阿里云
访问控制

用户指南

文档版本：20190314

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
	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，恢复业务所需时间约10分钟。
	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。	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。	单击 确定。
`courier` 字体	命令。	执行 `cd /d C:/windows` 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。	`swich` *`{stand | slave}`*

目录

1 概述

本文主要介绍 RAM 的核心功能及其典型应用场景。

核心功能

身份管理

- 用户
- 用户组
- 联合登录概述
- 外部 IdP 的 SAML 设置
- 企业 IdP 与阿里云身份联合示例
- 管理 RAM 角色

权限管理

- 权限策略管理
- RAM 授权
- 验权规则

典型场景

- 用户管理与分权
- 针对移动设备应用的临时授权
- 跨云账号的资源授权
- RAM 资源分组与授权
- 云上应用的动态身份与授权管理
- 使用企业本地账号登录阿里云

2 身份管理

2.1 用户管理

2.1.1 用户

用户，是 RAM 中的一种身份。用户对应某一个操作实体，如运维操作人员或应用程序。通过创建新的 RAM 用户并授权，RAM 用户便可以访问相关资源。

创建 RAM 用户的前提条件

使用云账号（或拥有 RAM 操作权限的 RAM 用户）登录 *RAM* 控制台。

创建 RAM 用户

1. 在 RAM 控制台首页，人员管理菜单下，单击用户 > 新建用户。
2. 按照页面提示，输入登录名称和显示名称。

   说明:
   单击添加用户，可一次性创建多个用户。
3. 选择用户的访问方式为控制台密码登录或编程访问。

   说明:
   建议仅为用户选定一种访问方式。

   - 若访问方式选择控制台密码登录：可以同时完成对登录安全的基本设置。包括自动生成或自定义登录密码、是否要求登录时重置密码，以及是否要求开启多因素认证。
   - 若访问方式选择编程访问：将会自动为用户创建 AccessKey（API 访问密钥）。

管理 RAM 用户

创建 RAM 用户后，您可以根据使用需求管理和调整用户配置。

- 修改用户基本信息

  1. 在 RAM 控制台的用户页面，找到需要修改基础信息的用户，然后单击其用户名。

     说明:

可使用用户名进行模糊查询。

2. 在用户基本信息区域，单击编辑基本信息。
3. 在弹出的编辑基本信息界面，可以修改用户的用户名、显示名称和备注，可以选择性地填入用户的手机号码和邮箱。

- 管理控制台登录

为需要通过控制台进行操作的用户设置登录密码。

1. 在 RAM 控制台的用户页面，找到需要修改基础信息的用户，然后单击其用户名。

说明:
可使用用户名进行模糊查询。

2. 在控制台登录管理区域，单击修改登录设置。
3. 在弹出的修改登录设置界面，为用户修改登录设置。登录设置包括：
   - 是否开启控制台密码登录。
   - 设置登录密码。
   - 是否要求重置密码。
   - 是否开启多因素认证。

- 多因素认证设备

多因素认证（Multi-Factor Authentication，MFA）是一种简单有效的最佳安全实践，它能够在用户名和密码之外再增加一层安全保护。这些多重要素结合起来将为您的账户提供更高的安全保护。

1. 确保控制台登录管理设置时，修改登录设置已选择要求开启 MFA 认证。
2. 用户在登录控制台后，会自动出现 MFA 绑定流程，根据流程指引完成 MFA 设备的绑定。

启用 MFA 后，用户登录阿里云时，系统将要求输入两层安全要素：

- 第一安全要素：用户名和密码。
- 第二安全要素：来自其虚拟 MFA 设备的可变验证码。

虚拟 MFA 设备是产生一个 6 位数字验证码的应用程序，它遵循基于时间的一次性密码（TOTP）标准 *RFC 6238*。此应用程序可在移动硬件设备上运行（例如智能手机）。

- 管理访问密钥（AccessKey）

  创建访问密钥

  为需要通过 API 进行调用的用户创建访问密钥（AccessKey）。

  1. 在 RAM 控制台的用户页面，找到需要创建 AccessKey 的用户，单击其用户名。

     说明:
     可使用用户名进行模糊查询。

  2. 在用户 AccessKey区域，单击创建新的 AccessKey。
  3. 在弹出的对话框中，查看新建的 AccessKey 信息并及时保存。

     说明:

     - AccessKeySecret 只在创建时显示，不提供查询。目前可以查询的信息：AccessKeyID、状态、最后使用时间、创建时间。
     - 若 AccessKey 泄露或丢失，则需要创建新的 AccessKey。

  禁用访问密钥

  在 用户AccessKey 页面下：

  - 单击禁用可以禁用 AccessKey。
  - 单击激活可以重新激活 AccessKey。

  删除访问密钥

  注意:
  删除访问密钥需慎重，在使用中的 AccessKey 一旦删除，可能会造成客户应用系统故障。

  如果需要删除 AccessKey，可以通过 AccessKey 最后使用时间确认 AccessKey 的使用情况。

RAM 用户登录控制台

RAM 用户登录账号为 UPN (User Principal Name) 格式，即 RAM 控制台用户列表中所见的用户登录名称。

在 RAM 用户登录入口，用户可以两种使方式登录：

- UPN 完整格式<$username>@<$AccountAlias>.onaliyun.com。

  说明:

<$AccountAlias>.onaliyun.com为默认域名，详情请参考：域名管理。

- <$username>@<$AccountAlias>。

RAM 用户和云账号的登录入口不同，RAM 用户不能通过云账户登录页面进行登录。

RAM 用户的登录入口如下：*https://signin.aliyun.com/login.htm*。

说明:

通过登录 *RAM* 控制台，在概览页可以快速查询登录链接。

为 RAM 用户配置权限

用户创建完成之后，单击添加权限可以继续完成用户的权限配置。

1. 在已创建的用户列表中勾选需要授权的用户。
2. 点击添加权限打开权限配置面板，被授权主体会自动填入。
3. 选择需要授权给用户的权限策略，单击确定。

删除 RAM 用户

注意:

删除 RAM 用户需要谨慎操作。如果有业务系统正在以此用户身份运行，那么可能会导致客户的业务故障。

1. 在 RAM 控制台的用户页面，找到要删除的 RAM 用户。
2. 单击删除。
3. 在弹出的删除用户对话框中，单击确定。

更多信息

- 可以选择为用户添加到一个或多个组，对 RAM 用户进行分类并授权。详情请参考：用户组。
- 可以为用户添加一个或多个权限策略，使 RAM 用户具有资源的访问能力。详情请参考：*RAM* 授权。

2.1.2 用户组

若云账号下有多个 RAM 用户，通过创建用户组对职责相同的 RAM 用户进行分类并授权，从而更好的管理用户及其权限。

背景信息

对 RAM 用户进行分组有如下好处：

- 在 RAM 用户职责发生变化时，只需将其移动到相应职责的群组下，不会对其他 RAM 用户产生影响。
- 当群组的权限发生变化时，只需修改用户组的权限策略，即可应用到所有 RAM 用户。

