ALIBABA CLOUD

阿里云

应用身份服务
开发指南

文档版本：20220224

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}`

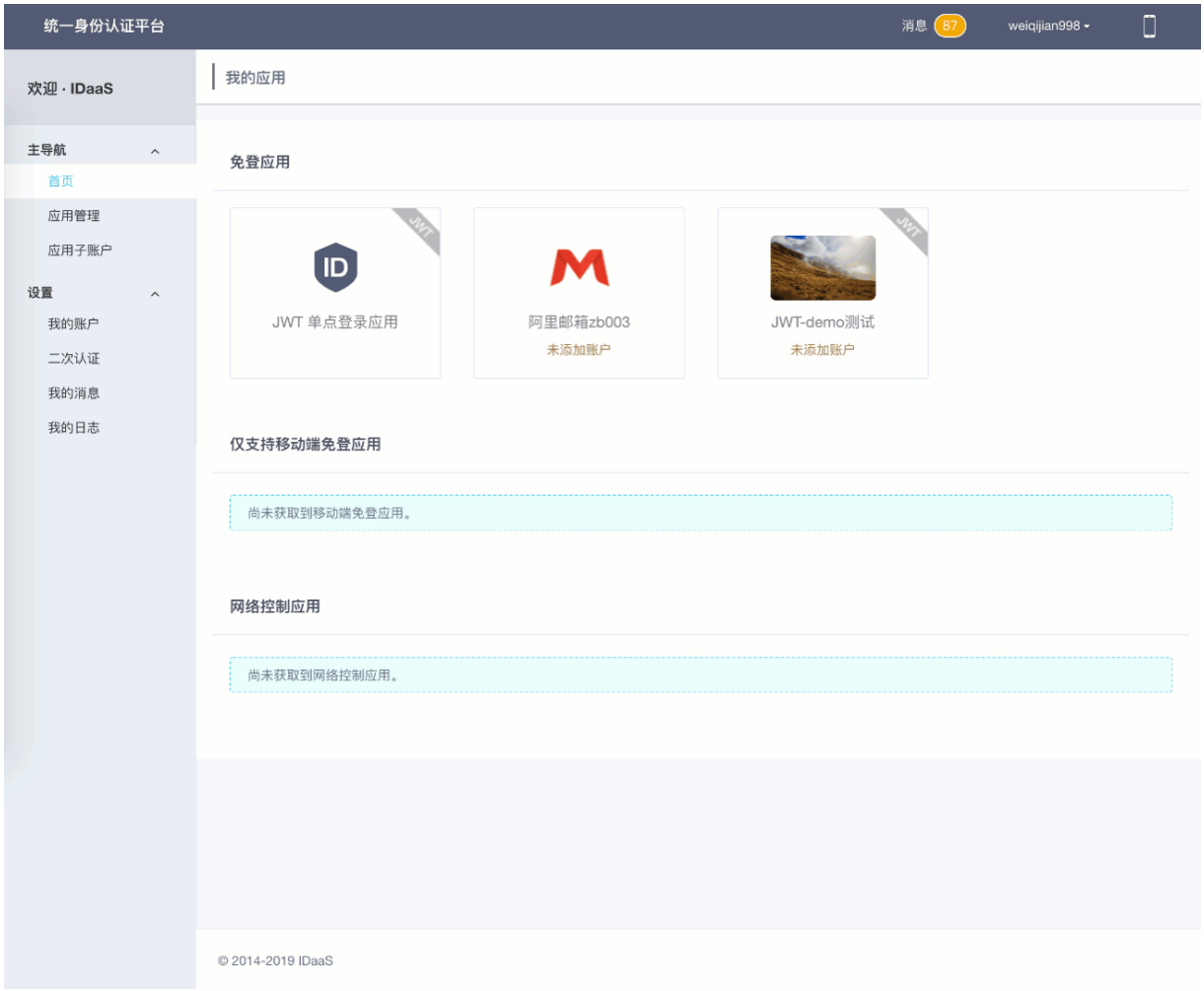
目录

1.单点登录SSO

1.1. 单点登录概述

单点登录（SSO）概述 v1.2

单点登录（SSO），英文全称为 Single Sign On。 SSO 是指在多个应用系统中，用户只需要登录一次，就可以访问所有相互信任的应用系统。IDaaS SSO 服务用于解决同一公司不同业务应用之间的身份认证问题，只需要登录一次，即可访问所有添加的应用。此服务可以涵盖用户在公有云和私有云中的双重需求。



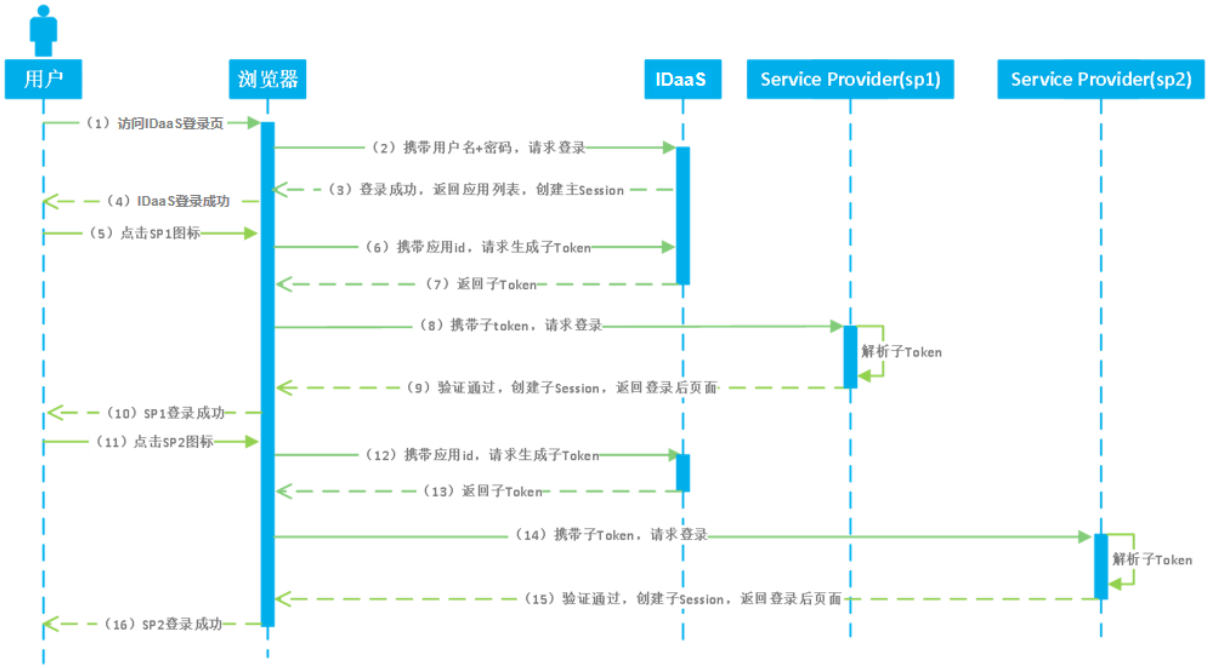
单点登录场景

在单点登录实现过程中，现已满足以下登录场景，包括：

1. IDaaS发起
2. SP发起
3. 接口后置
4. 登录跳转

IDaaS发起

用户登录 IDaaS 平台，从 IDaaS 登录到 SP 应用场景，如下图所示：



（1）用户访问 IDaaS 登录页面输入用户名和密码进行登录

（2）浏览器携带用户名密码向 IDaaS 请求登录

（3）IDaaS 认证通过后，创建主 session，并返回应用列表给浏览器

（4）IDaaS 登录成功，用户可以看到 IDaaS 展示的应用列表

（5）用户点击应用列表中的 SP1 应用图标

（6）浏览器携带 SP1 应用的应用 id，向 IDaaS 请求生成 SP1 应用子 token

（7）IDaaS 根据信息生成子 token 并返回给浏览器

（8）浏览器携带子 token，向 SP1 请求登录

（9）SP 1 应用系统解析获取的子 token，验证通过后，创建子 session 并返回登录后页面

（10）SP 1 系统单点登录成功，浏览器显示 SP1 系统登录后页面

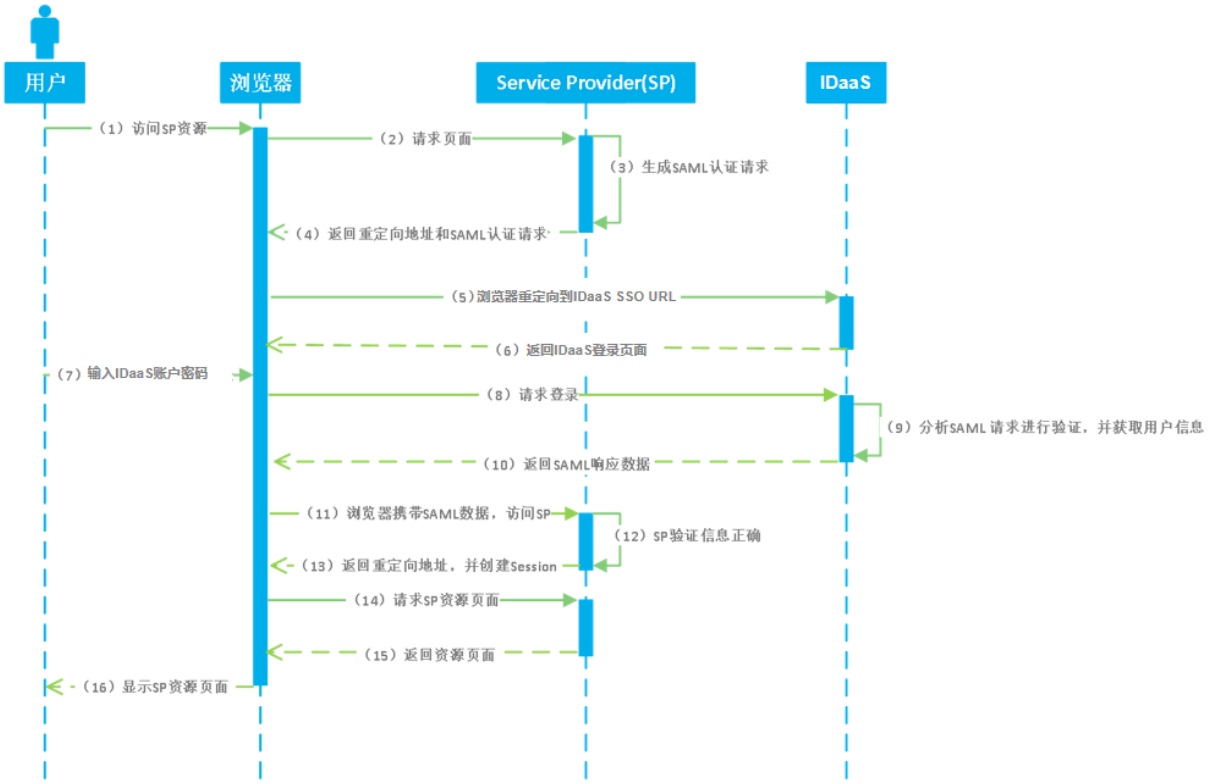
（11）~（16）表示 SP2 系统的单点登录流程，与 SP1 系统步骤一致，即在主 session 创建后，任何一个可单点登录应用进行单点登录的流程只需重复（5）~（10）即可

