文一样，也就不再赘述了。

以上就是 第三方账号认证 OpenID Connect 在 API 网关配置的全部内容，供您参考！

通过日志服务查看API调用日志

API网关和日志服务实现无缝集成，通过日志服务您可以进行实时日志查询、下载、多维度统计分析等，您也可以将日志投递到OSS或者MaxCompute。



更多日志服务功能请参照：**日志服务帮助文档。**

日志服务每个月前500MB免费，具体价格请参照：**日志服务定价。**

1 功能简介

1.1 日志在线查询

可根据日志中任意关键字进行快速的精确、模糊检索，可用于问题定位或者统计查询。

1.2 详细调用日志

您可以检索API调用的详细日志包含：

日志项	描述
apiGroupUid	API的分组ID
apiGroupName	API分组名称
apiUid	API的ID
apiName	API名称
apiStageUid	API环境ID
apiStageName	API环境名称
httpMethod	调用的HTTP方法
path	请求的PATH
domain	调用的域名

statusCode	HttpStatusCode
errorMessage	错误信息
appId	调用者应用ID
appName	调用者应用名称
clientIp	调用者客户端IP
exception	后端返回的具体错信息
providerAliUid	API提供者帐户ID
region	区域，如：cn-hangzhou
requestHandleTime	请求时间，UTC
requestId	请求ID，全局唯一
requestSize	请求大小，单位：字节
responseSize	返回数据大小,单位：字节
serviceLatency	后端延迟，毫秒

1.4 自定义分析图表

您可以根据统计需求将任意日志项自定义统计图表，以满足您日常的业务需要。

1.3 预置分析报表

为能让您快速使用，API网关预定义了一些统计图表（全局）。包括：请求量大小、成功率、错误率、延时情况、调用API的APP数量，错误情况统计、TOP 分组、TOP API、Top 延迟等等。

2 使用日志服务查看API日志

2.1 配置日志服务

使用此功能前，请确保您已经开通了日志服务，并创建Project和Logstore，点击开始创建。

您可以在API网关的控制台上来配置，也可以在日志服务控制台上来配置。

2.1.1 在API网关控制台配置

1）打开API网关控制台-【开放API】-【日志管理】，选择您服务所在的区域，下图以华北1为例：

2）点击”创建日志配置”，进入日志配置界面

3）选择您所需要输入的日志服务的Project或者Logstore，若无法选择，请点击“授权日志写操作”，点击后会进行授权操作

4）确认后，完成网关与日志服务的关联

5）开启索引后完成配置

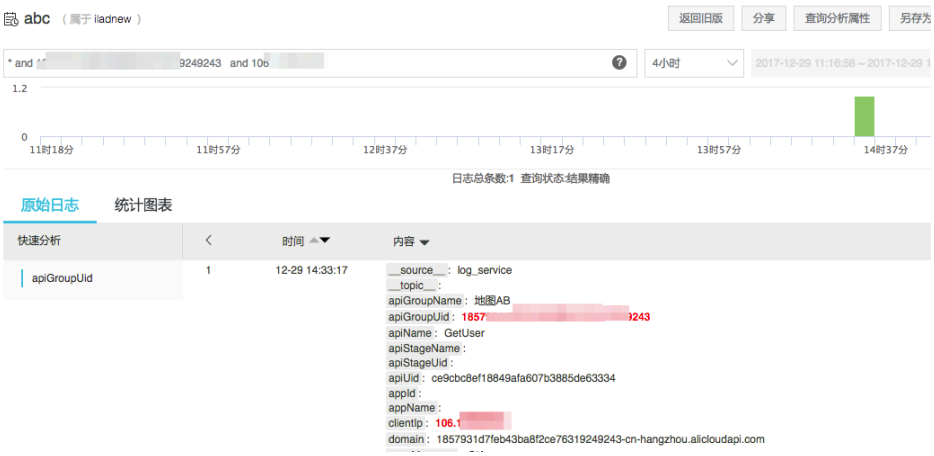
2.1.1.2 在日志服务控制台配置

您可以参照：API网关访问日志

配置完成后，您的API调用记录即可进入日志服务的Logstore中

2.2 查看日志

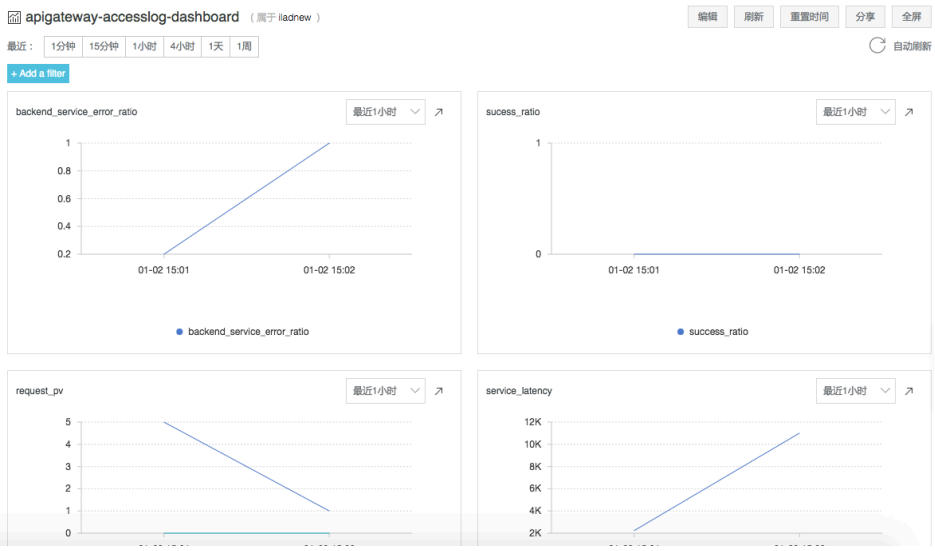
你可以点击API网关控制台-【开放API】-【日志管理】-“访问日志”，跳转到日志控制台，根据日志服务的查询语法，在线检索调用日志。如图：



你也可以登录日志服务控制台查看日志，可参照：查询日志

2.3 查询预定义报表

预定义报表，是API网关为了方便用户统计查询，而预置的一些统计报表，可以点击API网关控制台-【开放API】-【日志管理】-“日志统计”来访问。也可以在日志服务控制台来查看:



2.4 自定义查询报表

你可以根据自身业务需要自定义查询报表，请参照定义方法：仪表盘

3 维护日志

您打开API网关控制台-【开放API】-【日志管理】，进行“修改配置”或者删除配置。

- **修改配置**：为更换新日志服务的Project或者Logstore，更换后的API调用日志将会写入新的Logstore，但历史数据仍会保留在原Logstore中，不会跟随迁移。
- **删除配置**：为删除API网关与日志服务的映射关系，删除后将不会再同步API调用日志到日志服务，但并不会删除Logstore中已有的数据。

使用 RAM 管理 API

API 网关结合阿里云访问控制（RAM）来实现企业内多职员分权管理 API。API 提供者可以为员工建立子账户，并控制不同职员负责不同的 API 管理。

- 使用 RAM 可以允许子帐号，查看、创建、管理、删除 API 分组、API、授权、流控策略等。但子帐号不是资源的所有者，其操作权限随时都可以被主帐号收回。
- 在查看本文前，请确保您已经祥读了 RAM 帮助手册 和 API 网关 API 手册。
- **若您不无此业务场景， 请跳过此章节。**

你可以使用 RAM 的控制台 或者 API 来添加操作。

第一部分：策略管理

授权策略（Policy），来描述授权的具体内容，授权内容主要包含效力(Effect)、资源(Resource)、对资源所授予的操作权限(Action)以及限制条件(Condition)这几个基本元素。

系统授权策略

API 网关已经预置了两个系统权限，AliyunApiGatewayFullAccess和AliyunApiGatewayReadOnlyAccess，可以到 RAM 的在 RAM 控制台-策略管理 进行查看。

- AliyunApiGatewayFullAccess：管理员权限，拥有主帐号下包含 API 分组、API、流控策略、应用等所有资源的管理权限。
- AliyunApiGatewayReadOnlyAccess：可以查看主帐号下包含 API 分组、API、流控策略、应用等所有资源，但不可以操作。

自定义授权策略

您可以根据需要自定义管理权限，支持更为精细化的授权，可以为某个操作，也可以是某个资源。如：API

GetUsers 的编辑权限。可以在 RAM 控制台-策略管理-自定义授权策略查看已经定义好的自定义授权：自定义授权查看、创建、修改、删除方法请参照：授权策略管理。
授权策略内容填写方法请参照：Policy 基本元素 和 Policy 语法结构 和下文的授权策略。