创建用户组的前提条件

使用云账号（或拥有 RAM 操作权限的 RAM 用户）登录 *RAM* 控制台。

创建用户组

1. 在 RAM 控制台首页，人员管理菜单下，单击用户组 > 新建用户组。
2. 输入用户组名称、显示名称和备注，单击确定。

管理组成员

1. 在 RAM 控制台的用户组页面，找到需要管理基础信息的用户组，单击其用户组名。

   
   说明:
   可使用组名称进行模糊查询。

2. 单击组成员管理，可以管理用户组成员。
   - 添加组成员：单击添加组成员，从左列中勾选要添加到组中的用户（可使用关键字查询），单击确定。

     
     说明:
     单击“×”可取消选择。
   - 移出组成员：要找到要删除的用户，单击移出用户组，可将其从当前用户组中移出。

重命名用户组

1. 在 RAM 控制台的用户组页面，找到需要重命名的用户组，单击其用户组名。

   
   说明:
   可使用组名称进行模糊查询。

2. 单击编辑基本信息。
3. 输入显示名称并点击确定。

删除用户组

1. 在 RAM 控制台的用户组页面，找到需要删除的用户组，单击其用户组名。

   
   说明:
   可使用组名称进行模糊查询。

2. 单击删除。
3. 在弹出的删除用户组对话框中，单击确定。

> 说明:
> 此时会将用户从用户组中移出并移除用户组的授权。

更多信息

关于如何为用户组授权，请参考RAM 授权。

2.2 登录安全设置

2.2.1 基本安全设置

通过安全设置对所有 RAM 用户的登录密码、登录方式等进行设定。

登录密码设置

1. 在 RAM 控制台首页，单击人员管理 > 设置。
2. 在安全设置页面下，单击编辑密码规则。
3. 按照配置面板提示，配置密码长度、密码中必须包含元素、密码有效期、密码重试约束等规则，单击确定。

> 说明:
> 设置成功后，此密码策略适用于所有 RAM 用户。

用户安全设置

1. 在 RAM 控制台 首页，单击人员管理 > 设置。
2. 在安全设置页面下，单击修改 RAM 用户安全设置。
3. 按照配置面板提示，配置相关参数，单击确定。

2.2.2 域名管理

每个云账号有默认域名，除了默认域名，也可为账号配置域别名。RAM 用户可以通过默认域名或域别名登录，通过域名管理可以修改登录名后缀，便于用户记忆登录名称。

RAM 用户登录的前提条件

RAM 用户登录账号为 UPN (User Principal Name) 格式，即 RAM 控制台用户列表中所见的用户登录名称。

在 RAM 用户登录入口，用户可以两种使方式登录：

- UPN 完整格式<$username>@<$AccountAlias>.onaliyun.com。
- <$username>@<$AccountAlias>。

RAM 用户登录入口

RAM 用户和云账号的登录入口不同，RAM 用户不能通过云账户登录页面进行登录。

RAM 用户的登录入口如下：*https://signin.aliyun.com/login.htm*。

说明:

通过登录 *RAM* 控制台，在概览页可以快速查询登录链接。

域名管理

默认域名

1. 登录 RAM 控制台。
2. 在人员管理菜单下，单击设置 > 高级设置可以查询和修改默认域名。
   - 默认域名的格式为: <$AccountAlias>.onaliyun.com。
   - 若未设置过账号别名，账号别名默认值为 AccountID，此时默认域名的格式为：<$AccountID>.onaliyun.com。

域别名

除了默认域名，也可为账号配置域别名，RAM 用户同样可使用域别名登录。

1. 登录 RAM 控制台。
2. 在人员管理菜单下，单击设置 > 高级设置可以查询域别名。
3. 单击创建域别名。
4. 填写域名。
5. 单击确定。
6. 进行域名归属验证。

说明:

创建域别名成功后，复制弹出的随机验证码，到域名购买平台进行域名解析 TXT 记录设置；设置完毕后，再进行域名归属验证。

更多信息

关于创建域别名及单点登录设置，请参考联合登录概述。

具体的域别名操作方式，请参考云账号的 *SAML* 设置。

2.3 RAM 角色身份

2.3.1 理解 RAM 角色

RAM 角色与 RAM 用户一样，都是 RAM 中定义的身份。RAM 角色是一种虚拟用户，没有确定的身份认证密钥，需要被一个受信的实体用户扮演才能正常使用。

> **说明:**
> 如果没有特别说明，文中出现的角色都是指 RAM 角色。

RAM 角色

RAM 角色（RAM-Role）是一种虚拟用户，它是 RAM 身份类型的一种。

图 2-1: RAM 角色

RAM 角色相关概念

与 RAM 角色相关的概念关系如下图所示：

图 2-2: RAM 角色相关概念

表 2-1: 相关概念的具体释义

名称	释义
RoleARN	RoleARN 是角色的全局资源描述符，用来指定具体角色。 · RoleARN 遵循阿里云 ARN 的命名规范。比如，某个云账号下的 devops 角色的 ARN 为：`acs:ram::1234567890123456:role/samplerole`。 · 创建角色后，单击角色名后，可在 基本信息页查看其 ARN。
受信实体	角色的受信实体是指可以扮演角色的实体用户身份。 · 创建角色时必须指定受信实体，角色只能被受信的实体扮演。 · 受信实体可以是受信的云账号，或者受信服务。
权限策略	一个角色可以绑定一组权限策略。没有绑定权限策略的角色也可以存在，但不能访问资源。
扮演角色	扮演角色（Assume Role）是实体用户获取角色身份的安全令牌的方法。一个实体用户通过调用 AssumeRole 的 API 可以获得角色的安全令牌，使用安全令牌可以访问云服务 API。
切换身份	切换身份（Switch Role）是在控制台中实体用户从当前登录身份切换到角色身份的方法。 · 一个实体用户登录到控制台之后，可以切换到被许可扮演的某一种角色身份，然后以角色身份操作云资源。 · 用户不需要使用角色身份时，可以从角色身份切换回原来的登录身份。
角色令牌	角色令牌是角色身份的一种临时访问密钥。角色身份没有确定的访问密钥，当一个实体用户要使用角色时，必须通过扮演角色来获取对应的角色令牌，然后使用角色令牌来调用阿里云服务 API。

使用须知

RAM 角色必须与一种实体用户身份联合起来才能使用。

图 2-3: 使用 RAM 角色

RAM 角色与教科书式角色的区别

- RAM 角色：RAM 角色作为虚拟用户，它有确定的身份，可以被赋予一组权限策略，但它没有确定的身份认证密钥（登录密码或 AccessKey）。
- 教科书式角色：教科书式角色（Textbook-Role）或传统意义上的角色是指一组权限集合，类似于 RAM 里的权限策略。如果一个用户被赋予了这种角色，也就意味着该用户被赋予了一组权限，可以访问被授权的资源。

虚拟用户与实体用户的区别

- 实体用户：云账号、RAM用户账号、云服务账号作为实体用户，拥有确定的登录密码或 AccessKey。
- 虚拟用户：RAM 角色作为虚拟用户，没有确定的认证密钥。RAM 角色需要被一个受信的实体用户扮演，扮演成功后实体用户将获得 RAM 角色的临时安全令牌，使用这个临时安全令牌就能以角色身份访问被授权的资源。

RAM 角色类型

根据 RAM 受信实体的不同，RAM 支持以下两种类型的角色：

- 阿里云账号：允许 RAM 用户所扮演的角色。扮演角色的 RAM 用户可以属于自己云账号，也可以是属于其他云账号。用户角色主要用来解决跨账号访问和临时授权问题。
- 阿里云服务：允许云服务所扮演的角色。服务角色主要用于授权云服务代理您进行资源操作。

RAM 角色机制的适用场景

- 针对移动设备应用的临时授权

- 跨云账号的资源授权
- 云上应用的动态身份与授权管理

2.3.2 管理 RAM 角色

使用控制台或 API 都可以扮演 RAM 角色，创建 RAM 角色后，通过对角色进行授权可以访问相应的资源。

> 说明:
> 如果没有特别说明，文中出现的角色都是指 RAM 角色。

创建 RAM 角色的前提条件

使用云账号（或拥有 RAM 操作权限的 RAM 用户）登录到 *RAM* 控制台。

创建 RAM 角色

1. 创建可信实体为阿里云账号的角色。
   a. 单击 RAM 角色管理 > 新建 RAM 角色。
   b. 选择可信实体类型为阿里云账号。
   c. 选择受信云账号 ID。
      - 若创建的角色是给自己名下的 RAM 用户使用（例如授权移动 app 客户端直接操作 OSS 资源），请选择当前云账号为受信云账号。
      - 若创建的角色是给其他云账号名下的 RAM 用户使用（例如跨账号的资源授权），请选择其他云账号，并填写其他云账号的 ID。
   d. 输入 RAM 角色名称和备注后，单击确定。