SP发起

SP 发起主要应用于 SSO 后可以跳回发起 SSO 的应用页面。下面以 SAML 和 JWT 的实现为例来阐述 SP 发起的单点登录流程。

1、访问应用页面

用户到 SP 应用页面，会在 SP 通过 Redirect 或 POST 提交一个某种认证协议如 SAML / CAS 的挑战请求，在 IDaaS 登录后，再跳转回 SP 作为应答，实现统一认证，如下图所示：



以 SAML 协议 SP 发起的单点登录为例：

（1）用户访问 SP 资源页面

（2）浏览器向 SP 请求资源

（3）SP 生成 SAML AuthnRequest 请求，其中包含当前 URL 到 RelayState，并返回给浏览器

（4）浏览器携带 SAML AuthnRequest 请求，访问定义好的 IDaaS 中 SP 发起的 SSO URL

（5）IDaaS 验证 SAML AuthnRequest，若该用户已登录直接跳至步骤（9），否则继续步骤（6）

（6）重定向到 IDaaS 登录页面

（7）用户输入 IDaaS 的账户和密码

（8）浏览器携带账号密码请求登录 IDaaS

（9）登录 IDaaS 后， IDaaS 分析 SAML AuthnRequest 中的 SP 应用信息，并获取更多用户信息，然后组合生成响应包含 RelayState 的 SAML Response Token。

（10）返回 SAML Response Token 数据给浏览器

（11）浏览器进行跳转，携带 SAML Response Token，访问 SP ACS URL

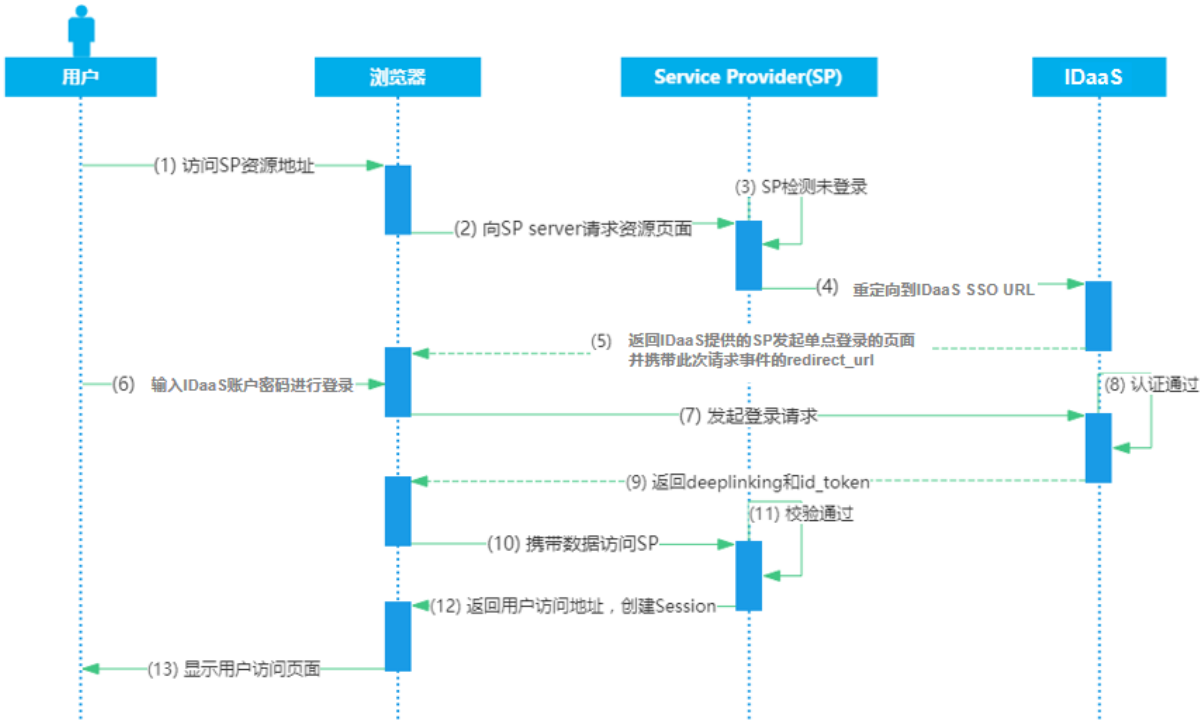
（12）SP 利用公钥验证 SAML Response Token

（13）校验成功后，创建 session 会话，从 RelayState 中取出开始时发起的 URL，返回给浏览器

（14）浏览器访问资源页面（15）SP 返回资源页面（16）用户登录进 SP 资源页面

2、访问 SP 资源

用户访问 SP 资源，SP 会重定向到 IDaaS 的 SSO 地址，在 IDaaS 认证通过后，会向 SP 返回 SP 发起登录的页面并携带请求事件redirect_url参数，用户登录后 SP会向浏览器返回并显示用户访问的资源页面，如下图所示：