第二部分：授权策略

授权策略是一组权限的集合，它以一种策略语言来描述。通过给用户或群组附加授权策略，用户或群组中的所有用户就能获得授权策略中指定的访问权限。
授权策略内容填写方法请参照：Policy 基本元素 和 Policy 语法结构。
示例：

```
{
"Version": "1",
"Statement": [
{
"Action": "apigateway:Describe*",
"Resource": "*",
"Effect": "Allow"
}
]
}
```

此示例表示：允许所有的查看操作

Action（操作名称列表）格式为：

```
"Action":"<service-name>:<action-name>"
```

其中：

- **service-name** 为：阿里云产品名称，请填写 **apigateway**。
- **action-name** 为：API 接口名称，请参照下表，支持通配符 ***。

```
"Action": "apigateway:Describe*" 表示所有的查询操作
" Action": "apigateway:*" 表示 API 网关所有操作
```

第三部分：Resource（操作对象列表）

Resource 通常指操作对象， API 网关中的 API 分组、流控策略、应用都被称为 Resource，书写格式：

```
acs:<service-name>:<region>:<account-id>:<relative-id>
```

其中：

- **acs**：Alibaba Cloud Service 的首字母缩写，表示阿里云的公有云平台。

- **service-name** 为：阿里云产品名称，请填写 apigateway。
- **region**：地区信息，可以使用通配符*号来代替，*表示所有区域。
- **account-id**：账号 ID，比如 1234567890123456，也可以用*代替。
- **relative-id**：与 API 网关相关的资源描述部分，这部分的格式描述支持类似于一个文件路径的树状结构。

示例：

```
acs:apigateway:$regionid:$accountid:apigroup/$groupId
```

书写：

```
acs:apigateway:*:$accountid:apigroup/
```

请结合 API 网关的 API 手册 来查看下表

action-name	资源(Resource)
AbolishApi	acs:apigateway:$regionid:$accountid:apigroup/$groupId
AddTrafficSpecialControl	acs:apigateway:$regionid:$accountid:trafficcontrol/$trafficcontrolid
CreateApi	acs:apigateway:$regionid:$accountid:apigroup/$groupId
CreateApiGroup	acs:apigateway:$regionid:$accountid:apigroup/*
CreateTrafficControl	acs:apigateway:$regionid:$accountid:trafficcontrol/*
DeleteAllTrafficSpecialControl	acs:apigateway:$regionid:$accountid:trafficcontrol/$trafficcontrolid
DeleteApi	acs:apigateway:$regionid:$accountid:apigroup/$groupId
DeleteApiGroup	acs:apigateway:$regionid:$accountid:apigroup/$groupId
DeleteDomain	acs:apigateway:$regionid:$accountid:apigroup/$groupId
DeleteDomainCertificate	acs:apigateway:$regionid:$accountid:apigroup/$groupId
DeleteTrafficControl	acs:apigateway:$regionid:$accountid:trafficcontrol/$trafficcontrolId
DeleteTrafficSpecialControl	acs:apigateway:$regionid:$accountid:trafficcontrol/$trafficcontrolId
DeployApi	acs:apigateway:$regionid:$accountid:apigroup/$groupId

DescribeApi	acs:apigateway:$regionid:$accountid:apigroup/$groupId
DescribeApiError	acs:apigateway:$regionid:$accountid:apigroup/$groupId
DescribeApiGroupDetail	acs:apigateway:$regionid:$accountid:apigroup/$groupId
DescribeApiGroups	acs:apigateway:$regionid:$accountid:apigroup/*
DescribeApiLatency	acs:apigateway:$regionid:$accountid:apigroup/$groupId
DescribeApiQps	acs:apigateway:$regionid:$accountid:apigroup/$groupId
DescribeApiRules	acs:apigateway:$regionid:$accountid:apigroup/$groupId
DescribeApis	acs:apigateway:$regionid:$accountid:apigroup/$groupId
DescribeApisByRule	acs:apigateway:$regionid:$accountid:trafficcontrol/$trafficcontrolId oracs:apigateway:$regionid:$accountid:secretkey/$secretKeyId
DescribeApiTraffic	acs:apigateway:$regionid:$accountid:apigroup/$groupid
DescribeAppsByApi	acs:apigateway:$regionid:$accountid:apigroup/$groupId
AddBlackList	acs:apigateway:$regionid:$accountid:blacklist/*
DescribeBlackLists	acs:apigateway:$regionid:$accountid:blacklist/*
DescribeDeployedApi	acs:apigateway:$regionid:$accountid:apigroup/$groupId
DescribeDeployedApis	acs:apigateway:$regionid:$accountid:apigroup/$groupId
DescribeDomain	acs:apigateway:$regionid:$accountid:apigroup/$groupId
DescribeDomainResolution	acs:apigateway:$regionid:$accountid:apigroup/$groupId
DescribeHistoryApi	acs:apigateway:$regionid:$accountid:apigroup/$groupId
DescribeHistoryApis	acs:apigateway:$regionid:$accountid:apigroup/*
DescribeRulesByApi	acs:apigateway:$regionid:$accountid:group/$groupId
DescribeSecretKeys	acs:apigateway:$regionid:$accountid:secretke

	y/*
DescribeTrafficControls	acs:apigateway:$regionid:$accountid:trafficcontrol/*
ModifyApi	acs:apigateway:$regionid:$accountid:apigroup/$groupId
ModifyApiGroup	acs:apigateway:$regionid:$accountid:apigroup/$groupId
ModifySecretKey	acs:apigateway:$regionid:$accountid:secretkey/$secretKeyId
RecoverApiFromHistorical	acs:apigateway:$regionid:$accountid:apigroup/$groupId
RefreshDomain	acs:apigateway:$regionid:$accountid:apigroup/$groupId
RemoveAccessPermissionByApis	acs:apigateway:$regionid:$accountid:apigroup/$groupId
RemoveAccessPermissionByApps	acs:apigateway:$regionid:$accountid:apigroup/$groupId
RemoveAllBlackList	acs:apigateway:$regionid:$accountid:blacklist/*
RemoveApiRule	acs:apigateway:$regionid:$accountid:apigroup/$groupId(acs:apigateway:$regionid:$accountid:secretkey/$secretKeyId oracs:apigateway:$regionid:$accountid:trafficcontrol/$trafficcontrolId)
RemoveAppsFromApi	acs:apigateway:$regionid:$accountid:apigroup/$groupId
RemoveBlackList	acs:apigateway:$regionid:$accountid:blacklist/$blacklistid
SetAccessPermissionByApis	acs:apigateway:$regionid:$accountid:apigroup/$groupId
SetAccessPermissions	acs:apigateway:$regionid:$accountid:apigroup/$groupId
SetApiRule	acs:apigateway:$regionid:$accountid:apigroup/$groupId(acs:apigateway:$regionid:$accountid:secretkey/$secretKeyId oracs:apigateway:$regionid:$accountid:trafficcontrol/$trafficcontrolId)
SetDomain	acs:apigateway:$regionid:$accountid:apigroup/$groupId
SetDomainCertificate	acs:apigateway:$regionid:$accountid:apigroup/$groupId
SwitchApi	acs:apigateway:$regionid:$accountid:apigroup/$groupId
CreateSecretKey	acs:apigateway:$regionid:$accountid:secretke

	y/*
DeleteSecretKey	acs:apigateway:$regionid:$accountid:secretkey/$secretKeyId

专属网络 VPC 环境开放 API

使用阿里云专属网络VPC，可以构建出一个隔离的网络环境，并可以自定义IP 地址范围、网段、路由表和网关等；此外，也可以通过专线/VPN/GRE等连接方式实现云上VPC与传统IDC的互联，构建混合云业务。

API网关也支持您部署在专属网络VPC中的服务开放API。在开始此文章之前，请确认您已经了解专属网络VPC的使用方法。

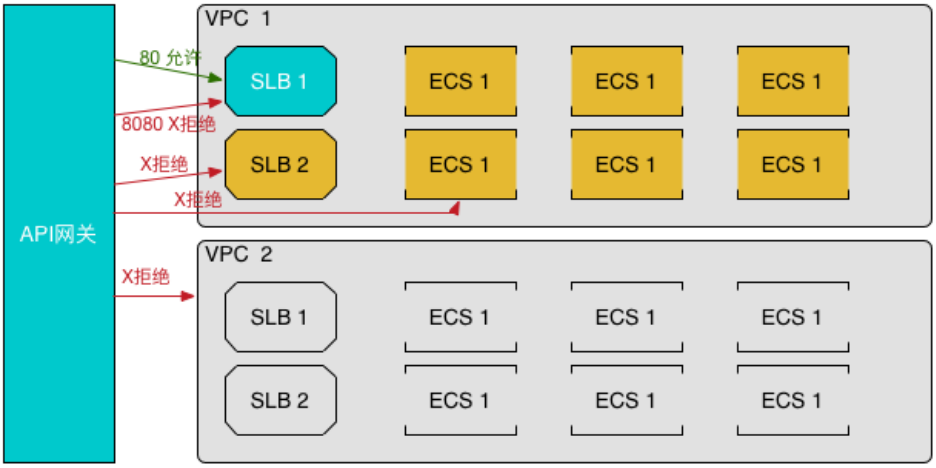
若您的后端服务在VPC环境，需要进行授权API网关访问才可开放相应API。创建API流程如下：