2. 创建可信实体为阿里云服务的角色。

   a. 单击 RAM 角色管理 > 新建 RAM 角色。

   b. 选择可信实体类型为阿里云服务。

   c. 根据需要选择受信服务。可用的服务角色举例：

      - MTS 多媒体转码服务：用于将 OSS Bucket 设置为 MTS 任务的数据源时，创建以 MTS 为受信服务的角色，并使用 MTS 服务扮演该角色访问 OSS 中的数据。
      - OAS 归档存储服务：用于将 OSS Bucket 设置为归档存储服务的数据源时，创建以归档存储为受信服务的角色，并使用归档存储服务扮演该角色访问 OSS 中的数据。
      - LOG 日志服务：用于将日志服务收集的日志导入 OSS 时，创建以日志服务为受信服务的角色，并使用日志服务扮演该角色将数据写入 OSS。
      - ApiGateway API 网关服务：用于将函数服务设置为 API 网关的后端服务时，创建以 API 网关服务为受信服务的角色，并使用 API 网关扮演该角色调用函数服务。
      - ECS 云服务器：用于授权 ECS 服务访问您在其他云服务中的云资源。

      
      说明：
      更多受信服务请以实际界面为准。

   d. 输入 RAM 角色名称和备注后，单击确定。


说明：
在 RAM 角色管理页面找到新创建的角色，单击其 RAM 角色名称可以查看相应的角色详情。

使用 RAM 角色

阿里云服务的角色只能被受信云服务扮演，而阿里云账号的角色可以通过 RAM 用户身份来扮演。

1. 创建一个 RAM 用户，并为该用户创建 AccessKey 或设置登录密码。
2. 给该 RAM 用户授权，授权时添加系统授权策略：AliyunSTSAssumeRoleAccess。


说明：
为了安全起见，阿里云不允许受信云账号以自己身份扮演角色。必须使用 RAM 用户进行角色扮演。

根据使用方式的不同，RAM 用户可以使用控制台或 API 扮演 RAM 角色：

- 使用控制台扮演 RAM 角色

  如果一个实体用户要想使用被赋予的某个 RAM 角色，实体用户必须先以自己身份登录，然后执行切换身份操作将自己从实体身份切换到角色身份。

  1. RAM 用户登录 RAM 控制台。
  2. 将鼠标悬停在右上角头像的位置，单击切换身份。
  3. 进入角色切换的页面，填写相应账号别名，并填写角色名，单击切换。

     说明:

     - 切换成功后，用户将以角色身份访问控制台。此时控制台右上角将显示角色身份（即当前身份）和登录身份。
     - 切换成功后，用户将只能执行该角色身份被授权的所有操作，而登录时实体身份所对应的访问权限被隐藏。
  4. 在扮演角色身份时，选择返回登录身份可以切换回登录身份。

     说明:

     此时将拥有实体身份所对应的访问权限，而不再拥有角色身份所拥有的权限。

- 使用 API 扮演 RAM 角色

  当 RAM 用户被授予 AssumeRole 权限之后，可以使用其 AccessKey 调用安全令牌服务(STS) 的 AssumeRole 接口，以获取某个角色的临时安全令牌，然后使用临时安全令牌访问云资源的 API 接口。

  关于 AssumeRole API 的调用方法，请参考*AssumeRole*。

更多信息

成功创建角色后，角色没有任何权限，单击添加权限可直接为该角色授权。详情请参考*RAM 授权*。

3 身份集成

3.1 联合登录概述

本文为您介绍当企业希望使用自有的身份系统来实现联合登录时，通过 RAM 所提供的联合登录解决方案，帮助用户快速理解和配置联合登录，并满足企业安全及合规性要求。

联合登录的前提条件

阿里云支持企业级 IdPs （identity providers）广泛使用的 *SAML 2.0 #Security Assertion Markup Language 2.0#* 身份联合开放标准。通过开启联合登录，用户可以使用组织内的账号认证机制登录到阿里云控制台。

下面简要介绍与 SAML 联合登录相关的一些基本术语。

表 3-1: 基本术语

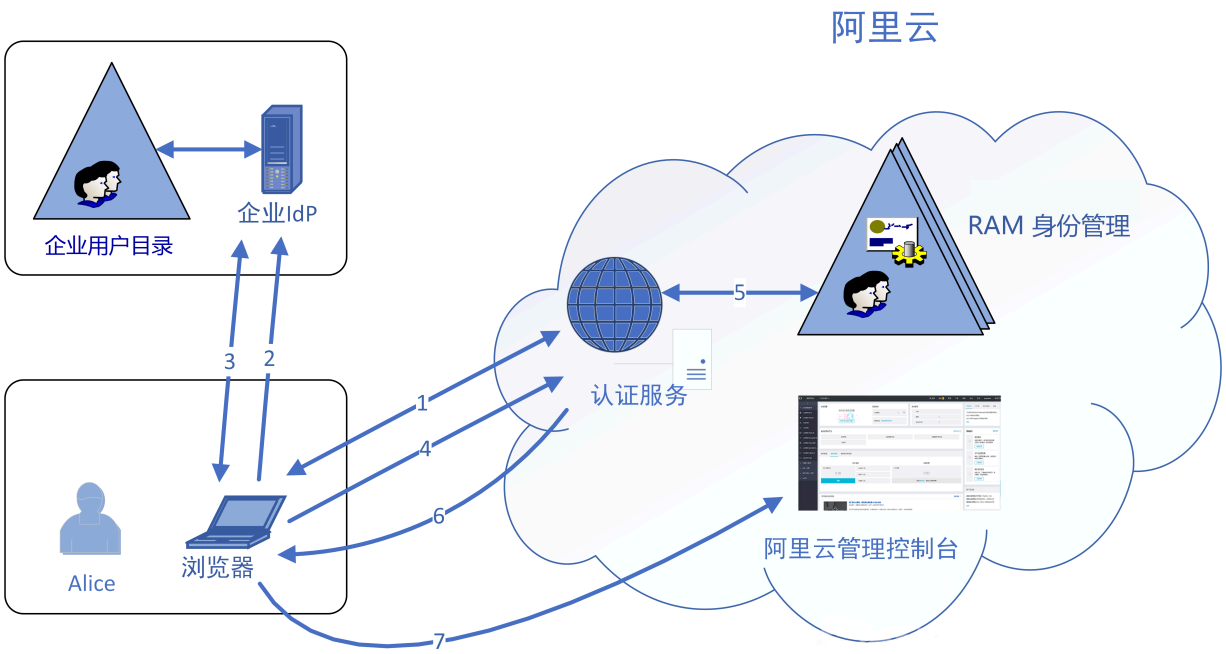
术语	说明
IdP（Identity Provider） 身份提供商	提供身份管理的服务。例如： · 企业本地 IdP： Microsoft Active Directory Federation Service （AD FS ）、Shibboleth 等。 · Cloud IdP：Azure AD，Google G Suite ，Okta，OneLogin 等。
SP（Service Provider） 服务提供商	利用 IdP 的身份管理功能，为用户提供具体服务的应用，SP 会使用 IdP 提供的用户信息。一些非 SAML 协议的身份系统（如 OpenID Connect），也把 Service Provider 称作 Relying Party，也就是 IdP 的依赖方。
SAML（Security Assertion Markup Language） 安全断言标记语言	实现企业级用户身份认证的标准协议，它是 SP 和 IdP 之间实现沟通的技术实现方式之一。SAML 2.0 已经是目前实现企业身份联合（Identity Federation）的一种事实标准。
SAML Assertion SAML 断言	SAML 协议中用来描述认证请求（Request ）和认证响应（Response）的核心元素。例如：用户的具体属性就包含在认证响应的断言里。