以 JWT 协议 SP 发起的单点登录为例：

（1）用户访问 SP 资源页面

（2）浏览器向 SP 请求资源

（3）SP 检测登录，若该用户已登录直接跳至步骤（9），未登录继续到（4）

（4）SP 把当前页面地址置入 redirect_url 参数，并重定向到 SP SSO URL

（5）重定向到 IDaaS 登录页面

（6）用户输入 IDaaS 的账户和密码进行登录

（7）浏览器携带账号密码请求登录 IDaaS

（8）IDaaS 对账户进行认证并通过

（9）返回 JWT SSO URL，在 URL 中包含了 id_token 和 target_url

（10）浏览器携带上述参数跳转到 redirect_url

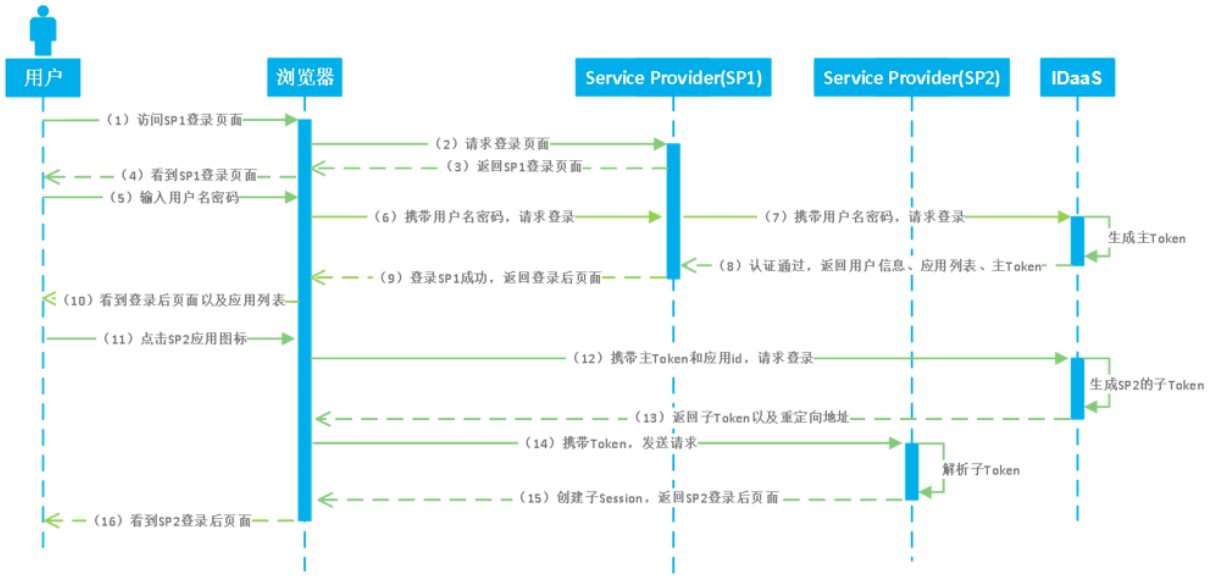
（11）SP 通过 PublicKey 对 id_token 进行校验

（12）校验成功后，创建 session 会话，向浏览器返回用户访问的 redirect_url 地址并进行跳转

（13）用户查看到之前访问的页面资源

接口后置

SP 登录时使用 IDaaS 进行接口后置认证登录场景，如下图所示：



（1）用户访问 SP1 登录页面

（2）浏览器请求 SP1 登录页面

（3）SP 1 返回其登录页面

（4）用户看到 SP1 登录页面

（5）用户输入用户名密码进行登录

（6）浏览器携带用户名、密码向 SP1 发起登录请求

（7）SP 1 使用 IDaaS 接口进行认证，将用户名、密码传递给 IDaaS 进行认证登录

（8）IDaaS 进行认证后，生成主 token ，并返回主 token 和应用列表以及用户信息给 SP1

（9）登录 SP1 成功，浏览器获取到 SP1 登录后页面

（10）用户可看到 SP1 登录后页面，可看到显示的应用列表

（11）用户在 SP1 显示的应用列表中点击 SP2 应用图标进行单点登录

（12）浏览器携带主 token 和应用id，向 IDaaS 请求生成子 token

（13）IDaaS 返回 SP2 的子 token 以及 SP2 重定向地址

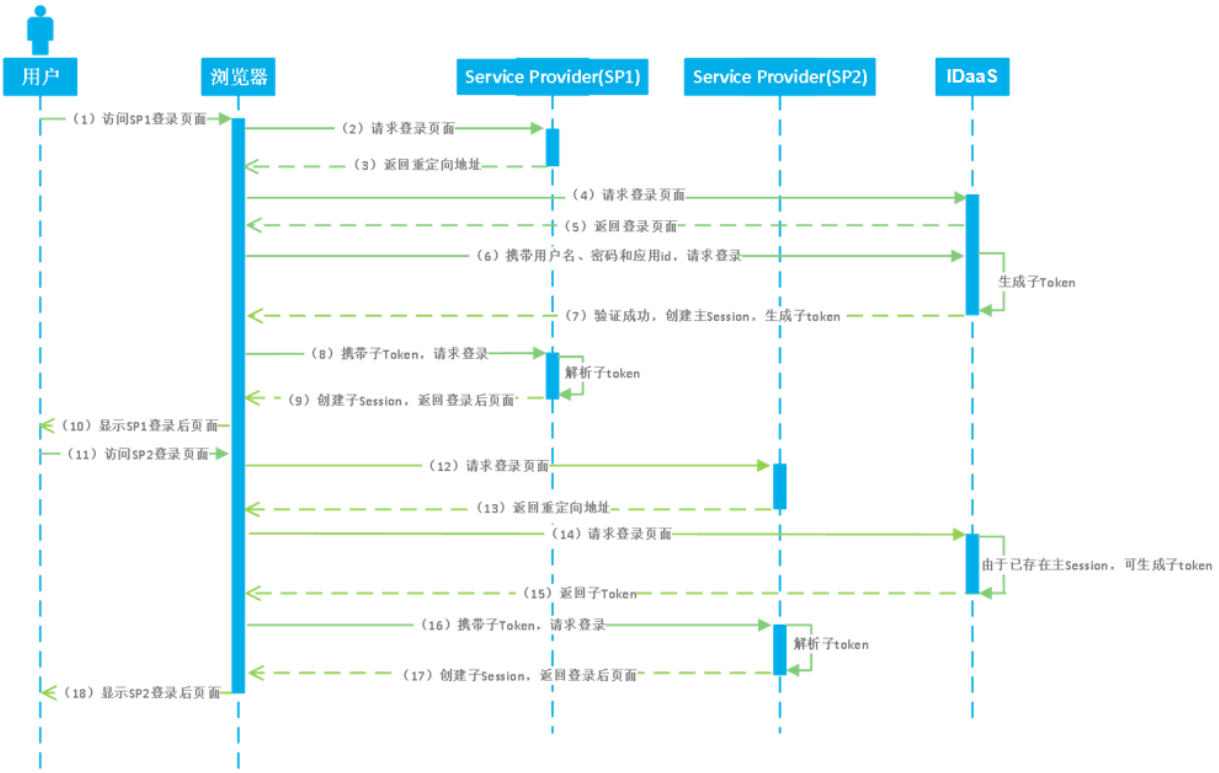
（14）浏览器携带子 token 访问 SP2 重定向地址

（15）SP2 解析子 token ，验证成功，并返回 SP2 登录后页面

（16）用户看到 SP2 登录后页面， SP2 应用系统登录成功

登录跳转

用户到 SP 应用页面，会跳转到 IDaaS 登录页面进行统一认证，登录后，返回一个包含更多用户信息 JWT token 到 SP 页面，包括应用列表和访问一个 SP 应用的子token，从而实现单点登录。如下图所示：



（1）用户访问 SP1 系统登录页面

（2）浏览器向 SP1 请求登录页面

（3）SP 1 返回重定向地址

（4）浏览器访问重定向地址

（5）IDaaS 返回登录页面

（6）浏览器携带用户提交的用户名密码以及应用id，向 IDaaS 请求登录

（7）IDaaS 本地认证成功后，创建主 session ，并返回应用列表的 JWT token 内含了子 token

（8）浏览器接收信息后，校验并解析出子 token ，向 SP1 请求认证登录

（9）SP 1 获取子 token 并解析后，创建子 session ，并返回登录后页面

（10）用户看到 SP1 登录后页面， SP1 登录成功

（11）用户在 SP1 已成功登录后，访问 SP2 登录页面

（12）浏览器向 SP2 请求登录页面

（13）SP2 返回重定向地址

（14）浏览器访问重定向地址

（15）IDaaS 由于主 session 已创建，可直接根据应用id生成 SP2 系统的子 token ，并返回子 token

（16）浏览器携带子 token ，向 SP2 发起请求

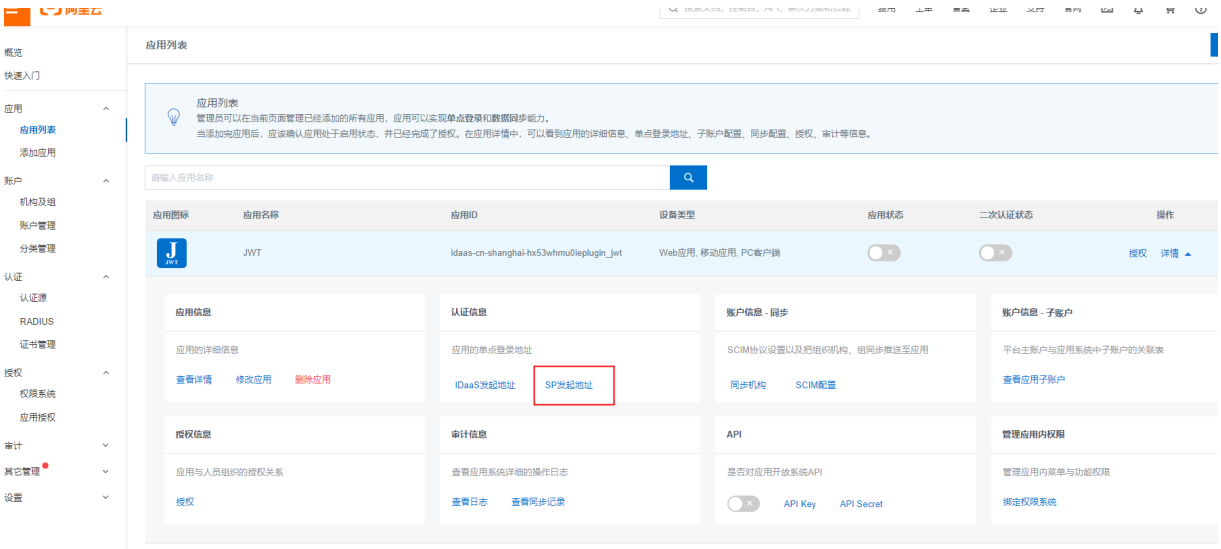
（17）SP2 解析子 token 后，创建子 session 并返回 SP2 登录后页面

（18）用户看到 SP2 登录后页面， SP2 登录成功

FAQ

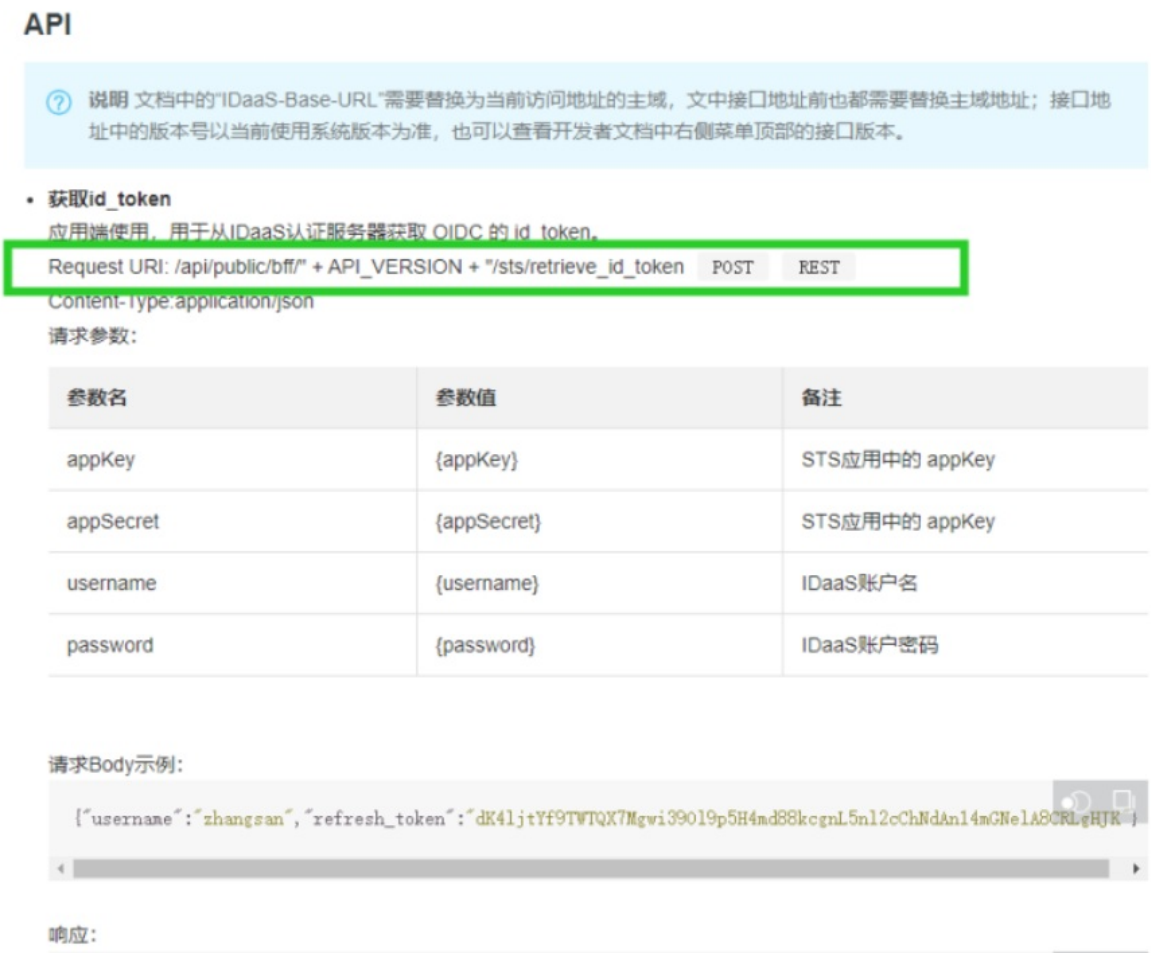
1. SP发起登录和接口后置有什么区别

SP向IDaas发起， 是直接使用IDaaS的应用模板进行配置，使用IDaaS提供的登录页面进行登录，如下图位置获取sp发起地址； 接口后置是使用API令牌(STS)获取id_token， 是应用集成IDaaS的登录接口， 登录页面由应用自身提供