1 授权与绑定VPC

开放VPC环境的API，需要您先授权API网关可访问您VPC内的服务。授权时需指定API网关可以访问的资源+端口，如：SLB 的443端口、ECS 的80端口。

- 授权成功后，API网关将通过内网访问VPC内部资源
- 此授权只会被用作API网关访问相应后端资源
- API网关不可访问未被授权的资源或者端口

例如：只将VPC 1中SLB 1的80端口授权给API网关，那么API网关只能访问vpc 1中SLB 1的80端口



1.1 准备VPC环境

1) 购买VPC环境的，SLB、ECS，并搭建服务，可以参照VPC使用手册

2）查询VPC信息，需要准备如下VPC信息

- VPC ID：您后端服务所在的VPCID
- 实例ID：您后端所在的实例ID，可以是ECS或者SLB的实例ID，若使用了SLB，请填写SLB的实例ID
- 端口号：调用您后端服务所使用的端口号

1.2 授权API网关访问

进入【API网关控制台】-【开放API】-【VPC授权】，点击“创建授权”

进入授权页面，填写相应信息

- VPC名称：为此条授权的名称标识，供创建API时选择后端地址使用，所以为了便于后续管理，请保证此名称的唯一。

点击确定，完成授权

若您有多个VPC，或者要授权多个实例、端口，请重复如上步骤

2 创建API

创建API的流程与其他类型API方式一致，请参照：创建API。

在选择后端服务地址时，请选择：

- VPC通道：请选择“使用VPC通道”
- VPC授权：请选按需求选择所创建的授权

其他部分内容，与其他API定义方式一致

保存后，则API创建完成。

3 安全组授权

视情况必需：对于后端使用SLB，或者没有改动过ECS安全组授权策略的用户，可以跳过此步。

若您API的后端服务为ECS，且您修改过安全组“内网入方向”访问策略，需要增加问策略允许如下IP段访问（请根据服务所区域配置）。

区域	方向	IP
杭州	内网入方向	100.104.13.0/24
北京	内网入方向	100.104.106.0/24
深圳	内网入方向	100.104.8.0/24
上海	内网入方向	100.104.8.0/24
香港	内网入方向	100.104.175.0/24
新加坡	内网入方向	100.104.175.0/24
德国	内网入方向	100.104.72.0/24
亚太东南3（吉隆坡）	内网入方向	100.104.112.0/24
亚太南部1（孟买）	内网入方向	100.104.233.0/24
亚太东南5（雅加达）	内网入方向	100.104.72.0/24
亚太东北1（东京）	内网入方向	100.104.188.0/24
亚太东南2（悉尼）	内网入方向	100.104.143.192/26

3 API测试

可以到通过以下方式测试您的API

- 调试API
- 下载SDK
- 使用API调用Demo

4 解除授权

若您授权的资源或者端口不再提供服务，请删除相应授权。

5 FAQ

使用此功能是否有额外费用？

否，此功能免费使用，无额外费用产生。

是否可以绑定多个VPC？

可以，若您的后端服务在多个VPC，可以添加多个授权。

为什么我无法授权我的VPC

请确认，VPCID、实例ID和端口号的正确，并保证授权策略和VPC在同一个区域。

授权API网关后，我的VPC安全么？

当您的VPC授权API网关可访问，网关与VPC的网络联通。在安全方面做了限制，不会造成VPC的安全问题。

1. 安全控制授权操作：只有VPC的所有者能够操作授权。
2. 授权后API网关与VPC建立专享通道：他人不可使用此同道。
3. 授权是针对某个资源的端口：无其他端口或资源的访问权限。

Mock 您的 API

在项目开发过程中，往往是多个合作方一同开发，多个合作方相互依赖，而这种依赖在项目过程中会造成相互制约，理解误差也会影响开发进度，甚至影响项目的工期。所以在开发过程中，一般都会使用 Mock 来模拟最初预定的返回结果，来降低理解偏差，从而提升开发效率。

API 网关也支持 Mock 模式的简单配置。

配置 Mock

在 API 编辑页面——后端基础定义，来配置 Mock。

选择 Mock 模式可以选择使用 Mock 或者不使用 Mock，选择使用 Mock 时会进行二次确认

填写 Mock 返回结果Mock 返回结果，可以填写您真实的返回结果。目前支持是 Json、XMl、文本

等格式作为 Mock 返回结果。如

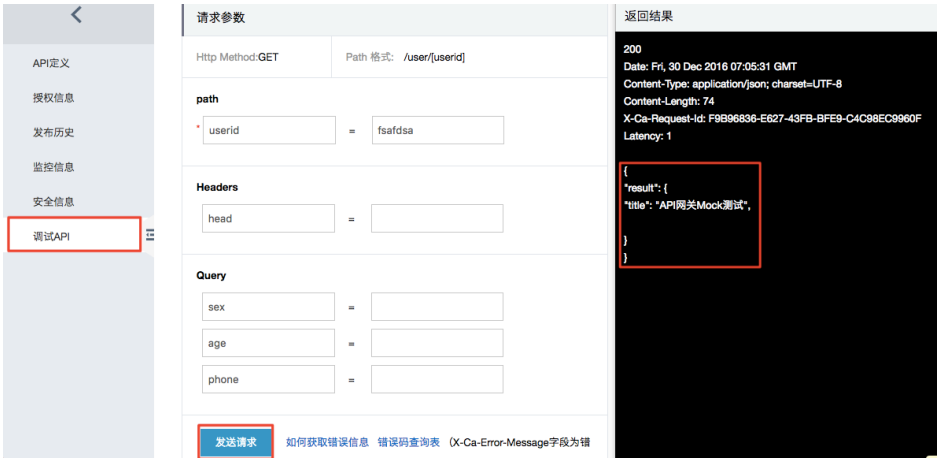
```
{
"result": {
"title": " API 网关 Mock 测试",

}
}
```

保存后 Mock 设置成功，请根据实际需要 **发布** 到测试或线上环境进行测试，也可以在 API 调试页面进行调试。

调试

可以在调试 API 页面发起 API 调用来测试设置结果：



这表示 Mock 设置成功。

解除 Mock

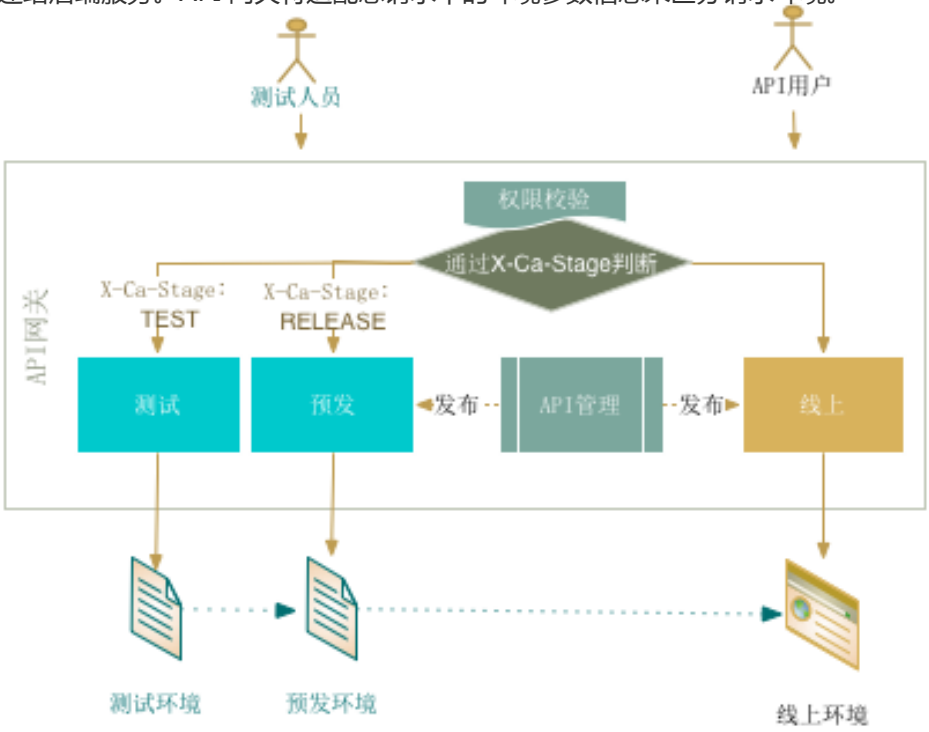
若您需要解除 Mock，可以将第一幅图中的 Mock，修改为 **不使用 Mock** 即可，而 Mock 返回结果中的值不会被清除，以便您进行下一次的 Mock 。修改完成后请 **发布**，只有发布后才会真正生效。