术语	说明
Trust 信赖	建立在 SP 和 IdP 之间的互信机制，通常由公钥和私钥来实现。SP 通过可信的方式获取 IdP 的身份联合元数据，元数据中包括了 IdP 签发SAML Assertion 的签名验证公钥，SP 则可以使用公钥来验证 Assertion 的完整性（Integrity）。

SAML 联合登录的基本思路

阿里云与外部企业身份系统的集成场景中，阿里云是服务提供商（SP），而企业自有的身份服务则是身份提供商（IdP）。企业员工通过企业自有身份服务登录到阿里云控制台的基本流程如下图所示：

图 3-1: 基本流程



当管理员在完成 SAML 联合登录的配置后，企业员工可以通过如图所示的方法登录到阿里云控制台：

1. 企业员工 Alice 使用浏览器登录阿里云，阿里云将 SAML 认证请求返回给浏览器。
2. 浏览器向企业 IdP 转发 SAML 认证请求。
3. 企业 IdP 提示企业用户登录，并且在用户登录成功后生成 SAML 响应返回给浏览器。
4. 浏览器将 SAML 响应转发给阿里云。
5. 阿里云通过 SAML 互信配置，验证 SAML 响应的数字签名以验证 SAML 断言的真伪，并通过 SAML 断言的用户名称，匹配到对应云账号中的 RAM 用户身份。

6. 登录服务完成认证，阿里云向浏览器返回登录 session 以及阿里云控制台的 URL。
7. 浏览器重定向到阿里云管理控制台。

> 说明:
> 在第 1 步中，企业员工从阿里云发起登录并不是必须的。企业员工也可以在企业自有 IdP 的登录页直接点击登录到阿里云的链接，向企业 IdP 发出登录到阿里云的 SAML 认证请求。

SAML 联合登录的配置步骤

为实现阿里云账号与企业 IdP 之间建立相互信任，需要进行外部 IdP 的 SAML 配置和阿里云账号作为 SP 的 SAML 配置，才能完成联合登录。

1. 外部 IdP 的 SAML 配置。

   为了建立企业 IdP 对阿里云账号的信任，需要在外部 IdP 中配置阿里云账号为可信 SAML SP。

2. 云账号的 SAML 配置。

   为了建立阿里云对企业 IdP 的信任，需要将企业 IdP 配置到阿里云。

   > 说明:
   > 配置时需要使用企业 IdP 提供的 SAML 元数据。元数据包含了 IdP 服务地址、用于验证签名的公钥以及断言格式等信息。常见的 IdP 服务都提供了特定的地址下载 SAML元数据。

3. 用户配置。

   在 SP 和 IdP 之间建立互信后，还需要在阿里云 RAM 中创建一个或多个与企业 IdP 可以匹配的用户。只有 RAM 用户才能使用 SSO 认证。

操作结果

配置完成后，即可实现 RAM 用户与企业 IdP 的联合登录。当云账号下的 RAM 用户登录时，阿里云会自动跳转到您的企业 IdP 登录服务地址，成功登录后会自动跳转到阿里云控制台。

3.2 用户联合登录

3.2.1 外部 IdP 的 SAML 配置

阿里云 SAML 联合登录涉及的配置操作包括外部 IdP 的 SAML 配置和阿里云账号作为 SP 的SAML配置，以建立外部 IdP 与阿里云账号之间的信任关系。本文主要介绍外部 IdP 的 SAML 配置流程。

一、配置阿里云账号为可信 SAML SP

1. 从阿里云获取您的云账号的 SAML 服务提供方元数据 URL。
   a. 登录 *RAM* 控制台。
   b. 单击人员管理 > 设置 > 高级设置，在SSO 登录设置下可以查看当前云账号的SAML 服务提供方元数据 URL。
2. 在外部 IdP 中创建一个 SAML SP，并配置阿里云的 SAML 服务提供方元数据 URL。

> 说明:
> 如果您的 IdP 不支持 URL 配置，那么您可以从SAML 服务提供方元数据 URL下载 XML 文件，并在创建 SAML SP 的相应流程中上传此 XML文件。

二、在 IdP 中定制 SAML 断言

阿里云需要通过 User Principal Name (UPN)来定位一个 RAM 用户，所以要求外部 IdP 生成的 SAML 回应里包含用户的 UPN。阿里云通过解析 SAML 断言中的 NameID 节点，来匹配 RAM 用户的 UPN 从而实现用户的联合登录。

所以，在定制 IdP 颁发的 SAML 断言时，需要将对应于 RAM 用户 UPN 的字段映射为 SAML 断言中的 NameID。

示例

由于不同 IdP 的系统差异，关于 SAML SP 配置和断言定制的操作流程都有些差异。在本章内容中，我们会提供一个以 Microsoft Active Directory 与阿里云进行联合 SSO 集成的企业 *IdP* 与阿里云身份联合示例，用于帮助客户理解企业 IdP 与阿里云身份联合的端到端配置流程。

3.2.2 云账号的 SAML 配置

本文介绍阿里云账号的 SAML 配置（SP 侧的配置），包括默认域名，域别名以及 SAML 单点登录（SSO）的设置方法。

云账号的默认域名

每个云账号会有一个默认域名，可在 *RAM* 控制台，单击人员管理 > 设置 > 高级设置，进行查询和修改。

- 默认域名的格式为：“<$AccountAlias>.onaliyun.com”。
- 若未设置过云账号别名，那么云账号别名默认值是 AccountID，此时默认域名的格式为：“<$AccountID>.onaliyun.com”。

为默认域名设置域别名

使用域别名可以简化客户配置 SAML 单点登录的流程。

在 RAM 用户域名管理中，您可以为默认域名创建一个域别名。域别名通常可以设置为企业的真实域名，方便用户使用。

假设您的默认域名是：secloud.onaliyun.com，对应的域别名是：secloud.net，那么 RAM 用户使用alice@secloud.onaliyun.com 或 alice@secloud.net 都可以进行登录，这两种表示都代表同一个 RAM 用户。

创建域别名的方法

1. 在 RAM 控制台首页，单击人员管理 > 设置 > 高级设置。
2. 单击创建域别名。
3. 填写域名。
4. 单击确定。
5. 进行域名归属验证。

> 说明：
> 创建域别名成功后，复制弹出的随机验证码，到域名购买平台进行域名解析 TXT 记录设置；设置完毕后，再进行域名归属验证。

SAML 单点登录(SSO)的设置

1. 在 RAM 控制台，单击人员管理 > 设置 > 高级设置，可以查看当前 SSO 登录设置的状态。

2. 通过编辑 SSO 登录设置进入设置页。

   · 元数据文件：元数据文件是由外部身份提供商(IdP)提供的元数据文档，一般为 XML 格式，它包括 IdP 的登录服务地址以及 X.509 公钥证书（用于验证 IdP 所颁发的 SAML Assertion 的有效性）。这里需要用户上传由外部 IdP 提供的元数据文档。
   · SSO 功能状态：可以选择开启或关闭。
     - 此功能默认为关闭，此时 RAM 用户可以使用密码登录方式登录，所有 SSO 设置不生效。
     - 如果选择开启此功能，RAM 用户密码登录方式将会被关闭，统一跳转到外部 IdP 登录服务进行身份认证。如果再次关闭，用户密码登录方式自动恢复。