2. 接口后置使用哪个接口

使用STS安全令牌的获取id_token的接口，用户在应用页面输入完信息后， sp转发请求给IDaaS做验证，IDaaS正常返回id_token代表验证用户成功 。接口：/api/public/bff/v1.2/sts/retrieve_id_token
https://help.aliyun.com/document_detail/145016.html



3. 如何确定使用哪个协议模板对接，是否需要开发

如果是支持标准协议的，如jira，confluence，可以自行在管理员侧选择对应模板，如SAML模板进行配置；
如果是登录方式简单的，只有用户名和密码登录，并且不是前后端分离，可以尝试使用表单代填自己对接；
如果上面两种方式不行，那么应用需要改造以对接单点登录，IDaaS推荐使用JWT方式，集成简单，直接集成IDaaS提供的SDK实现。

1.2. JWT

简介

整个插件式 JWT 的流程，是接收 IDaaS 平台向 callback url 发出的 id_token 参数（即 JWT 令牌），并使用我们提供的（或第三方提供的） JWT 解密库/方法对 JWT 进行解析，并验证身份。 你可以自己按照这个逻辑完成代码，也可以在我们提供的 代码/demo 的基础上进行修改。

实现流程

1. 通过浏览器登录集成的 IDaaS 系统后，确认要单点登录的应用，发起 SSO 请求到 IDaaS 系统。
2. IDaaS 生成 token 令牌发送到业务应用。
3. 你的应用获取到 token 令牌，用我们提供的插件或方法解析 token 令牌，解析成功获取到用户信息并验证后，重定向进行登录；如果解析失败则拒绝登录。

JWT 具体对接

请查看 JWT 模板使用指南

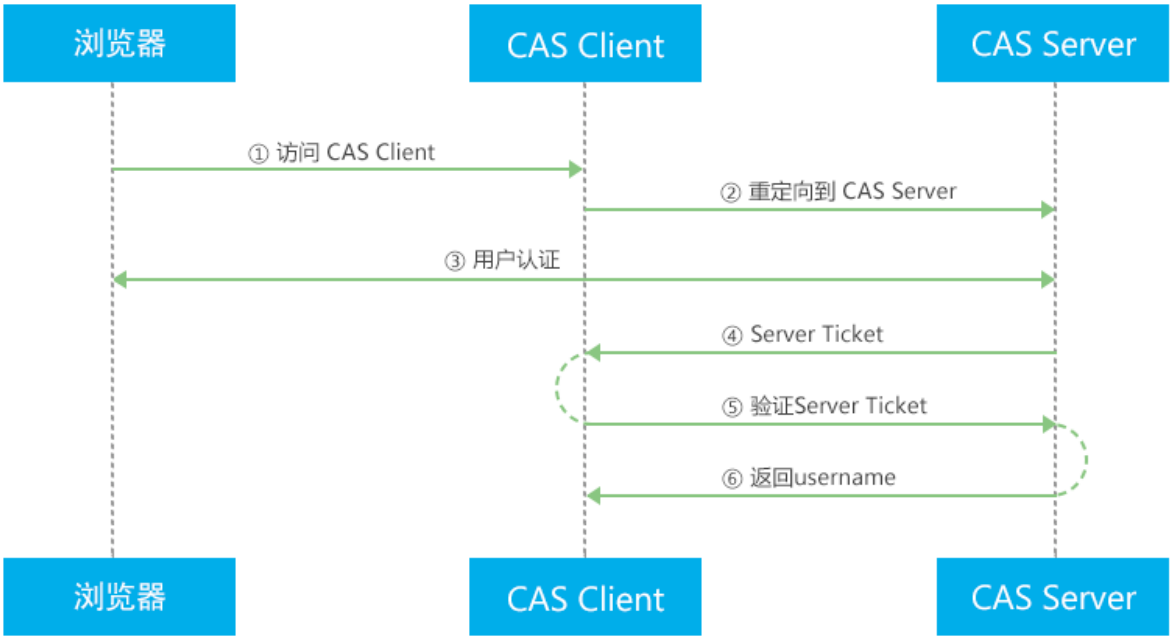
1.3. CAS

使用CAS 标准时，首先是由 CAS Client 发起， CAS Client 会重定向到 CAS Server（由 IDaaS 充当）进行登录，由 CAS Server 进行账户校验且多个 CAS Client 之间可以共享登录的 session ， Server 和 Client 是一对多的关系；

原理和协议

从结构上看，CAS 包含两个部分： **CAS Server** 和 **CAS Client** 。 CAS Server 需要独立部署，主要负责对用户的认证工作； CAS Client 负责处理对客户端受保护资源的访问请求，需要登录时，重定向到 CAS Server。

下图是标准 CAS 最基本的协议过程：



CAS Client 与受保护的客户端应用部署在一起，以 Filter 方式保护受保护的资源。对于访问受保护资源的每个 Web 请求， CAS Client 会分析该请求的 Http 请求中是否包含 Service Ticket。如果没有，则说明当前用户尚未登录，于是将请求重定向到指定好的 CAS Server 登录地址，并传递 Service （也就是要访问的目的资源地址），以便登录成功过后转回该地址。

用户在上图流程中的**第 3 步**输入认证信息，如果登录成功，CAS Server 随机产生一个相当长度、唯一、不可伪造的 Service Ticket，并缓存以待将来验证。之后系统自动重定向到 Service 所在地址，并为客户端浏览器设置一个 Ticket Granted Cookie（TGC），CAS Client 在拿到 Service 和新产生的 Ticket 过后，在第 5，6 步中与 CAS Server 进行身份核实，以确保 Service Ticket 的合法性。

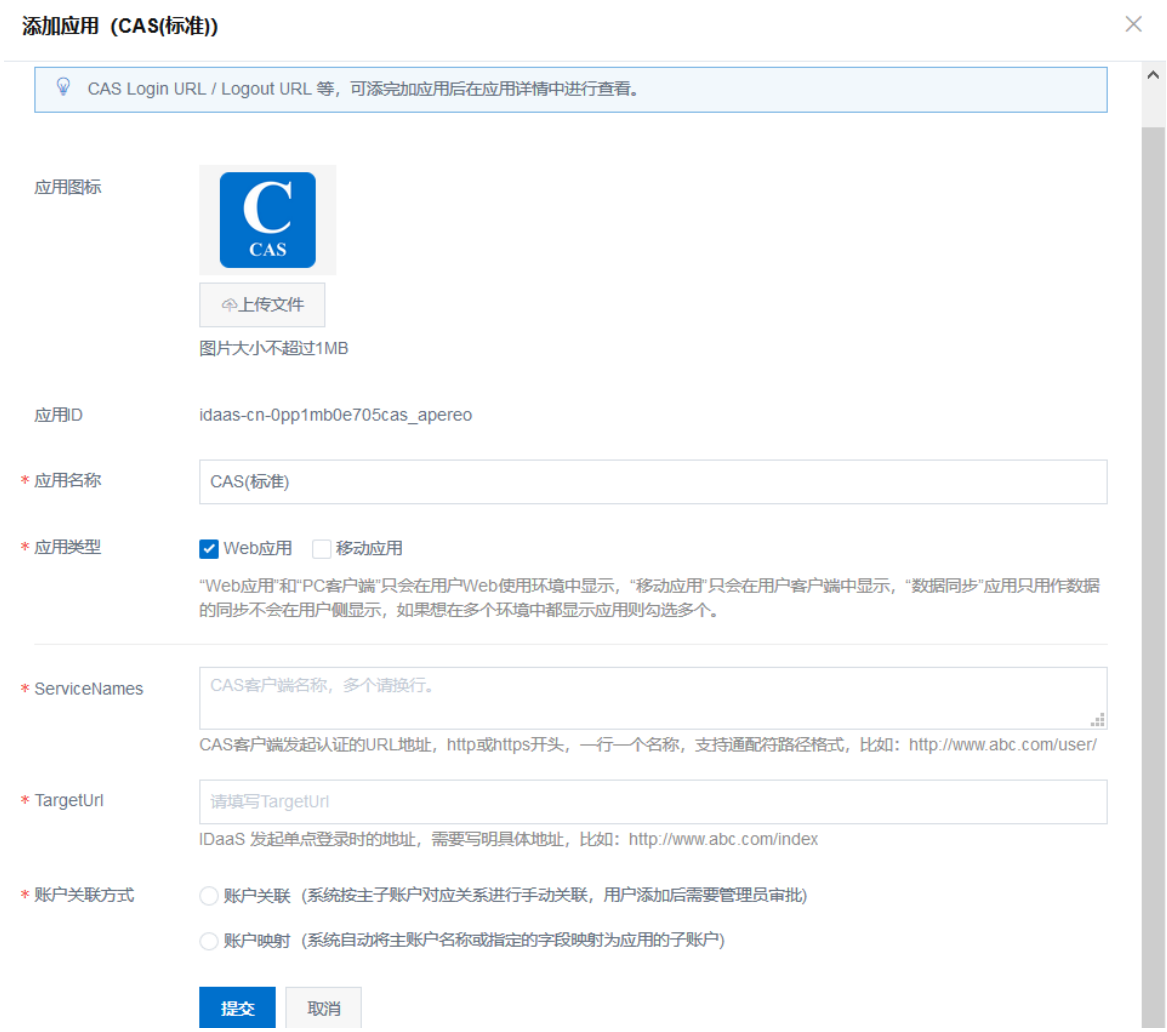
在 IDaaS 中， CAS （标准）应用模板实现了标准的 CAS 流程。它充当一个 CAS Server的角色。当 CAS Cient 决定使用IDaaS作为 CAS Server 时。在登录认证时需要使用 IDaaS 系统中公司的主账号，密码进行认证。

IDaaS 中添加 CAS （标准）应用

> **说明** CAS 标准应用目前只能由 IT 管理员 在应用添加菜单中添加。下面是 IT 管理员的应用添加流程。如果希望使用 CAS（标准）单点登录，可以请管理员进行协助添加和配置。