环境管理

什么是环境管理

目前每个 API 分组有三个环境：测试、预发和线上。其中测试环境和预发环境为测试人员测试或调试 API 时使用的环境，线上环境主要为 API 用户使用的环境。

通过 API 分组的 **环境管理** 设置环境变量参数，为 API 分组的测试、预发、线上环境分别定义一个变量。环境变量参数，即为每个环境自定义的公共常量参数。当调用 API 时，可以将环境参数放置于请求的任意位置，传递给后端服务。API 网关将适配您请求中的环境参数信息来区分请求环境。



环境变量参数配置方法

首先，在 API 分组的 **环境管理** 中，为每个环境创建变量。然后，在 **API 定义** 中配置已创建的环境变量。

创建环境变量

要实现通过环境变量参数区分请求环境，您需为测试、预发、线上三个环境，分别新增一个变量。

目前，每个环境允许配置最多 50 个环境变量。

1. 登录 API 网关控制台。
2. 单击 **分组管理** > **环境管理**。

3. 选择要增加变量的环境：线上、预发、或测试，再单击 **新增变量**。您需为不同的环境逐个新增一个变量。

   填写变量名称和值，再单击 **新增**。

Name：自行定义变量名称，但需 **保持三个环境中的对应的变量名称相同。**

如果您有多个 API，建议 Name 标识有实际意义，以便后续查询。

Value：变量的值。

如果以函数计算为 API 网关的后端服务，Value 值请填写您在函数计算服务中创建的服务名称或者函数名称。您需填写正确的服务名称或者函数名称，否则可能造成无法调用 API。以函数计算服务为例。一个函数服务，测试、预发、线上环境的名称分别为：TestServiceD、PreServiceD、ServiceD。（函数计算中环境变量设置，请参见函数计算文档 **环境变量。**）为 API 分组的测试、预发、线上环境分别定义一个的变量。变量可命名为 Service，并填写相应的服务名称作为 Value。

您还可以以函数名称录入名为 Function 的环境变量，并为三个环境分别设置变量。

在 API 定义里配置环境变量

API 定义时，在 **请求Path、入参定义、定义 API 后端服务**等部分加入变量。

表示方法：#变量名#。如，#Service#、#Function#。如，在定义以函数计算作为 API 网关后端服务时，将服务名称和函数名称定义为已创建的变量。

调用多环境 API

API 发布后，便可发起 API 调用。

线上环境调用

直接发起 API 调用，即调用线上环境。

预发环境调用

调用预发环境的 API， 则在调用 API 时，在 Header 中增加入参 X-Ca-Stage: PRE， 即可访问预发环境的 API。

测试环境调用

调用测试环境的 API，则在调用 API 时，在 Header 中增加入参 X-Ca-Stage: TEST， 即可访问测试环境的 API。

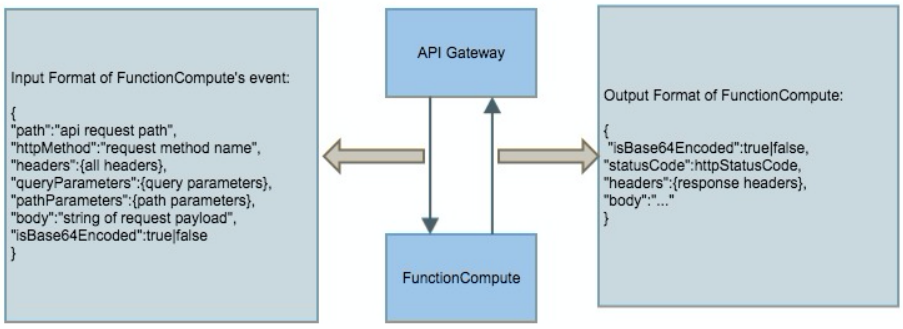
以函数计算作为 API 网关后端服务

函数计算是一个事件驱动的服务。函数的执行可以由事件驱动，即当某个事件发生时，该事件触发函数的执行。现在，函数计算支持以 API 网关作为事件源。当请求设置函数计算为后端服务的 API 时，API 网关会触发相应的函数，函数计算会将执行结果返回给 API 网关。

API 网关与函数计算对接，可以让您以 API 形式安全地对外开放您的函数，并且解决认证、流量控制、数据转换等问题（查看 API 网关功能）。

实现原理

API 网关调用函数计算服务时，会将 API 的相关数据转换为 Map 形式传给函数计算服务。函数计算服务处理后，按照下图中 Output Format 的格式返回 statusCode、headers、body 等相关数据。API 网关再将函数计算返回的内容映射到 statusCode、header、body 等位置返回给客户端。



API 网关向函数计算传入参数格式

当以函数计算作为 API 网关的后端服务时，API 网关会把请求参数通过一个固定的 Map 结构传给函数计算的入参 event。函数计算通过如下结构去获取需要的参数，然后进行处理。

```
{
"path":"api request path",
"httpMethod":"request method name",
"headers":{all headers,including system headers},
"queryParameters":{query parameters},
"pathParameters":{path parameters},
"body":"string of request payload",
"isBase64Encoded":"true|false, indicate if the body is Base64-encode"
}
```

注意：

- 如果 "isBase64Encoded" 的值为 "true"，表示 API 网关传给函数计算的 body 内容已进行 Base64 编码。函数计算需要先对 body 内容进行 Base64 解码后再处理。
- 如果"isBase64Encoded" 的值为 "false"，表示 API 网关没有对 body 内容进行 Base64 编码。

函数计算的返回参数格式

函数计算需要将输出内容通过如下 JSON 格式返回给 API 网关，以便 API 网关解析。

```
{
"isBase64Encoded":true|false,
"statusCode":httpStatusCode,
"headers":{response headers},
"body":"..."
}
```

注意：

- 当 body 内容为二进制编码时，需在函数计算中对 body 内容进行 Base64 编码，设置 "isBase64Encoded" 的值为 "true"。如果 body 内容无需 Base64 编码，"isBase64Encoded" 的值为 "false"。API 网关会对 "isBase64Encoded" 的值为 "true" 的 body 内容进行 Base64 解码后,再返回给客户端。
- 在 Node.js 环境中，函数计算根据不同的情况设置 callback。
  - 返回成功请求： callback{null,{ “statusCode” :200,” body” :” …” }}。
  - 返回异常：callback{new Error(‘internal server error’),null}。
  - 返回客户端错误： callback{null,{ “statusCode” :400,” body” :” param error” }}。
- 如果函数计算返回不符合格式要求的返回结果，API 网关将返回 503 Service Unavailable 给客户端。