    说明:
   该功能只对云账号下的所有 RAM 用户生效，不会影响云账号的登录。

3.2.3 企业 IdP 与阿里云身份联合示例

本文提供一个以 Microsoft Active Directory 与阿里云进行 SSO 集成的示例，用于帮助用户理解企业 IdP 与阿里云身份联合的端到端配置流程。

前提条件

用户对 Microsoft AD 需进行合理正确的配置，在 Windows Server 2012 R2 上配置以下服务器角色 （Server Role）：

· DNS 服务器：将身份认证请求解析到正确的 Federation Service 上。
· Active Directory 域服务 （AD DS）：提供对域用户和域设备等对象的创建，查询和修改等功能。
· Active Directory Federation Service （AD FS）：提供配置身份联合信赖方的功能，并对配置好的信赖方提供单点登录认证。

注意事项

本文以 Windows Server 2012 R2 为例，介绍如何配置 Microsoft AD 作为阿里云的单点登录 IdP。

 注意:
本文中涉及到 Microsoft Active Directory 配置的部分属于建议，仅用于帮助理解阿里云身份联合的端到端配置，阿里云不提供 Microsoft Active Directory 配置官方咨询服务。

示例配置

示例中用到相关配置如下：

- 云账号的默认域名为：`secloud.onaliyun.com`
- 云账号下包含 RAM 用户：`alice`，其完整的 UPN（User Principal Name）为：`alice@secloud.onaliyun.com`
- 自建 Microsoft AD 中的 AD FS 服务名称为：`adfs.secloud.club`
- 自建 Microsoft AD 的域名为：`secloud.club`，NETBIOS 名为：`secloud`
- 用户 alice 在 AD 中的 UPN 为：`alice@secloud.club`，域内登录也可以使用：`secloud\alice`

在 RAM 中将 AD FS 配置为可信 SAML IdP

1. 在浏览器中输入如下地址：

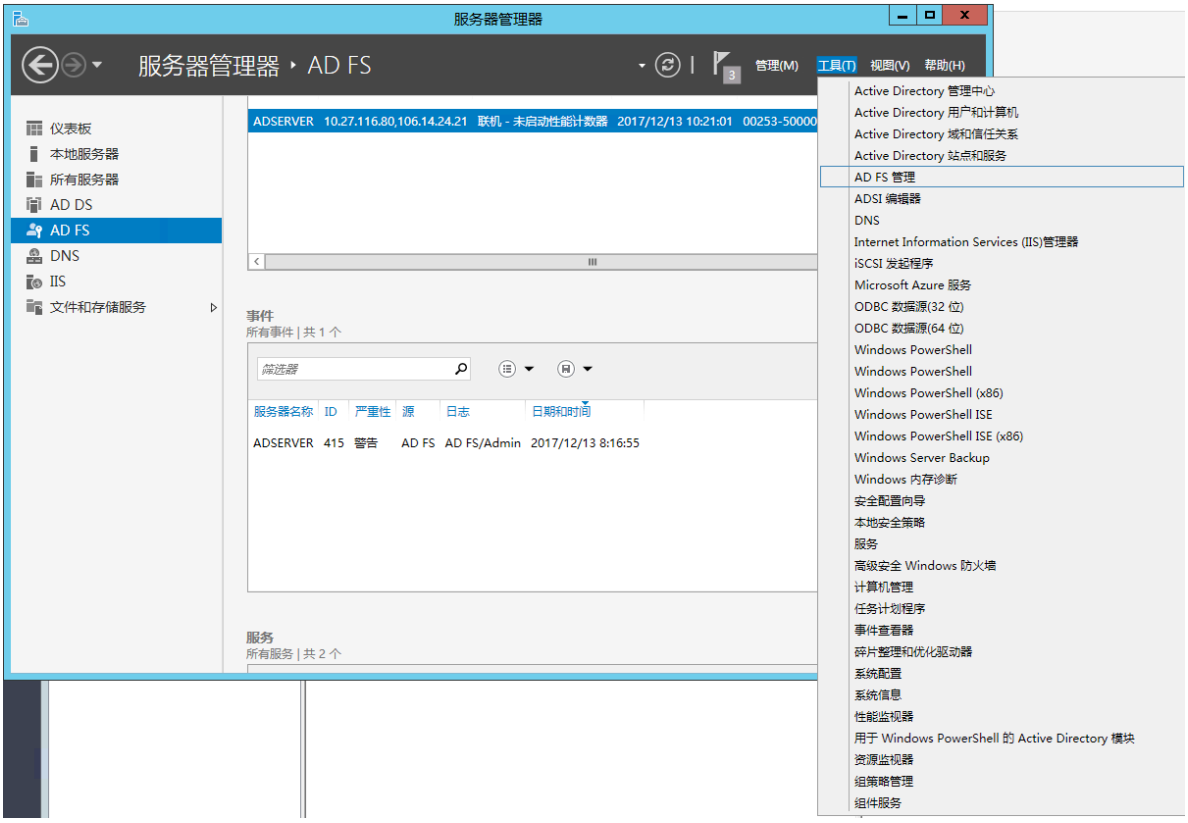
```
https://adfs.secloud.club/FederationMetadata/2007-06/FederationMetadata.xml
```