1. 以IT管理员身份登录 IDaaS，点击添加应用。找到 CAS （标准），点击 添加应用

2. 配置CAS 应用



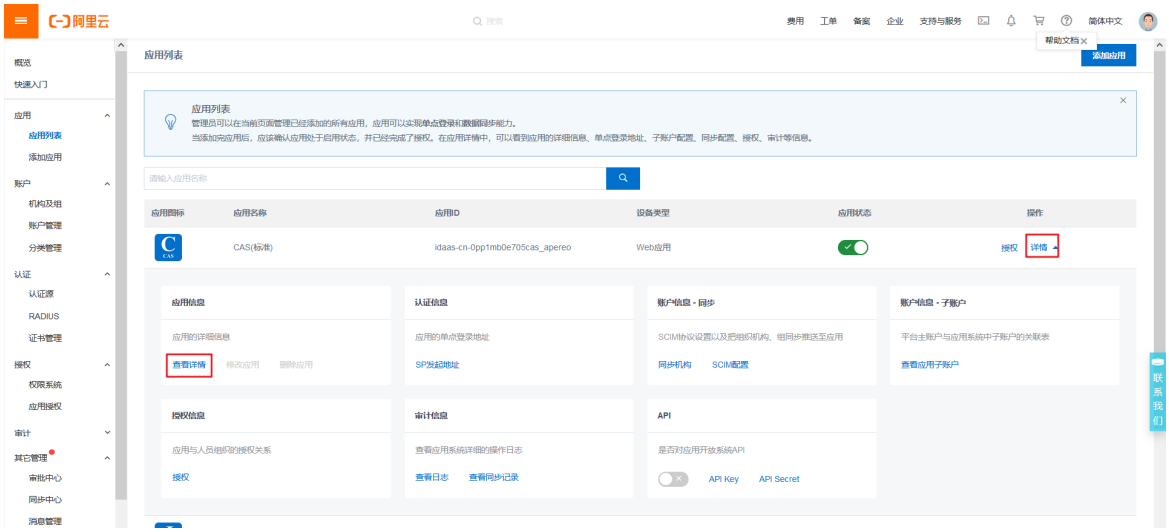
CAS Client 也就是业务系统需要提供两个参数：

ServiceNames：CAS Client 发起认证的URL地址。如有您有多个 CAS Client，直接换行添加即可。

> **说明** 如果您的CAS Client 在内网，经过反向代理到外网。如果您的ServiceName参数值是自动获取的，有可能获取到的ServiceName值是内网的 IP 地址，需要您将ServiceName参数值写成固定的URL地址。

TargetUrl：IDaaS发起单点登录地址。

3. 点开应用详情，主要注意 **CAS Login URL** 和 **CAS Server URL Prefix** 两个参数以便接下来在 CAS Client里面配置使用。

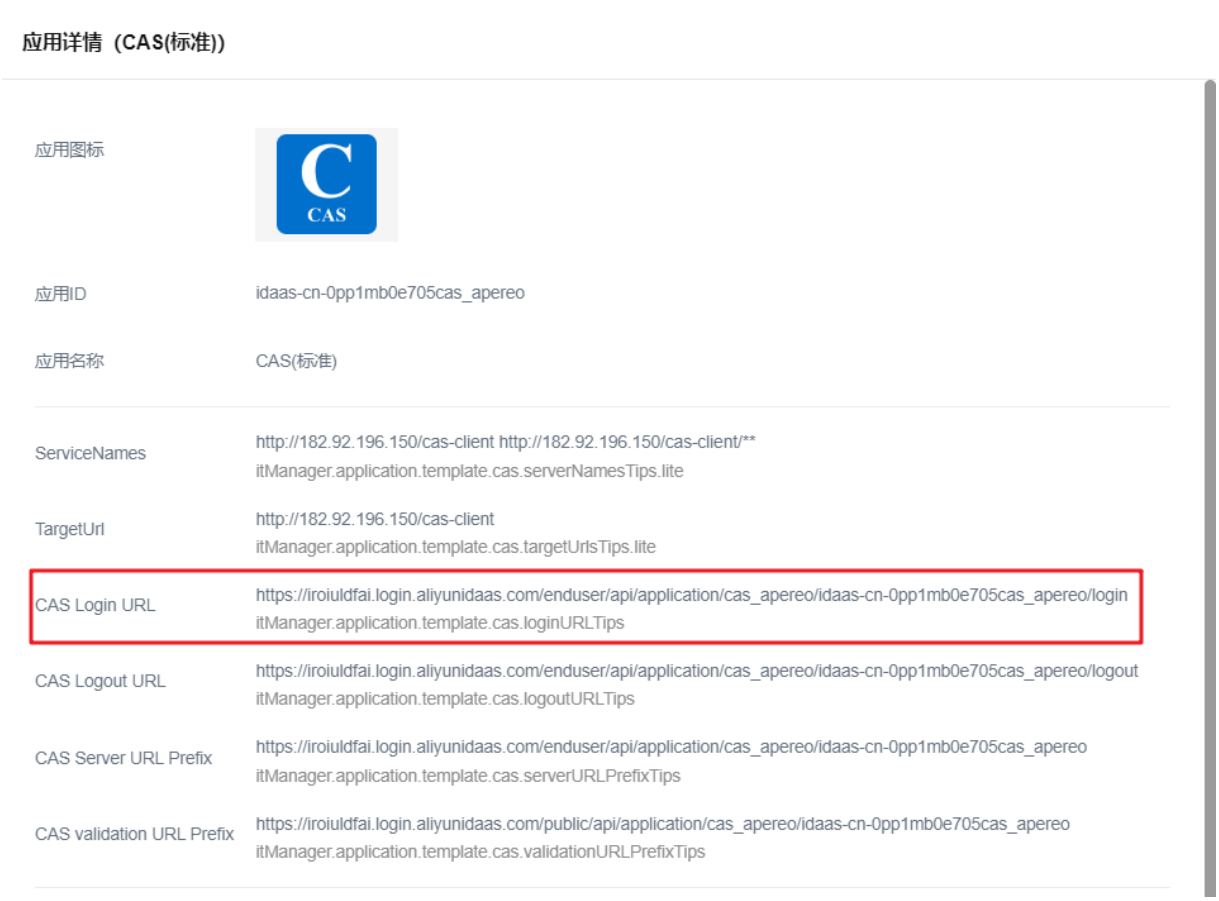


在 CAS Client 中使用 CAS（标准）应用作为 CAS Server

修改 CAS Client 的配置信息，一般是修改 web.xml 文件。

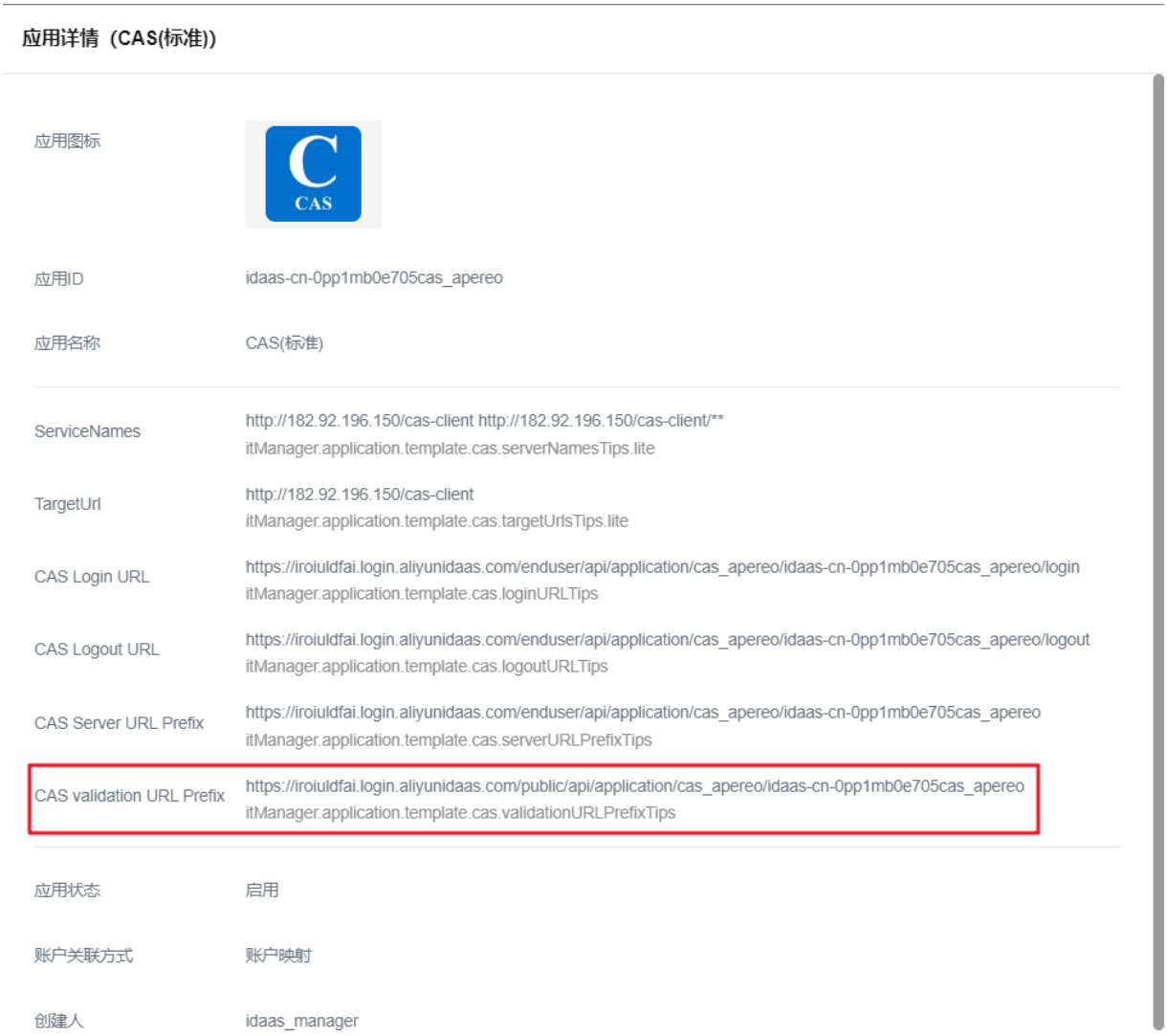
1、修改 casServerLoginUrl 参数值，修改为 IDaaS 中创建的 CAS 应用的 CAS Login URL 值。