配置 API 网关触发函数计算

配置 API 网关触发函数计算服务的操作步骤：

1. 在函数计算控制台创建函数
2. 创建并定义以函数计算为后端服务的 API
3. 调试 API
4. 将 API 发布到线上

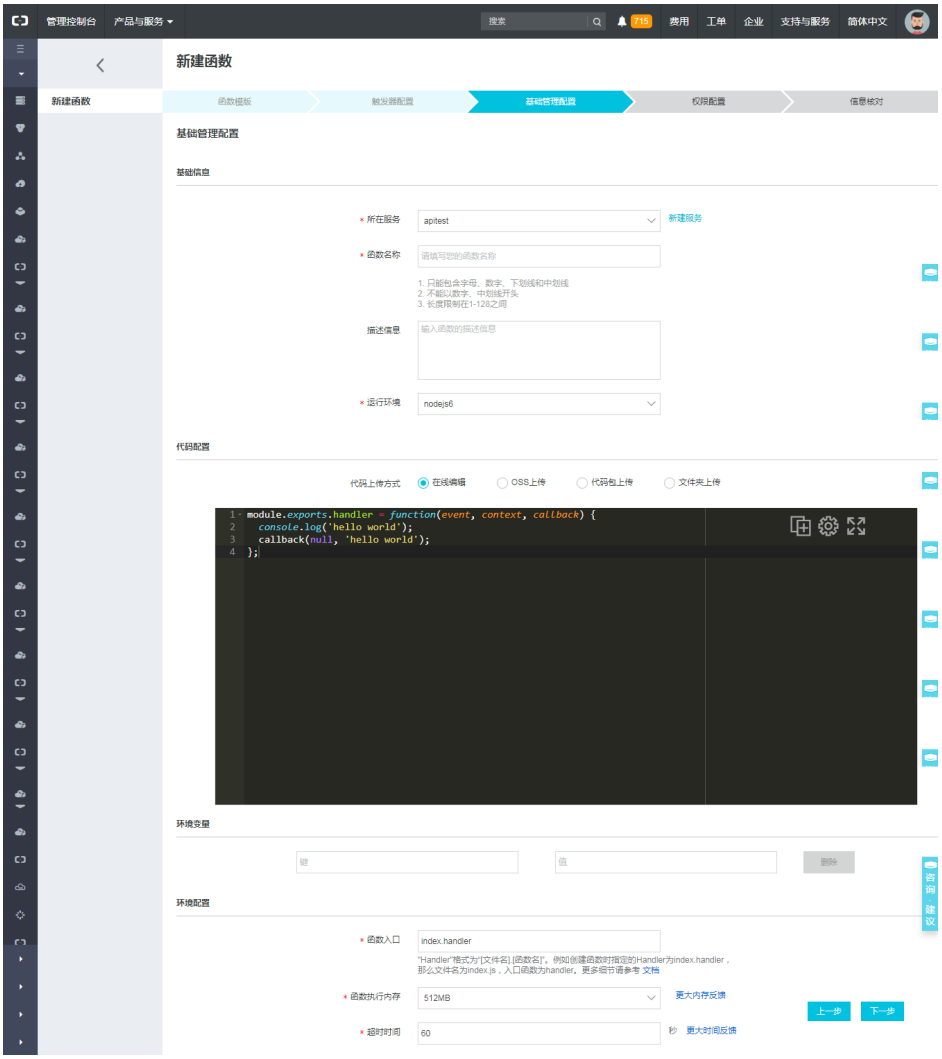
在函数计算控制台创建函数

创建服务。登录 **函数计算控制台**， 选择您要创建服务和函数的 **所属区域**，单击 **新建服务**，并在弹出对话框中完成服务创建。

注意：服务创建成功后，无法更换区域，请谨慎选择 **所属区域**。

2. 在已创建的服务中，创建函数。在该服务页面上，单击 **新建函数** 进入函数创建流程：
   i. 选择函数模板。函数计算控制台中，提供了 Node.js 6 环境的 API 网关后端实现模板 api-gateway-nodejs6 供您使用。如果 api-gateway-nodejs6 模板不适用您的业务场景，请选择 **空白函数**。选择使用 **空白函数** 模板后，在 **基础管理配置** 中需提交您自己编写的代码。请提前准备好代码包，以便上传。

   ii. 触发器配置。**触发器类型** 选择为 **不创建触发器**，单击 **下一步**。
   iii. 基础管理配置：填写基础信息、配置代码、设置环境变量、和配置环境，再单击 **下一步**。关于代码编写示例，可参见 API网关触发函数计算 文档中，**编写函数代码** 部分。



iv. 请忽略权限配置，直接单击 **下一步**。因为我们已在 RAM 控制台配置了相应的 roleArn 的权限，所以无需您手动配置。 您只需在 API 网关控制台创建 API 时，单击 **获取授权** ，即可自动获取授权。

v. 核对信息，信息无误，则单击 **创建**。成功创建函数后，您可以在 **函数列表** 中，查看所创建函数的基本信息。

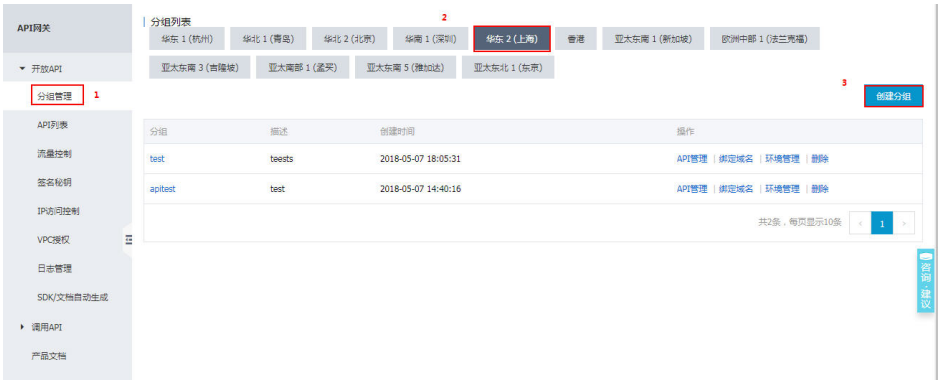
创建并定义以函数计算为后端服务的 API

您需在 API 网关控制台 创建 API， 并定义此 API 的后端服务为函数计算。

1. 登录 API 网关控制台。

   单击控制台左侧导航栏中 **分组管理**，选择分组列表的地域，再单击 **创建分组**。（如果已创建分组，请忽略此步骤。）

   注意：如果函数计算与 API 不在同一地域，将通过公网访问您的函数计算服务。若您对数据安全和网络延迟有较高要求，请选择API 与函数计算为同一地域。



API 分组创建成功后，您可以通过 **环境管理** 为此分组设置环境变量。目前有三种环境：测试、预发、和线上。为避免环境转换导致后端地址变化，您可以通过增加环境变量参数来实现请求的自动路由。环境变量配置方法，请参见 **环境管理**。

创建和定义 API。

i. 创建分组成功后，单击该分组操作栏中 **API 管理** 按钮，进入相应的 **API 列表** 页面。
ii. 单击 **创建 API**, 进入 API 创建和定义流程。

填写基本信息，再单击 **下一步**。

注意：如果选择类型为 **私有**，该 API 将不能在云市场上架。

定义 API 请求，再单击 **下一步**。

注意：如果 **入参请求模式** 选择为 **入参透传**，则发送给 API 网关的参数 body 内容不经处理，直接作为参数透传给函数计算。

定义 API 后端服务，再单击 **下一步**。

注意：在此页面，您需：

i. 选择 **后端服务类型** 为 **函数计算**。
ii. 填写 **服务名称** 为您在 **函数计算控制台** 创建的服务名称。
iii. 填写 **函数名称** 为您在 **函数计算控制台** 创建的函数名称。
iv. 单击 **获取授权**，自动获取角色 Arn。如果这是您第一次获取函数计算为 API 网关后端服务的角色授权，当您单击 **获取授权** 后，会弹出 RAM 控制台的授权页面。您需单击 RAM 控制台的授权权限，然后返回 API 创建页面再次单击 **获取授权**，该角色 Arn将自动显示在选项框中。

定义返回结果，然后单击 **创建**。

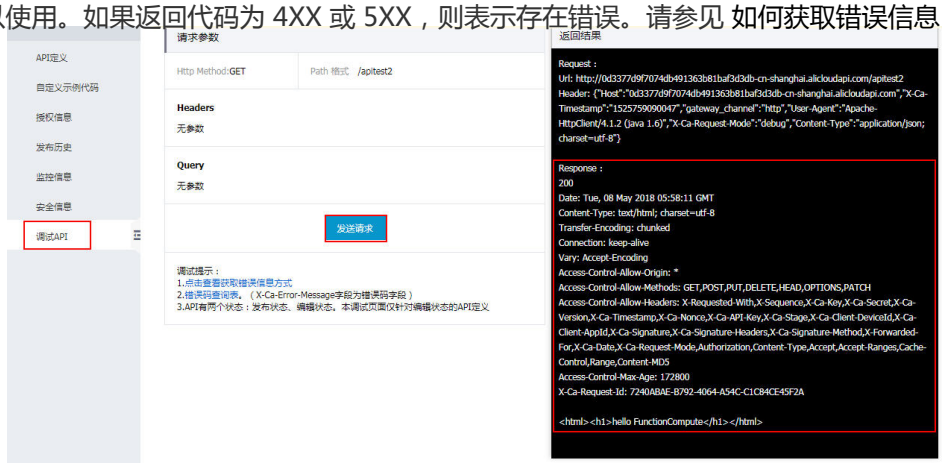
注意：返回结果示例为必填，且格式需遵循 **函数计算的返回参数格式** 。

如需更多帮助，请参见 API 创建。

调试 API

API 创建、定义完成后，页面自动跳转到 **API 列表** 页。您可以通过此页面按钮，对创建的 API 进行测试是否可用，请求链路是否正确。

1. 单击 API 名称或 **管理** 按钮，进入 **API 定义** 页面。
2. 单击左侧导航栏中 **调试 API**。
3. 输入请求参数，单击 **发送请求**。返回结果将显示在右侧页面。如果调试返回成功结果，则说明该 API 可以使用。如果返回代码为 4XX 或 5XX，则表示存在错误。请参见 **如何获取错误信息** 和 **错误代码表**。