2. 将元数据 XML 文件下载到本地。
3. 在 RAM 控制台的 SSO 配置时使用下载好的元数据文件。

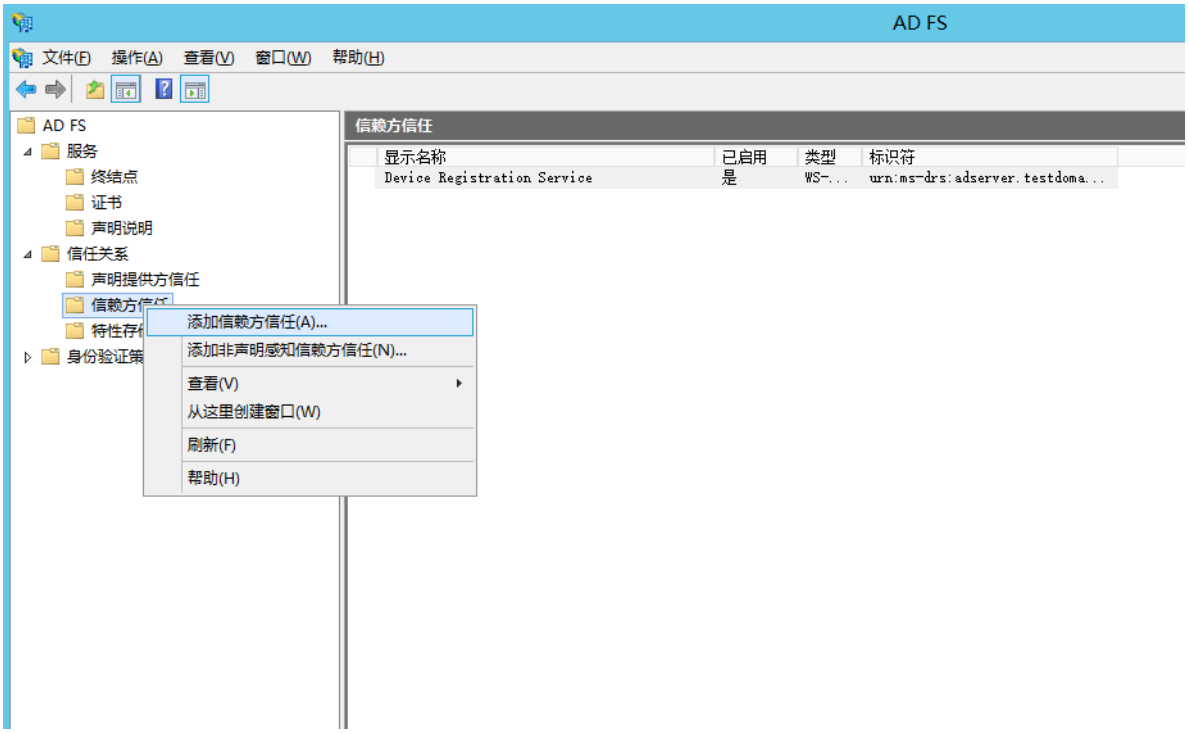
在 AD FS 中将阿里云配置为可信 SAML SP

在 AD FS 中，SAML SP 被称作信赖方（Relying Party）。设置阿里云作为 AD FS 的可信 SP 的操作步骤如下：

1. 在服务器管理器的工具菜单中选择AD FS 管理 。

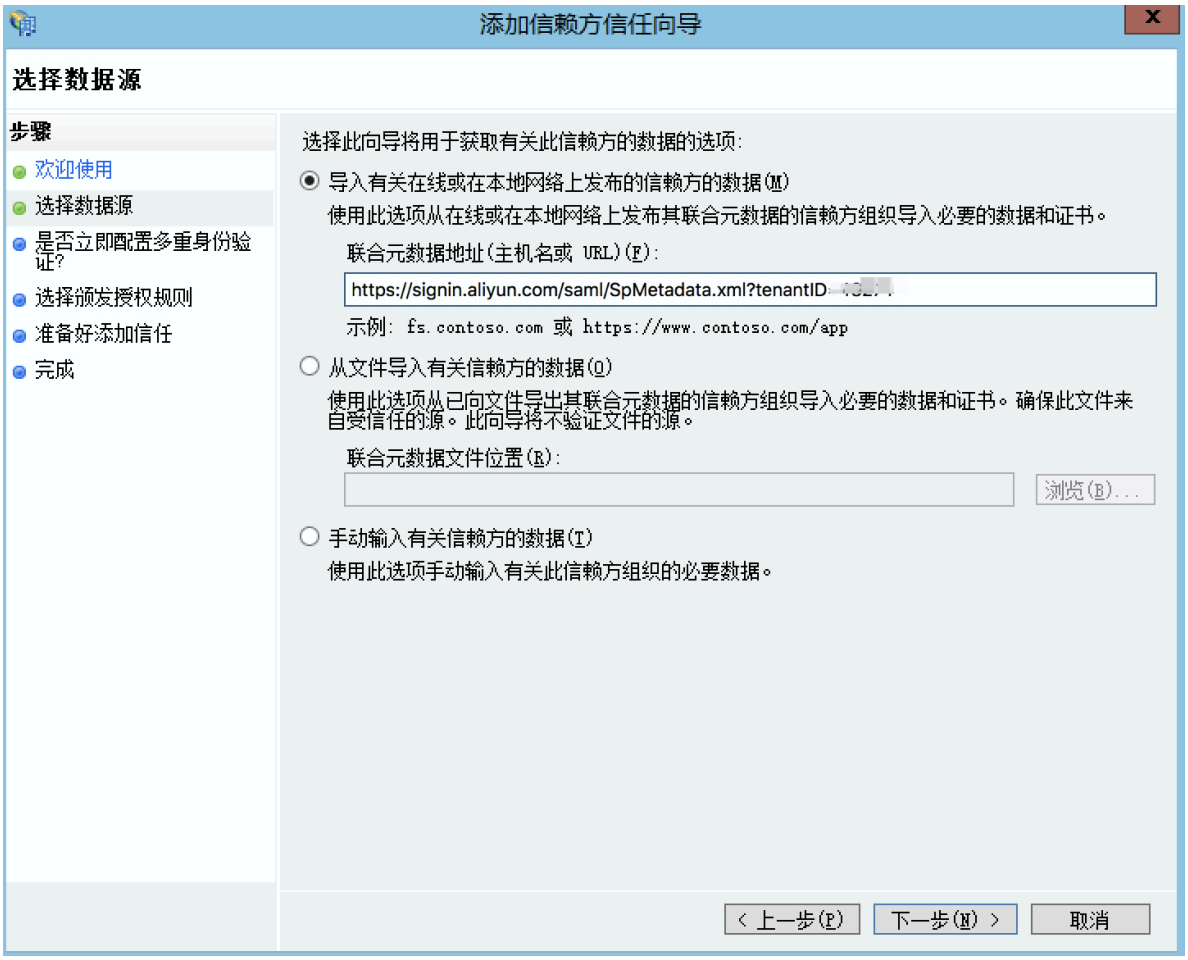


2. 在 AD FS 管理工具中添加信赖方信任（Relying Party Trust）。



3. 为新创建的信赖方设置阿里云的 SAML 元数据。

   阿里云账号的 SAML 元数据 URL 可以在 RAM 控制台页面，单击人员管理 > 设置 > 高级设置中查看。AD FS 信赖方可以直接配置元数据的 URL。



   完成配置信赖方之后，阿里云和 AD FS 就产生了互信，阿里云会将`secloud.onaliyun.com`域名的 RAM用户认证请求转发到 AD FS `adfs.secloud.club` 上，AD FS 也会接受来自于阿里云的认证请求并向阿里云转发认证响应。

为阿里云 SP 配置 SAML 断言属性

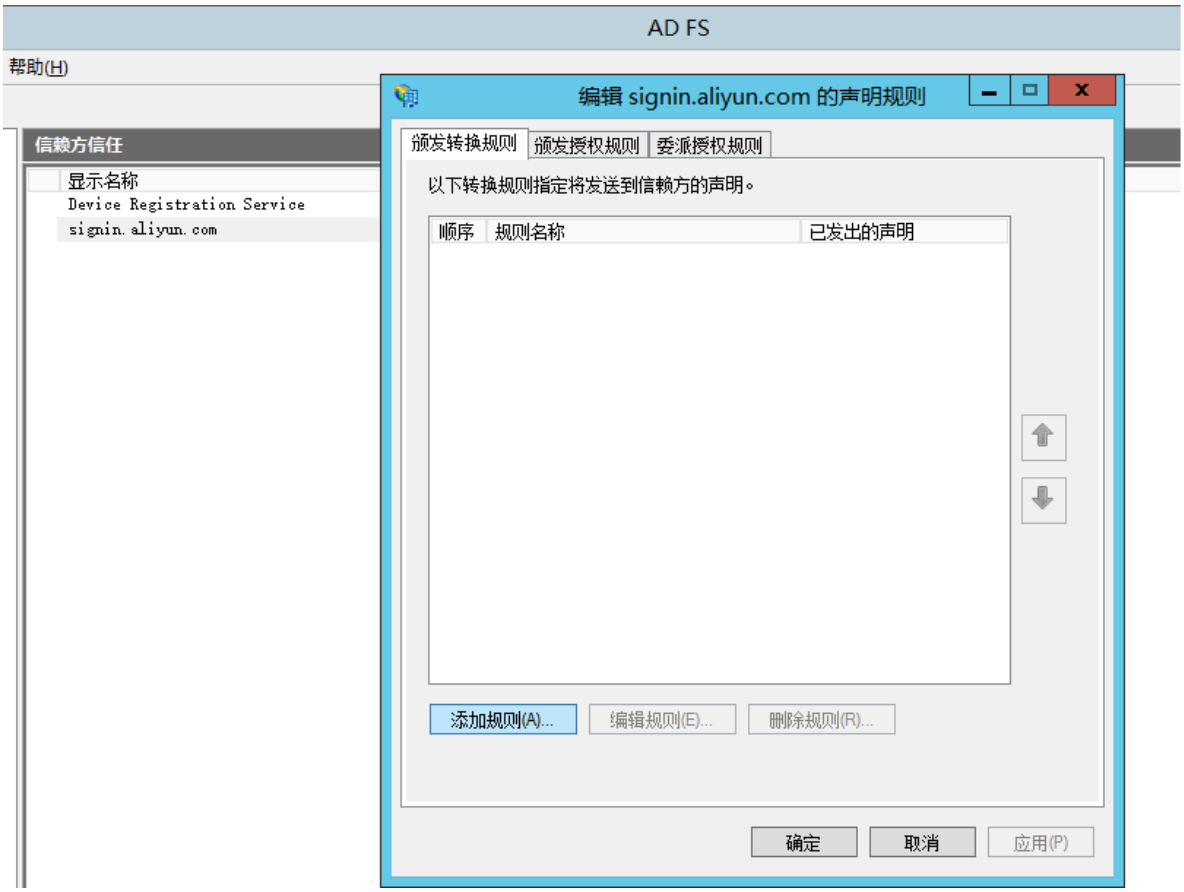
为了让阿里云能使用 SAML 响应定位到正确的 RAM 用户，SAML 断言中的 NameID 字段取值应为 RAM 用户的 UPN。