> **说明** CAS logout url 参数值配置也类似。如果您有使用到，直接复制粘贴IDaaS中CAS应用的CAS Logout URL即可。



2、修改 casServerUrlPrefix 参数值，修改为 IDaaS 中创建的 CAS 应用的 CAS validation URL Prefix 值。

> **注意** CAS票据校验地址的URI地址为 public 开头的，与上面的其他地址不一样。



说明 CAS validation URL Prefix 为 CAS 票据验证地址的前缀。CAS Client 请求 CAS server 验证 ticket 的完成 URL 请求为 {{CAS validation URL Prefix}}/serviceValidate?ticket=''&service='ServiceNames'

1.4. 常见问题

Q：有插件和无插件集成的区别是什么，哪种方式更好一些？

A：这就得根据我们所用的协议来决定了：

- JWT 协议的流程是，浏览器在 IDaaS 中发起一个 SSO 请求的时候，IDaaS 会利用秘钥产生一个token，然后将 token 放到请求里面作为参数 id_token 的值传到业务系统中去，业务系统就需要解析这个 token 进行身份识别，但是解析token 的关键就是需要集成我们提供的一个解析使用的 SDK，结合业务系统在 IDaaS 中添加应用的时候生成的一个不变的公钥，SDK+公钥才能对 token 进行校验识别；

- CAS 协议的流程是浏览器在 IDaaS 中发起一个 SSO 请求的时候 IDaaS 会产生一个一次性随机 code（类似于 JWT 中的 token）作为参数 code 的值传到业务系统，然后业务系统并不解析这个 code，而是通过一个 callback 回调在将其原封不动的传给 IDaaS， IDaaS 对这个code进行验证，如果是我们传过去的就说明请求合法，IDaaS 会将用户信息传给业务系统，如果不是就验证失败。

Q：每个用户登录后，系统会返回一个 token，这个 token 在我们服务器端是永远生效的还是我们要注销之后才会失效？业务系统怎么保持登录状态，是每次页面请求需要请求token 吗？

A：token 是有一定的时效的，业务系统得到我们的 token 后可以建立自己系统的 session 机制。session 当然有自己的 timeout 设定， 有了 session 以后，每次业务应用的访问和我们是无关的。每一次获取到 token 的请求 IDaaS 都会返回一个状态码，可以根据我们返回的状态码来判断 token 是否失效，如果失效就通过刷新 token 来重新获取 session 。

Q：JDK 版本问题，导致 jar 包无法使用？

A：根据 JDK 的版本提供了 3 个版本的 jar 包，1.8版本的可以直接导入 jar，并且添加依赖。JDK1.6或者 JDK1.7需要手动导入相关的依赖 jar 包。请根据自己的 jdk 版本选择相对应的 jar 包。

2.API令牌（STS）

API令牌（STS）概述 v1.2

为了将众多的API数据源快速安全整合起来，客户可以通过我们的STS（SecurityToken Service，安全令牌服务）服务， API令牌STS是使用OIDC协议实现的API认证与授权解决方案，它将API认证授权集成到我们的IDaaS平台，生成密钥对后，即可快速导出公钥，更可以设定不同的认证方式，让移动及API应用的认证与授权更安全。

立刻获得基于token的身份校验方式，让自己的应用拥有和我们一样的身份安全等级。拥抱安全，告别长期不变的密码，或是固定设备ID等容易被捕获重放的认证方式。

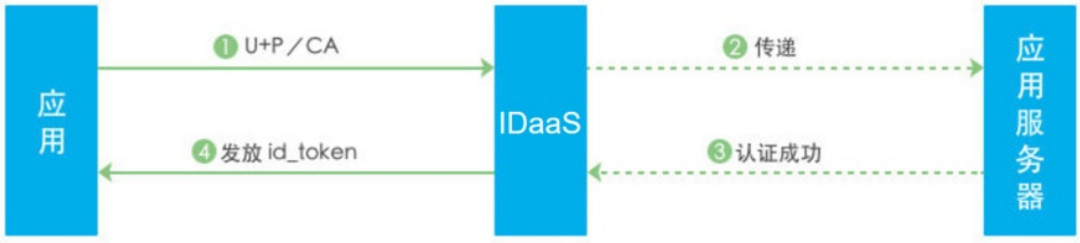
实现原理

STS服务要求开发的应用端和服务器端做出相应的修改。应用端(Web)不再直接向服务器端发送认证请求或索取资源，而是先往IDaaS的认证服务器（AS, Authentication Server）发送账号验证请求，IDaaS在向你的服务器验证成功后，会返回一个id_token作为身份令牌。向服务器发出请求的时候，就需要携带这个令牌以验证身份。

获取id_token

IDaaS的认证平台负责id_token的分发，需要进行认证的账号已经在IDaaS平台的情况下，也可以对用户的身份进行自己认证；如果IDaaS没有用户信息，不能自己进行认证，则需要传递到开发者的应用服务器，由应用服务器进行身份认证，并返回结果给IDaaS。具体流程如下图：

步骤分解：



应用携带账号认证信息向IP

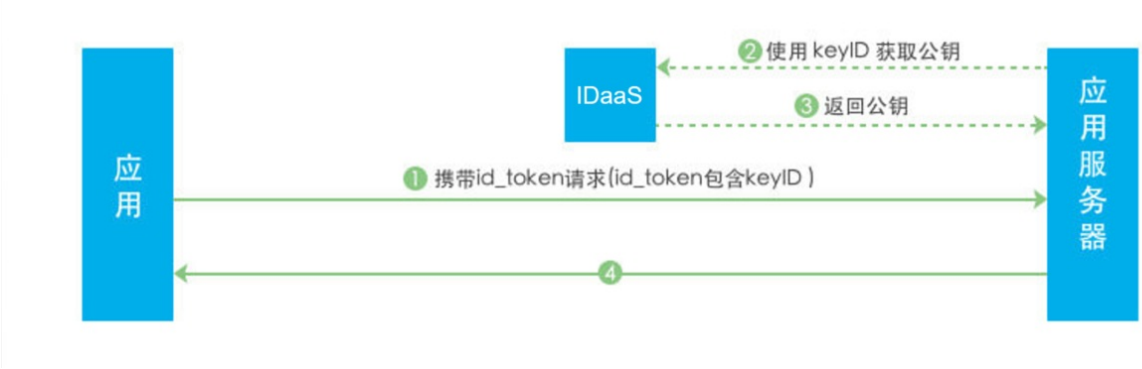
1. IDaaS发出id_token申请，认证信息可以使用户名+密码，也可以是证书。
2. IDaaS如果可以自行校验，直接根据情况返回校验结果以及id_token。如果不能自行校验，会将信息传递到应用服务器开发的接口，请求应用服务器对认证信息进行验证。
3. 应用服务器验证完认证信息返回结果。
4. IDaaS从应用服务器接收到验证成功或失败的结果，并根据情况返回id_token给应用。

> 说明
>
> 如果在创建STS应用的时候启用了Refresh Token，那么在最后一步从IDaaS认证服务器还会返回一个refresh_token给应用。应用可以将其存储起来，用来进行id_token过期并需要刷新的请求。

使用id_token

开发者需要对服务器需要被id_token保护起来的接口添加一个过滤器（filter）。只有通过了该过滤器验证的才可以访问。具体服务器端实现流程如下：

1. 获取keyID：每次应用请求的时候会携带id_token，服务器的过滤器需要对其进行初步解析并获得keyID。
2. 获取publicKey：keyID是密钥的全局唯一标识，我们需要使用这个keyID对应的公钥，来对id_token进行解密。如果这个keyID对应的公钥在应用服务器上有保存的话，无须再次申请直接使用。没有存储记录的话，那么需要用携带该keyID向IDaaS认证服务器请求得到公钥publicKey，并安全地保存到本地。
3. 解密验证：使用公钥进行解密，如果成功的话，那么认证即成功，请求可以通过。



步骤分解：

1. 开发的应用向应用服务器请求资源，携带从IDaaS认证服务器获取到的id_token。
2. 应用服务器对id_token进行解析获得keyID，如果应用服务器本地没有与keyID对应的公钥，那么向IDaaS认证服务器请求。
3. 应用服务器得到公钥。
4. 应用服务器对id_token进行解密，在验证后通过过滤器，继续放行想要请求的资源，并返回给应用。

刷新id_token

如果添加的API令牌已经启用了Refresh Token功能，当id_token过期的时候，可以通过refresh_token去刷新换取一个新的id_token。

如果没有启用Refresh Token功能，那么过期后需要按照获取id_token的方式重新用身份认证信息去IDaaS获取。

步骤分解：

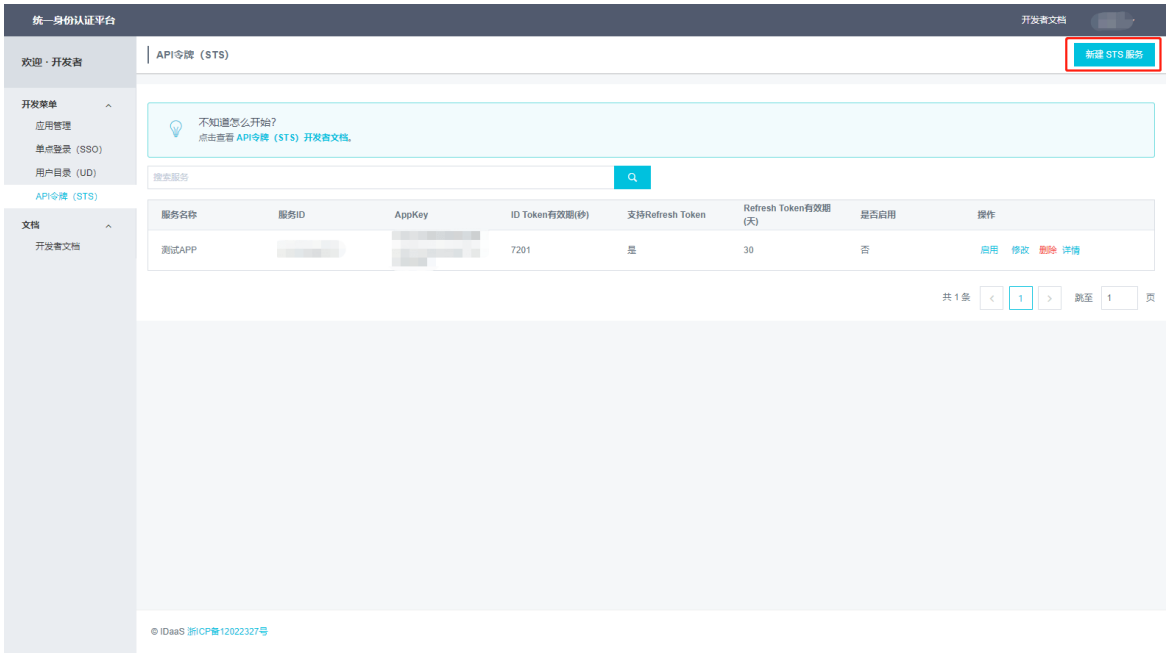
1. 应用携带refresh_token向IDaaS进行刷新id_token请求
2. IDaaS返回新的id_token给应用