4. 将 API 发布到 **预发** 环境，做上线前测试。测试证明 API 可用后，可返回 **API 定义** 页面，将 API 发布到 **预发** 环境。然后，通过访问二级域名来进行测试调用，模拟真实的用户请求。

   注意： 如果在 API 定义中设置了环境变量，需在请求 Header 中增加入参 X-Ca-Stage: RELEASE 才能正确调用预发环境的 API。

将 API 发布到线上

API 通过调试，证明可以使用后，可将 API 进行发布。

1. 在 **API 列表** 页面，单击 API 名称或 **管理** 按钮，进入 **API 定义** 页面。

2. 单击页面右上方 **发布** 按钮，弹出 **发布 API** 对话框。
3. 选择要发布的环境为**线上**，填写备注信息，单击 **发布**。将 API 发布到线上后，您的用户便可以调用此 API。

更多发布相关细节，请参见文档 发布 API。

示例

以下提供三个示例，分别为：函数代码示例、API 请求示例、和 API 网关返回示例。

函数代码示例

在函数计算中配置的代码示例。

```
module.exports.handler = function(event, context, callback) {
var responseCode = 200;
console.log("request: " + JSON.stringify(event.toString()));
//将event转化为JSON对象
event=JSON.parse(event.toString());
var isBase64Encoded=false;
//根据用户输入的statusCode返回，可用于测试不同statusCode的情况
if (event.queryParameters !== null && event.queryParameters !== undefined) {
if (event.queryParameters.httpStatus !== undefined && event.queryParameters.httpStatus !== null &&
event.queryParameters.httpStatus !== "") {
console.log("Received http status: " + event.queryParameters.httpStatus);
responseCode = event.queryParameters.httpStatus;
}
}
//如果body是Base64编码的，FC中需要对body内容进行解码
if(event.body!==null&&event.body!==undefined){
if(event.isBase64Encoded!==null&&event.isBase64Encoded!==undefined&&event.isBase64Encoded){
event.body=new Buffer(event.body,'base64').toString();
}
}
//input是API网关给FC的输入内容
var responseBody = {
message: "Hello World!",
input: event
};
```

```
//对body内容进行Base64编码，可根据需要处理
var base64EncodeStr=new Buffer(JSON.stringify(responseBody)).toString('base64');

//FC给API网关返回的格式，须如下所示。isBase64Encoded根据body是否Base64编码情况设置
var response = {
isBase64Encoded:true,
statusCode: responseCode,
headers: {
"x-custom-header" : "header value"
},
body: base64EncodeStr
};
console.log("response: " + JSON.stringify(response));
callback(null, response);
};
```

请求示例

以 POST 形式请求 path 为如下的 API：

```
/fc/test/invoke/[type]


POST http://test.alicloudapi.com/fc/test/invoke/test?param1=aaa&param2=bbb

"X-Ca-Signature-Headers":"X-Ca-Timestamp,X-Ca-Version,X-Ca-Key,X-Ca-Stage",
"X-Ca-Signature":"TnoBIdxxRHrFferGlzzkGcQsaezK+ZzySloKqCOsv2U=",
"X-Ca-Stage":"RELEASE",
"X-Ca-Timestamp":"1496652763510",
"Content-Type":"application/x-www-form-urlencoded; charset=utf-8",
"X-Ca-Version":"1",
"User-Agent":"Apache-HttpClient\/4.1.2 (java 1.6)",
"Host":"test.alicloudapi.com",
"X-Ca-Key":"testKey",
"Date":"Mon, 05 Jun 2017 08:52:43 GMT","Accept":"application/json",
"headerParam":"testHeader"

{"bodyParam":"testBody"}
```

API 网关返回示例

```
200
Date: Mon, 05 Jun 2017 08:52:43 GMT
Content-Type: application/json; charset=UTF-8
Content-Length: 429
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET,POST,PUT,DELETE,HEAD,OPTIONS，PATCH
Access-Control-Allow-Headers: X-Requested-With, X-Sequence,X-Ca-Key,X-Ca-Secret,X-Ca-Version,X-Ca-
Timestamp,X-Ca-Nonce,X-Ca-API-Key,X-Ca-Stage,X-Ca-Client-DeviceId,X-Ca-Client-AppId,X-Ca-Signature,X-Ca-
Signature-Headers,X-Forwarded-For,X-Ca-Date,X-Ca-Request-Mode,Authorization,Content-Type,Accept,Accept-
Ranges,Cache-Control,Range,Content-MD5
```

```
Access-Control-Max-Age: 172800
X-Ca-Request-Id: 16E9D4B5-3A1C-445A-BEF1-4AD8E31434EC
x-custom-header: header value

{"message":"Hello World!","input":{"body":"{\"bodyParam\":\"testBody\"}","headers":{"X-Ca-Api-Gateway":"16E9D4B5-3A1C-445A-BEF1-4AD8E31434EC","headerParam":"testHeader","X-Forwarded-For":"100.81.146.152","Content-Type":"application/x-www-form-urlencoded; charset=UTF-8"},"httpMethod":"POST","isBase64Encoded":false,"path":"/fc/test/invoke/test","pathParameters":{"type":"test"},"queryParameters":{"param1":"aaa","param2":"bbb"}}}
```

常见问题

为什么我无法录入我已有的函数？

请确认您输入的函数计算的服务名称和函数名称是否与您在函数计算控制台创建的服务和函数的名称一致。

我是否可以将多个函数作为一个 API 的后端服务？

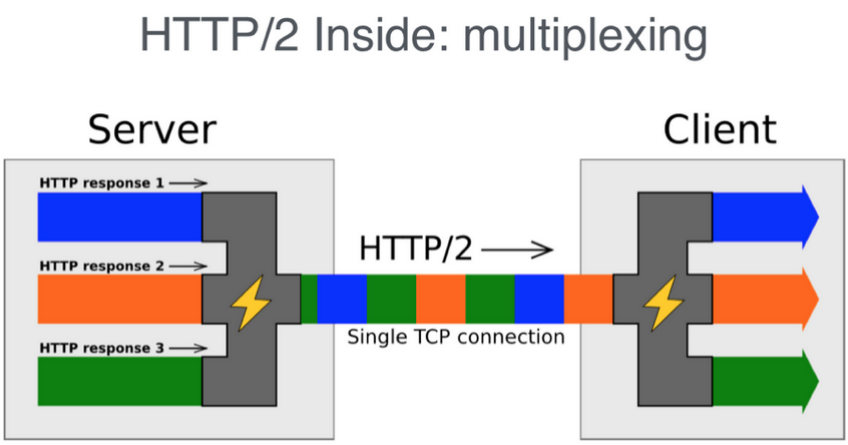
不可以，目前 API 和函数是一对一的关系存在。

支持HTTP2.0

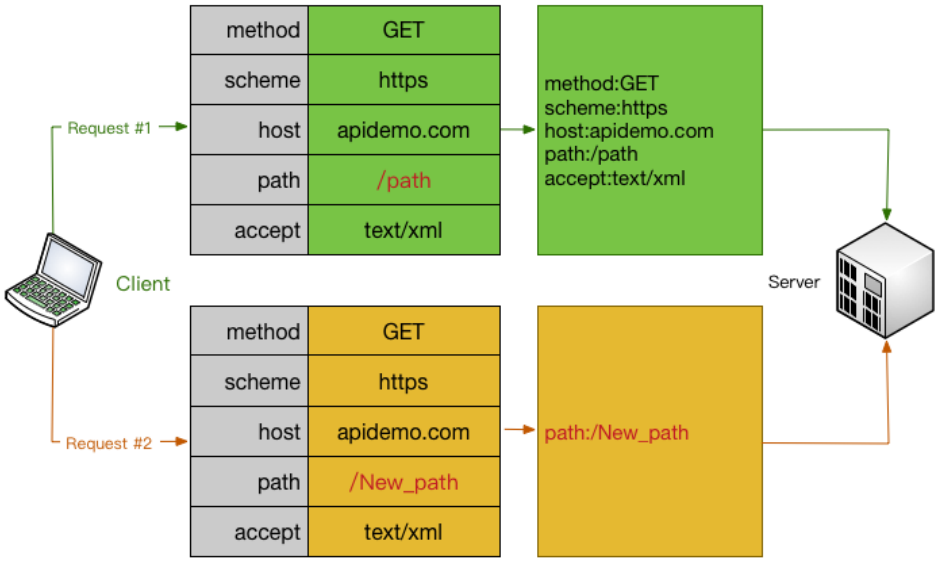
API网关支持HTTP2.0

API网关支持HTTP2.0新特性，支持API请求多路复用、支持请求头压缩。

- 多路复用（MultiPlexing）：消除了 HTTP 1.x 中并行处理和发送请求及响应时对多个连接的依赖。可客户端和服务器可以把HTTP消息分解为互不依赖的帧，然后乱序发送，最后再在另一端把它们重新组合起来。从而避免不必要的延迟，提升效率，在请求量比较大的场景，客户端也可以轻松使用少量连接完成大量请求数据的传输。