我们需要配置 Active Directory 中的 UPN 为 SAML 断言中的 NameID，其操作步骤如下：

1. 为信赖方编辑声明规则。

2. 添加颁发转换规则。

> **说明:**
> 颁发转换规则（Issuance Transform Rules）：指如何将一个已知的用户属性，经过转换后，颁发为 SAML 断言中的属性。由于我们要将用户在 AD 中的 UPN 颁发为 NameID，因此需要添加一个新的规则。



3. 声明规则模版选择转换传入声明（Transform an Incoming Claim）。

4. 编辑规则。

说明:

由于示例中的云账号里的 UPN 域名为`secloud.onaliyun.com`，而 AD 中的 UPN 域名为`secloud.club`，如果直接将 AD 中的 UPN 映射为 NameID，阿里云将无法匹配到正确的 RAM 用户。

下面提供两种方法来设置 RAM 用户的 UPN 与 AD 用户的 UPN 将保持一致：

a. 方法一：将 AD 域名设置为 RAM 的域别名。

前提条件：

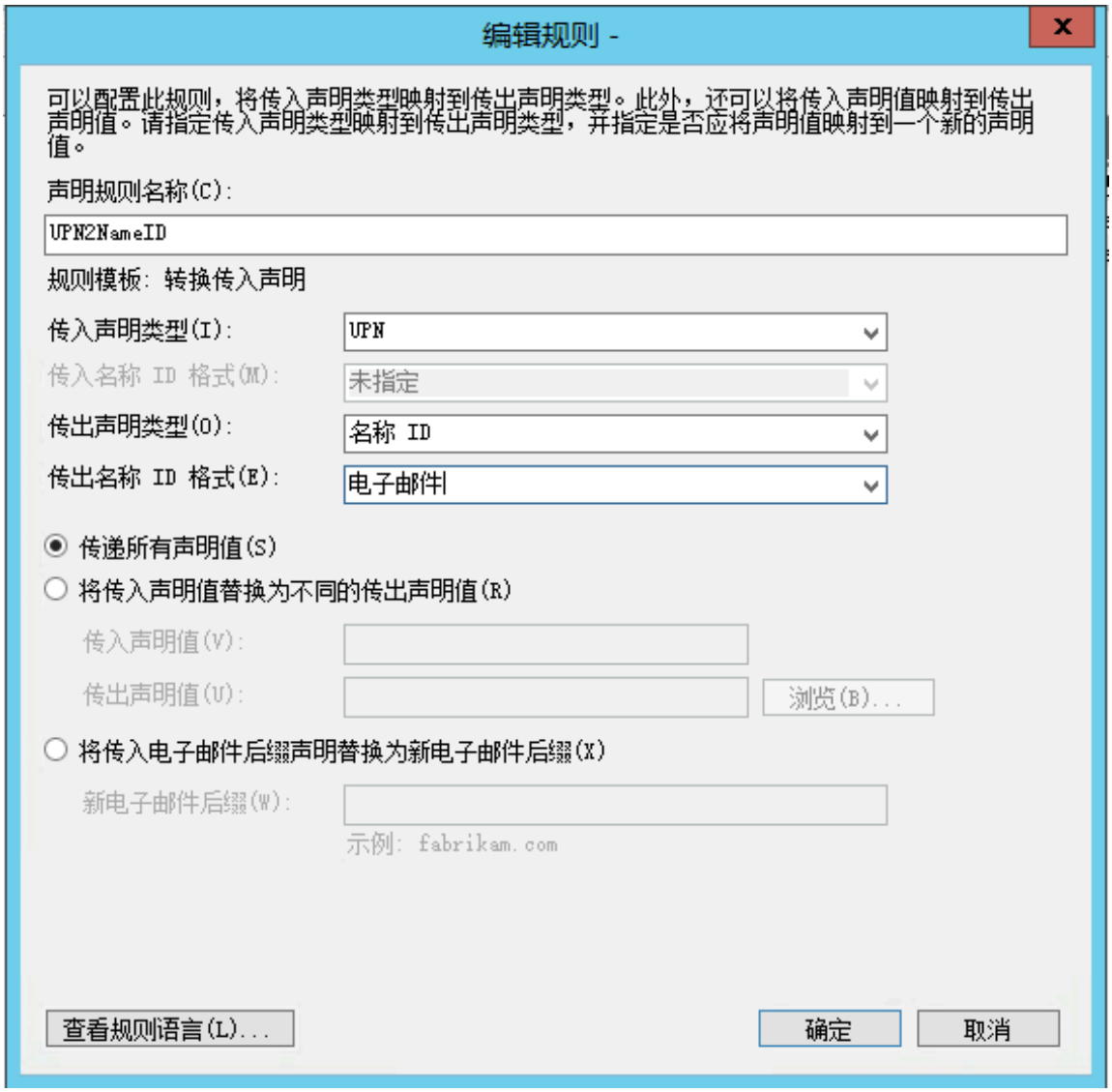
如果 AD 域名`secloud.club`是一个在公网 DNS 中注册的域名。