申请STS应用

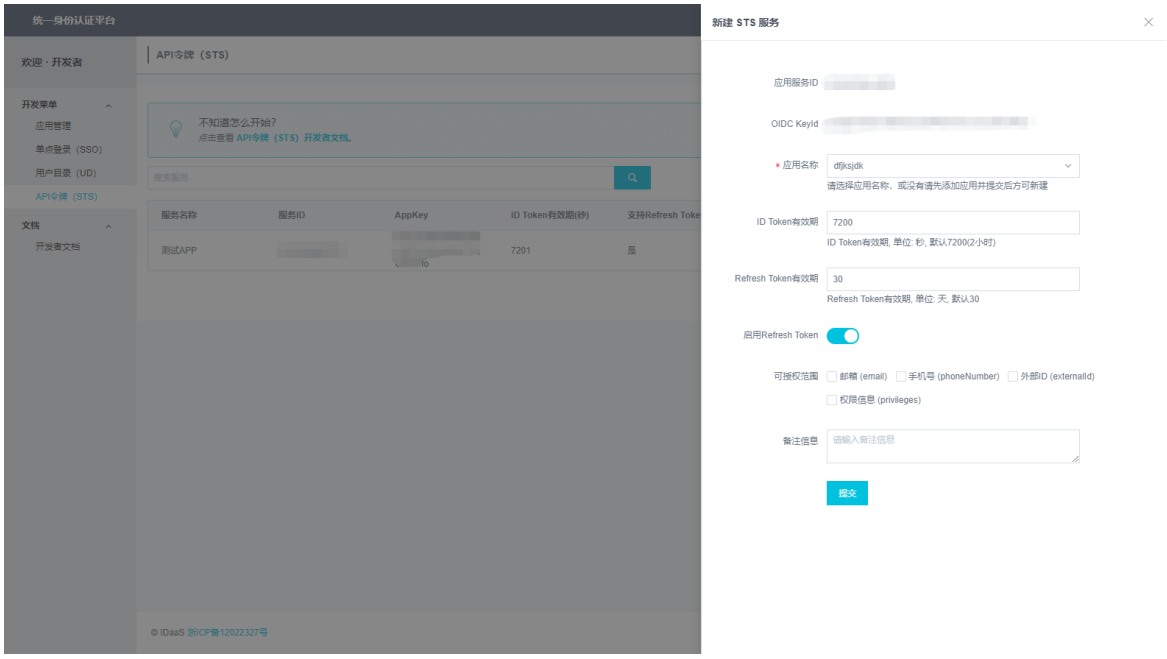
在开发之前，我们需要先申请一个STS应用，对应我们要开发的STS功能。

添加STS应用只能使用开发者角色进行创建，访问开发者。

1. 在开发者的后台管理系统中，选择左侧的列表中选择API令牌（STS），选择添加API令牌（STS）。如下图：



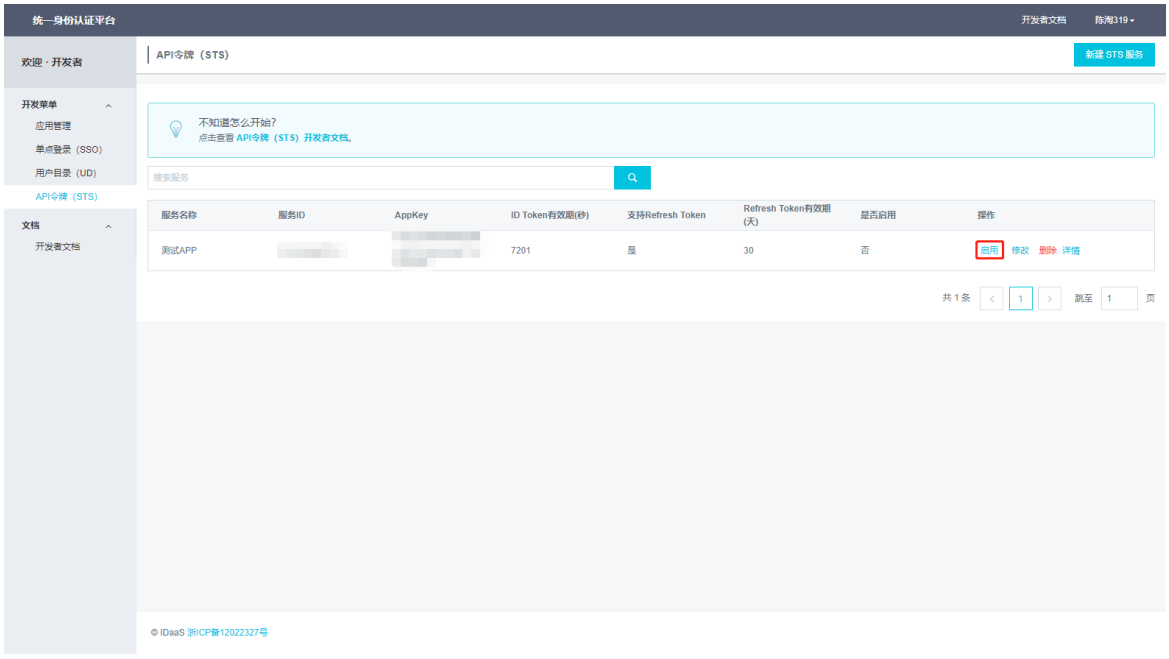
2. 填写STS应用信息。如下图：



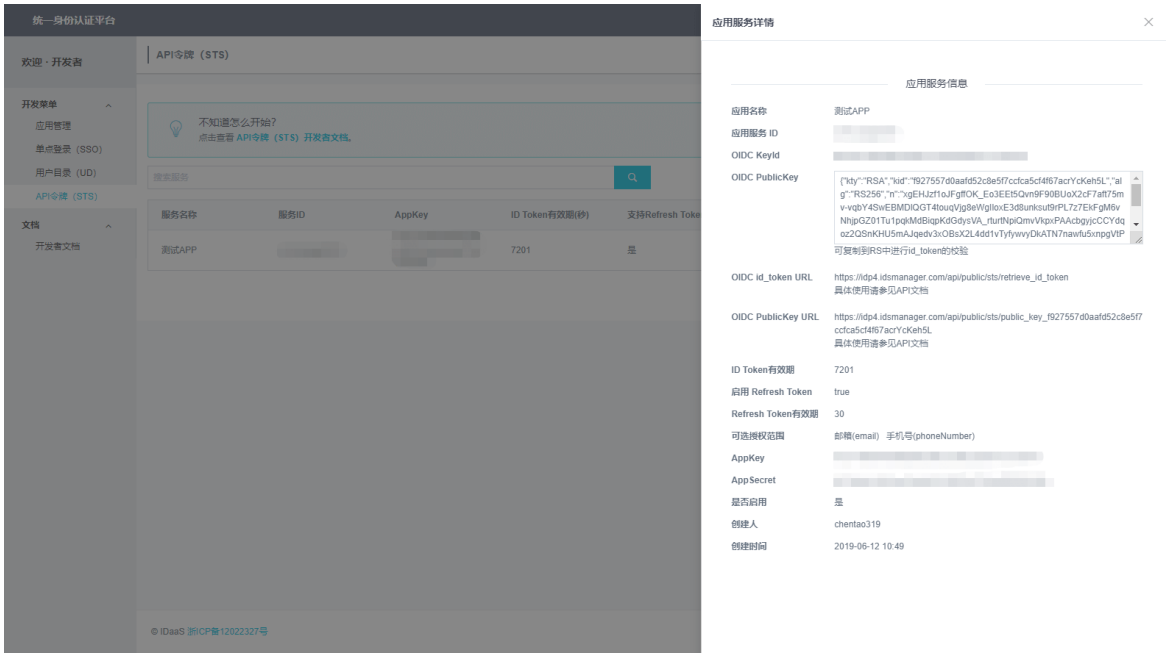
表单字段名称	说明
应用ID	默认生成的，应用的唯一标识

表单字段名称	说明
OIDC KeyId	默认生成的唯一标识，在服务器端向IDaaS认证服务器请求获取公钥的时候会使用到
应用名称	选择要添加的应用
ID Token有效期	ID Token有效期，单位：秒，默认7200(2小时)
Refresh Token有效期	Refresh Token有效期，单位：天，默认30
启用Refresh Token	是否启用Refresh Token有效期
可授权范围	接口可获取的参数
备注信息	可选项，可以备注应用的相关信息

3. 添加应用完成后，在操作中选择启用，确定后即可集成使用STS服务。如下图：



4. 点击详细查看添加的STS应用的详细信息。如下图：



API

> **说明**
>
> 文档中的“IDaaS-Base-URL”需要替换为当前访问地址的主域，文中接口地址前也都需要替换主域地址；接口地址中的版本号以当前使用系统版本为准，也可以查看开发者文档中右侧菜单顶部的接口版本。

- **获取id_token**

  应用端使用，用于从IDaaS认证服务器获取 OIDC 的 id_token。

  Request URI: /api/public/bff/v1.2/sts/retrieve_id_token `POST` `REST`

  Content-Type:application/json

  请求参数：

参数名	参数值	备注
appKey	{appKey}	STS应用中的 appKey
appSecret	{appSecret}	STS应用中的 appSecret
username	{username}	IDaaS账户名
password	{password}	IDaaS账户密码