- header压缩：如上文中所言，HTTP1.x 的header带有大量信息，而且每次都要重复发送。HTTP 2.0 使在客户端和服务器端使用“首部表”来跟踪和存储之前发送的键值对，对于相同的数据，不再通过每次请求和响应发送；“首部表”在 HTTP 2.0 的连接存续期内始终存在，由客户端和服务器共同渐进地更新；每个新的首部键值对要么追加到当前表的末尾，要么替换表中之前的值。从而减少每次请求的数据量。



如何开启HTTP 2.0？

新建的API分组（2017年7月14日以后）

HTTPS的API都可以使用HTTP2协议进行客户端和API网关的通信。（由于 HTTP 2.0 只许在HTTPS下运行，所以需要您开启 HTTPS方可启用 HTTP 2.0）

存量API分组

您需稍做等待，后续将提供功能手动开启。

支持 HTTPS

HTTPS 在 HTTP 的基础上加入了 SSL 协议，对信息、数据加密，用来保证数据传输的安全。现如今被广泛使用。

API网关也支持使用HTTPS对您的API请求进行加密。可以控制到 API 级别，即您可以强制您的 API 只支持 HTTP、HTTPS 或者两者均支持。

如果您的API需要支持HTTPS，以下为操作流程：

步骤1:准备

您需要准备如下材料：

- 一个自有可控域名。
- 为这个域名申请一个 SSL 证书。

自定义上传证书，包含证书/私钥，均为 PEM 格式， 证书格式说明 (注：API网关Tengine服务是基于Nginx，因此只支持Nginx能读取的证书，即PEM格式)。

SSL 证书会包含两部分内容:XXXXX.key、XXXXX.pem，可以使用文本编辑器打开。

示例：

KEY

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEA8GjIleJ7rlo86mtbwcDnUfqzTQAm4b3zZEo1aKsfAuwcvCud
....
-----END RSA PRIVATE KEY-----
```

PEM

```
-----BEGIN CERTIFICATE-----
MIIFtDCCBJygAwIBAgIQRgWF1j00cozRl1pZ+ultKTANBgkqhkiG9w0BAQsFADBP
...
-----END CERTIFICATE-----
```

步骤2:绑定SSL证书

准备好以上材料，需要进行如下操作进行，登陆 API 网关管理控制台【开放API】-【分组管理】，单击您需要

绑定 SSL 证书的分组，查看分组详情。

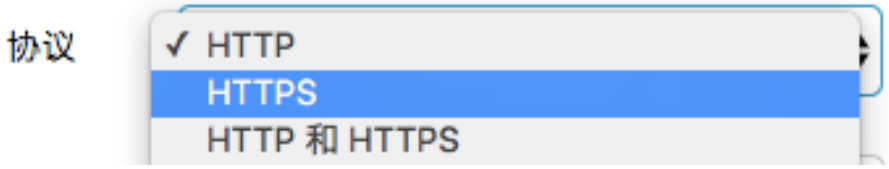
在绑定 SSL 证书，您首先需要在 API 分组上绑定【独立域名】。

【独立域名】-添加 SSL 证书。

- 证书名称：用户自定义名称，以供后续识别。
- 证书内容：证书的完整内容，需要复制 XXXXX.pem 中的全部内容。
- 私钥：证书的私钥，需要复制 XXXXX.key 中的内容。点击【确定】后，完成 SSL 证书的绑定。

步骤3：API配置调整

绑定 SSL 证书后，您可以按 API 控制不同的访问方式，支持 HTTP、HTTPS、HTTP 和 HTTPS 三种，出于安全考虑，建议全部配置成 HTTPS。



可以在【开放API】-【API列表】找到相应 API，【API定义】-编辑-【请求基础定义】中进行修改。

API支持的协议包括：

HTTP：只允许 HTTP 访问，不允许 HTTPS。

HTTPS：只允许 HTTPS 访问，不允许 HTTP。

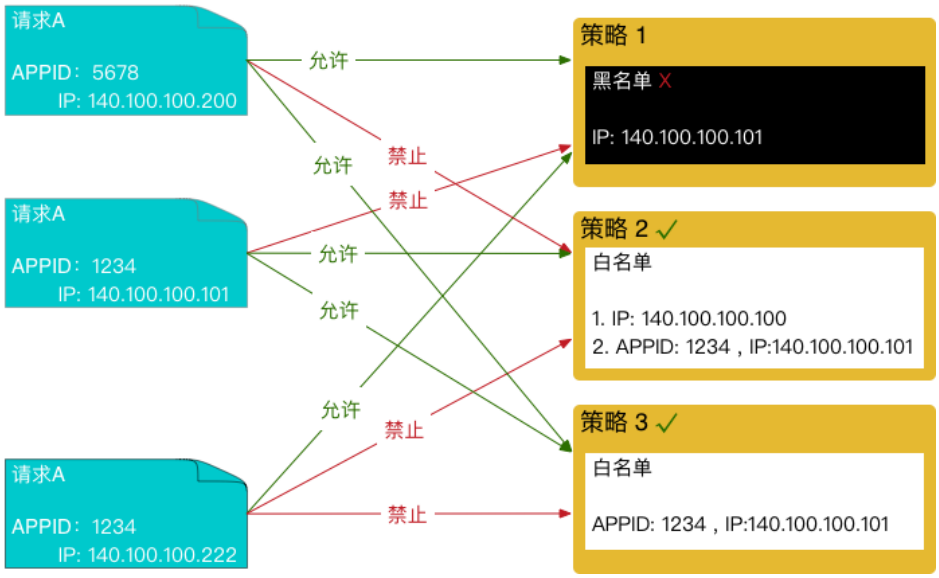
HTTP 和 HTTPS：两者均可。

调整后，API 支持 HTTPS 协议配置完成。您的 API 将支持 HTTPS 访问。

IP访问控制

IP 访问控制是 API 网关提供的 API 安全防护组件之一，负责控制 API 的调用来源 IP （支持IP段）。您可以通过配置某个 API 的 IP 白名单/黑名单来允许/拒绝某个来源的API请求。

- 白名单，支持配置 IP 或者 APPID + IP 的白名单访问，不在白名单列表的请求将会被拒绝。
    - IP 白名单，只允许设定的调用来源 IP 的请求被允许。
    - APPID + IP 此APPID只能在设定的 IP 下访问，其他 IP 来源将被拒绝。
- 黑名单，您可以配置 IP 黑名单，黑名单中 IP 的访问将被 API 网关拒绝。



使用方法

添加流程

您需要先创建IP访问控制策略，并绑定到需要控制的API上即可完成设置。

创建IP访问控制

打开API网关控制台-【开放API】-【IP访问控制】

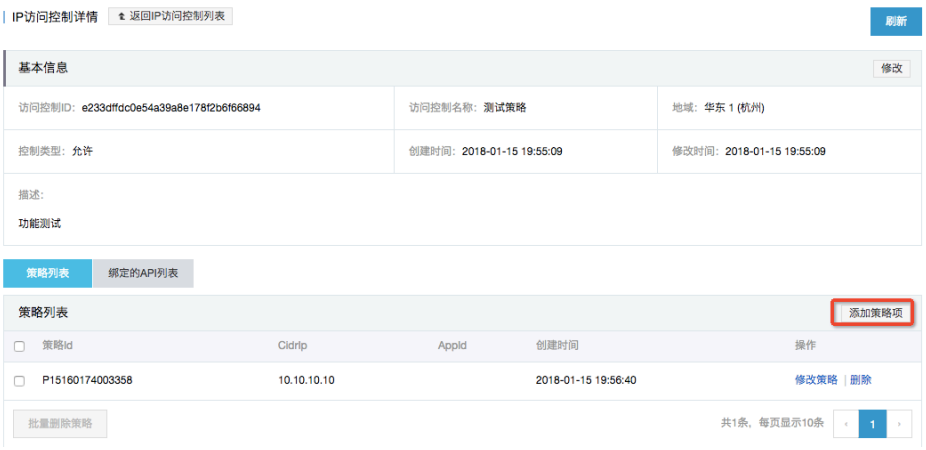
点击“创建访问控制”，弹出IP访问控制创建窗口：

按要求填写相应信息，并确定后完成创建。

- 允许：白名单
- 拒绝：黑名单

添加策略

创建后需要填写相应控制策略，白名单可以填写APP、IP 或者APPP+ IP，黑名单，可以填写 IP



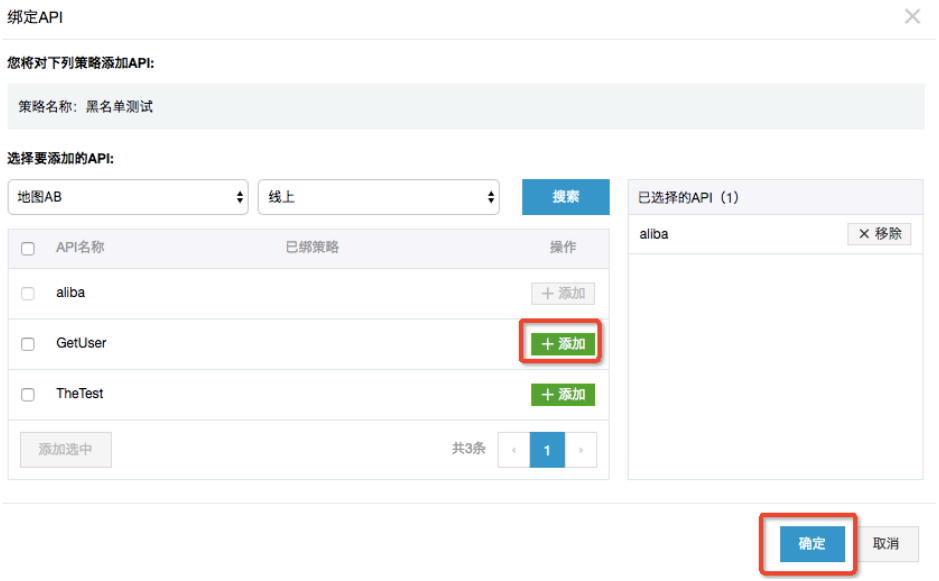
确定后完成策略添加。

绑定API

IP 访问控制需要绑定到API之后才能真正发挥作用。

在 IP 访问控制列表页面：

找到您所需要的策略，点击“绑定API”：



按要求选择相应API即可完成绑定。

注：无论是黑名单还是白名单，在一个API只能绑定一条访问控制。

删除IP访问策略

您只需要在 IP 访问控制列表页面，删除相应策略即可。

注：已绑定API的访问控制，需要先解绑API再做删除。

查询绑定的API

您可以在 IP 访问控制详情页面，查询此访问控制的策略即可。

FAQ

1. 绑定/删除 IP 访问控制策略何时生效？API网关采用实时策略，绑定即生效，无延迟。
2. 一个API不同环境是否可以绑定不同的 IP 访问控制？是的，您可以针对不同环境绑定相应 IP 访问控制。建议您的测试环境、预发环境、绑定您的指定 IP ，来保护您测试环境的安全。
3. 为什么不设计 APP 黑名单？API调用需要进行APP授权，若想禁止某个APP调用，删除其授权即可，无需配置黑名单。

使用简单认证（AppCode）方式调用API

1.概述

阿里云API网关提供多种针对客户端请求的安全认证方式，包括阿里云APP认证方式、OpenID Connect等。对于”阿里云APP“这种认证方式，目前用户可以设置两种认证形式：

1. 签名认证；
2. 简单认证。

对于请求的签名认证方式，可以参考这个文档：请求签名说明文档：
https://help.aliyun.com/document_detail/29475.html

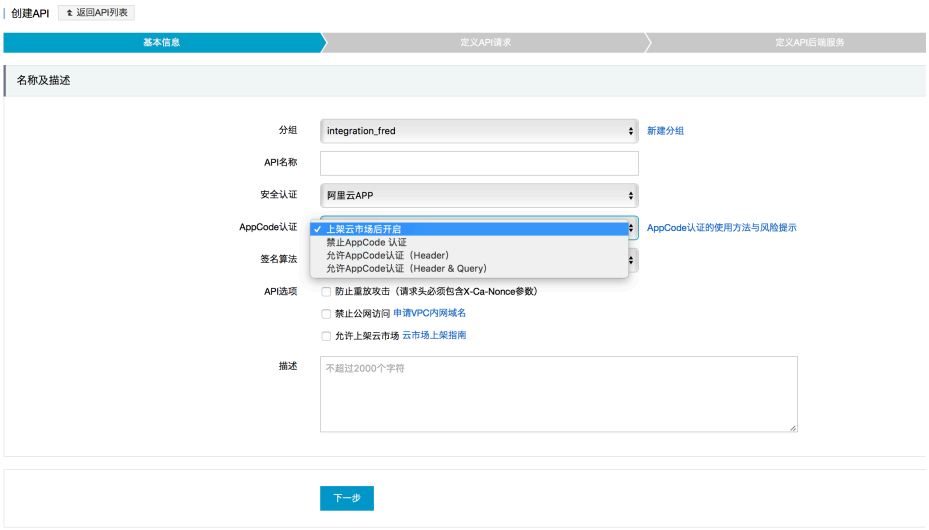
本文将详细描述简单认证方式的设置方式和调用方式。简单认证，顾名思义，和签名认证方式相比要简单很多，省去了复杂的生成签名的过程。**简单认证方式直接使用API网关颁发的AppCode进行身份认证，调用者将AppCode放到请求头中，或者放到请求的Query参数中进行身份认证，实现快速调用API的能力。**下面是对整个流程的描述：

1. API提供者创建API的时候选择”阿里云APP“认证模式，且支持AppCode认证（所有云市场的API默认都支持AppCode）；

2. API调用者在API网关”应用管理“创建一个APP。云市场用户在云市场购买API时云市场会为您创建一个 APP；
3. API提供者给调用者的APP进行API授权，具体授权方式请参考文档：https://help.aliyun.com/document_detail/29497.html
4. API调用者到API网关控制台的”应用管理“找到AppCode/AppSecret进行签名认证的调用或者AppCode进行简单认证的API调用。

2. 创建支持简单认证方式API

1. API提供者在创建API安全认证方式时需要选择”阿里云APP”，或者选择”OpenId Connect & 阿里云APP“；
2. AppCode认证选项选择允许AppCode认证相关的几个选项；



下面解释下AppCode认证四个选项的含义进行详细解释：

- 上架云市场后开启：默认不开启，如果上架API上架云市场，则支持将AppCode放在Header中进行认证；
- 禁止AppCode认证：无论API是否上架云市场，都不开启，都需要使用签名方式调用；
- 允许AppCode认证（Header）：无论API是否上架云市场，都开启，但只支持将AppCode放在Header中进行认证；
- 允许AppCode认证（Header & Query）：无论API是否上架云市场，都开启，同时支持将AppCode放在Header中，或者将AppCode放在Query中进行认证；

注意，定义API参数的时候，不需要定义携带AppCode的头或者Header参数

3. 调用方法

API提供者将API设置成允许AppCode调用之后，API调用者就可以使用简单认证方式进行调用了，不用再在客

户端实现复杂的签名算法了。本章我们介绍下如何通过简单认证方式调用API。主要有两种方式，一种是将AppCode放在Header中进行调用，一种是将AppCode放在Query参数中进行调用。

3.1 将AppCode放在Header中

- 请求Header中添加一个”Authorization“参数；
- Authorization字段的值的格式为“APPCODE + 半角空格 + APPCODE值”。

格式：

```
Authorization:APPCODE AppCode值
```

示例：

```
Authorization:APPCODE 3F2504E04F8911D39A0C0305E82C3301
```

3.2 将AppCode放在Query中

- 请求Query中添加的”AppCode“参数（同时支持”appcode“，“appCode”，“APPCODE”，“APPCode”四种写法）；
- Authorization字段的值AppCode的值。

示例：

```
http://www.aliyum.com?AppCode=3F2504E04F8911D39A0C0305E82C3301
```

4.风险提示

简单认证方式调用非常省事，免去了复杂的签名过程，但是把AppCode作为明文暴露网络中传输，会带来一些安全隐患，务必重视：

客户端和API网关之间务必使用HTTPS进行通信，避免使用HTTP/WebSocket协议进行数据传输。因为简单认证方式，AppCode在传输过程中使用明文，而HTTP/WebSocket通信协议没有加密，一旦网络通信的网络包被黑客抓取，有非常大的丢失AppCode的风险。