  响应示例：

```
{"statusCode":0,"errors":[],"id_token":"eyJhbGciOiJSUzI1NiIsImtpZCI6IjY2...","refresh_token":"Z0zGd07UFFcAL6AgOYGn...","successful": true}
```

statusCode为0表示成功，HTTP响应码为401(Unauthorized)则表示失败，具体为:400表示请求参数错误，483表示系统未找到refresh_token数据，484表示refresh_token已过期，501表示账户名错误，479表示应用不再支持 refresh_token 操作。

- **通过refresh_token刷新id_token**

  应用端使用，用于从IDaaS通过refresh_token获取id_token

  Request URI: /api/public/bff/v1.2/sts/refresh_id_token `POST`

  Content-Type: application/json

  请求参数:

参数名	参数值	备注
username	{username}	IDaaS账户名
refresh_token	{refresh_token}	获取id_token时返回的refresh_token

  请求Body示例:

```
{"username":"zhangsan","refresh_token":"dK4ljtYf9TWTQX7Mgwi39Ol9p5H4md88kcgnL5nl2cChNdAn14mGNelA8CRLgHJK"}
```

  响应:

```
{"statusCode":0,"errors":[],"id_token":"eyJhbGciOiJSUzI1NiIsImtpZCI6IjY2...","refresh_token":"Z0zGd07UFFcAL6AgOYGn...","successful": true}
```

  statusCode为0表示成功，HTTP响应码为401(Unauthorized)则表示失败，具体为:400表示请求参数错误，483表示系统未找到refresh_token数据，484表示refresh_token已过期，501表示账户名错误，479表示应用不再支持 refresh_token 操作。

- **通过keyId获取OIDC Public Key**

应用服务器端使用，用于从IDaaS通过keyId获取OIDC Public Key公钥。

Request URI: /api/public/bff/v1.2/sts/load/public_key `GET`

请求参数:

参数名	参数值	备注
keyId	{keyId}	STS应用的 keyId

请求示例:

```
http://sz.idp-local.com/api/public/bff/v1.2/sts/load/public_key?keyId=d037e0cbf59042d09ed4609127a37ff5pbb8c5dwEn8
```

响应：

```
{"statusCode":0,"errors":[],"publicKey":"eyJrdHkiOiJSNjd2NE5ZbtZDRBbzRZaUVFBUTMtRGdTaVB0ekdtRkZ...","successful":true}
```

> 说明
>
> 提示: 返回的publicKey是Base64 Encoded，使用时需要先调用Base64 Decoded。

statusCode为0表示成功，HTTP响应码为401(Unauthorized)则表示失败，具体为: 233表示未找到对应的keyId，482表示应用尚未启用。

常见问题

1. 场景：登录时，通过IDaaS接口进行验证流程

使用STS安全令牌的获取id_token的接口，用户在应用页面输入完信息后， sp转发请求给IDaaS做验证，IDaaS正常返回id_token代表验证用户成功 。接口：/api/public/bff/v1.2/sts/retrieve_id_token

2. 场景：应用是前后端分离，使用IDaaS接口做认证后，前端和后端交互流程

如移动端APP 需要访问SP的rest资源，可以集成STS令牌进行接口保护。 当用户在APP中输入账户和密码后，APP调用IDaaS接口获取id_token, APP携带id_token向SP发送请求， SP需要先通过在STS应用上复制的keyId获取public_key, 并使用public_key 校验APP请求中带的id_token，id_token校验通过，则认证通过。public_key校验的demo需要到开发者中下载

SP验证APP身份后，后续APP 向SP发送请求，使用什么凭据：方案1： 仍然使用id_token，每次请求带上id_token向SP发送请求。因为id_token有时效性， 过期可以通过refresh_token获取新的id_token。方案2: 使用SP自己生成的token，app获取SP颁发的新token进行访问，灵活控制有效性。

3. 下载解析token的demo

开发者访问方式：https://help.aliyun.com/document_detail/147490.html

下载解析token的demo

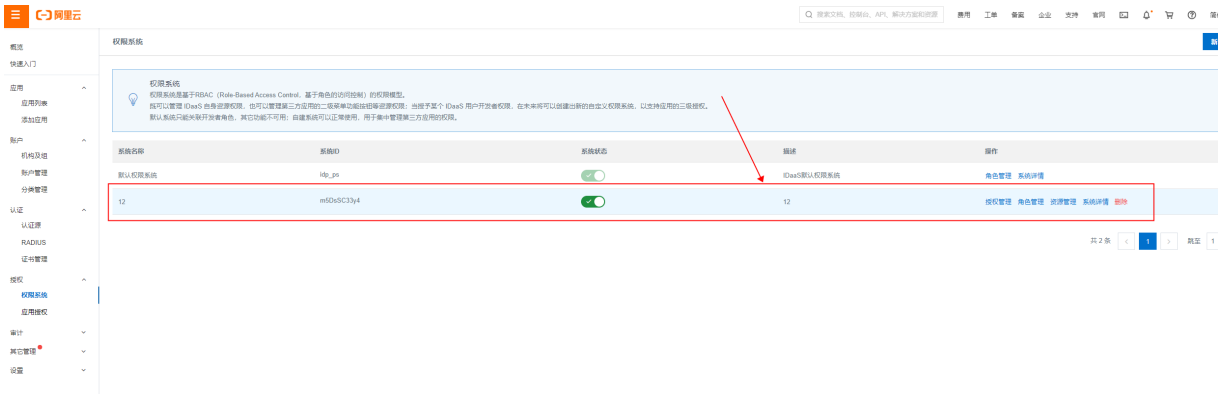
4. 当使用接口refresh_id_token刷新id token后，为什么生成的refresh token还是原来的

当刷新id token后，新生成的id token重新开始计算有效期，但是refresh token在有效期内是不变的，可以多次用于刷新id token。

可以在STS应用详情中查看refresh token有效期。

3.权限系统API

针对自建系统（对应SP的权限）的相关操作，支持接口访问，具体请查看权限系统API。



4.开发指南FAQ

IDaaS有哪几种获取token方式，不同token是否可以通用

1. 同步账户接口

接口具体信息，请查看SP(第三方应用)推数据至IDaaS。

此处token，实现从应用同步组织和账户到IDaaS，实现账户的同步。

2. 获取STS令牌

接口具体信息，请查看API令牌(STS)。

此处token主要用于登录认证，当用户登录成功后，token中会携带用户相关信息，用于应用识别账户。

3. 权限系统接口

接口具体信息，请查看权限系统接口。

此处token，主要用于调用权限接口，当用户登录应用后需要鉴别用户的细粒度权限，调用权限接口进行鉴权。

这三个token权限不同，不能交互使用。

Base url获取方式

访问IDaaS EIAM实例列表，获取用户登录地址

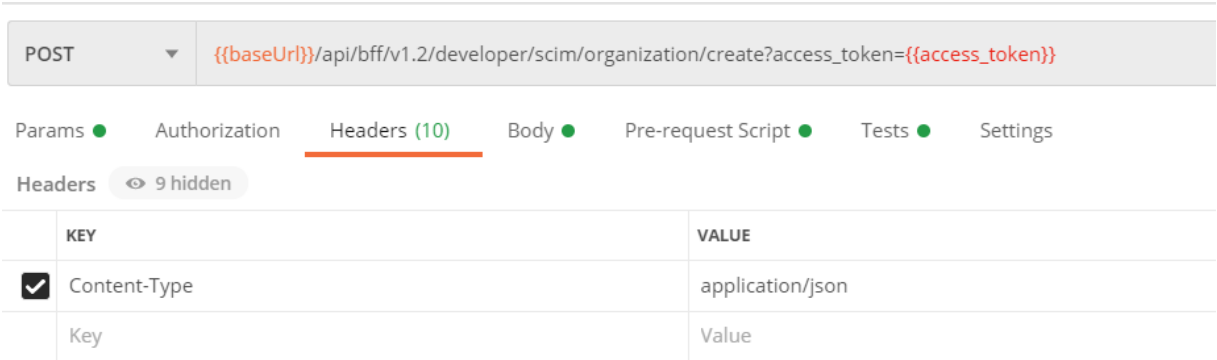
Token传输举例

当接口需要携带token进行请求时，支持两种方式请求。

1. 放到请求中

URL?access_token={access_token}

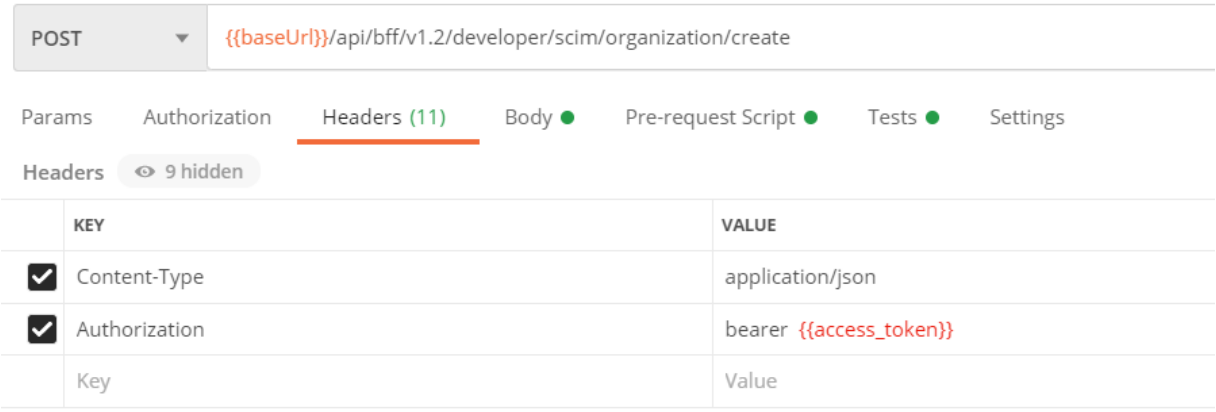
请求示例：



2. 放到header中

Authorization bearer {access_token}（注意 bearer与access_token之间的空格）

请求示例：



什么情况下需要访问开发者角色

- IDaaS开发文档，以在线帮助文档为准，见上面三种token获取方式文档。
- 只有在STS对接场景下，需要在开发者中创建STS应用，其它情况不需要访问开发者角色。