解决方案：

用户可以将`secloud.club`设置为 RAM 域别名。

详情请参考域别名及 SSO 设置。

完成设置后，在编辑规则窗口，将 UPN 映射为名称ID（NameID）。



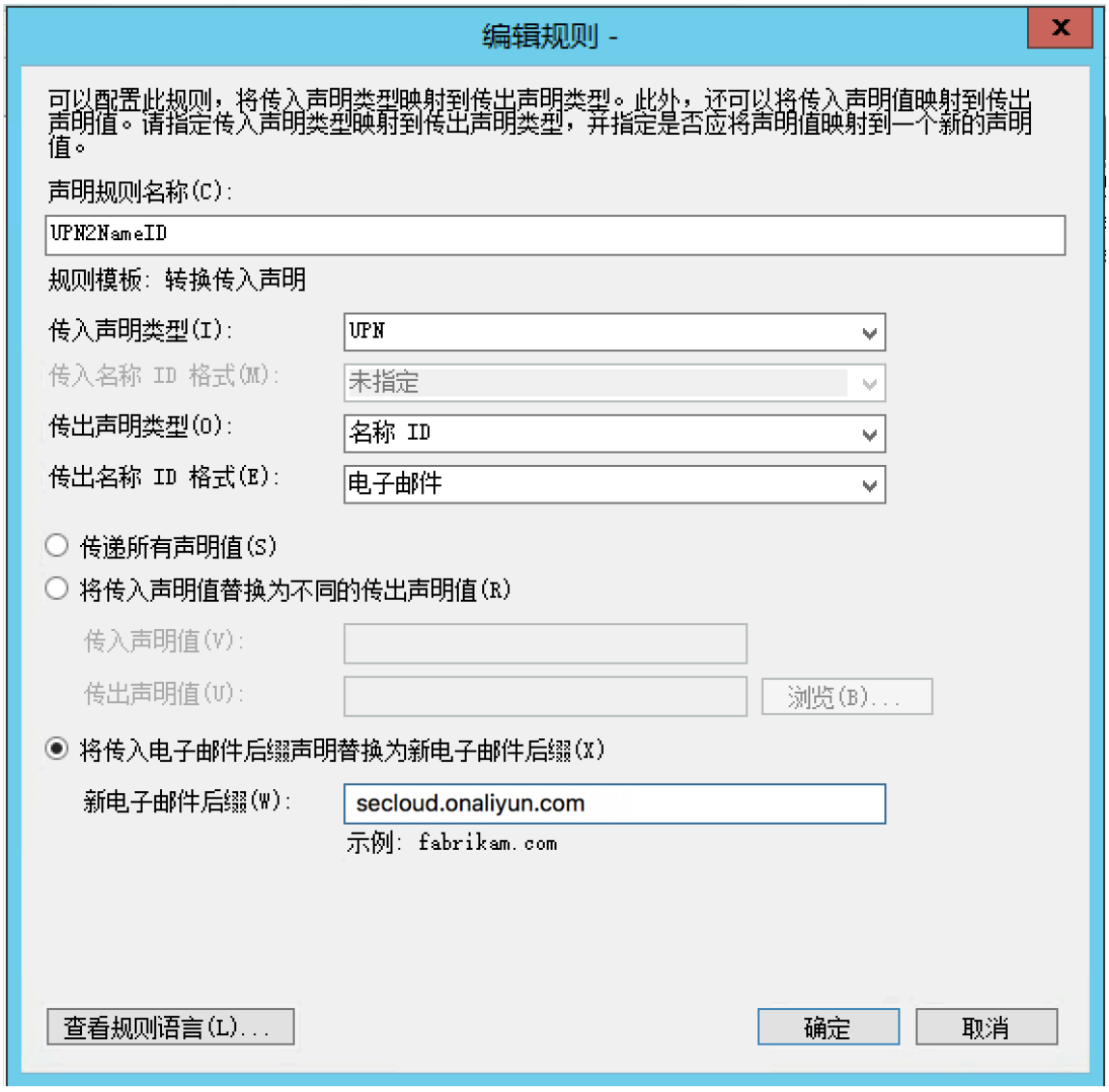
b. 方法二：在 AD FS 中设置域名转换。

前提条件：

如果域名`secloud.club`是企业的内网域名，那么阿里云将无法验证企业对域名的所有权。RAM 就只能采用默认域名`secloud.onaliyun.com`。

解决方案：

在 AD FS 给阿里云颁发的 SAML 断言中必须将 UPN 的域名后缀从`secloud.club`替换为`secloud.onaliyun.com`。

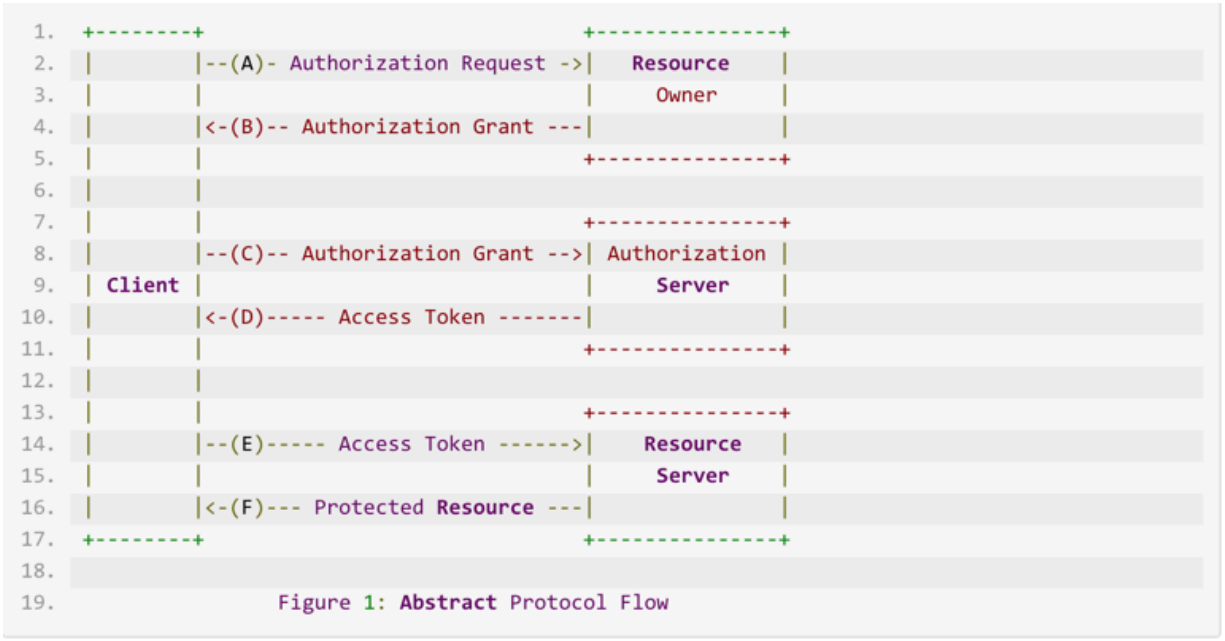


3.3 OAuth应用管理

3.3.1 OAuth 应用开发概述

RAM 支持使用 *OAuth 2.0 协议*进行用户认证和应用授权。RAM 支持常见的 OAuth 2.0 场景，比如RAM用户登录到网络应用或本地应用后，应用可以代表RAM用户去调用阿里云 API；或者通过 OIDC (OpenID Connect) 支持RAM用户登录到企业自有应用。本文介绍OAuth 2.0的基本概念、阿里云支持的OAuth scope以及控制台操作方式等基础信息。不同的OAuth使用场景详见“场景与协议”章节。

概述



OAuth 2.0 包含一系列不同的流程，用来解决不同应用特征带来的具体问题，典型的流程遵循一种通用的模式来实现应用对用户的扮演（User Impersonation）。其经典流程如下：

1. 应用获取用户授权扮演用户。

   应用向 Authorization Server 发起授权请求。这一过程中通常要求用户通过浏览器登录，在某些情况下还需要用户同意Authorization Server 向应用授权。用户授权的结果通常是授权码（authorization code），并由 Authorization Server 返回给应用。

2. 应用获取代表用户身份的令牌，完成用户扮演。

   应用拿到扮演用户的授权之后，可以在一定时间内向 Authorization Server 换取代表用户身份的令牌。如果是 OIDC 场景，令牌只包含身份令牌（ID Token），如果是需要访问阿里云 API 的场景，令牌还会包含访问令牌（Access Token）。

3. 应用使用代表用户身份的访问令牌访问云 API。

   应用拿到访问令牌之后，可以发起访问阿里云 API 的请求，由于令牌代表用户身份，因此应用可以访问当前用户的资源。

概念解析

下表描述了 OAuth 2.0 协议中的角色主体以及各个角色的功能描述。

角色名称	角色描述	角色功能
Resource Owner	用户	这里指 RAM 子用户，而不是阿里云主账号。RAM 子用户登录到阿里云并授权应用访问资源。 说明: 在经典的 OAuth 模型里，更多是面向个人用户场景，Resource Owner 一般指独立个人用户。但在阿里云的账号结构模型里，主账号代表企业用户，代表 Resource Owner，而 RAM 子用户代表主账号下的成员，主账号可以将资源管理操作代理给其成员账号（即 RAM 子用户）。
Resource Server	阿里云 API	被应用访问。
Authorization Server	阿里云的登录服务	对用户进行认证，并接受用户对应用的授权，生成代表用户身份的访问令牌并返回给被授权的应用
Client	应用（客户端）	获取用户授权，并获取代表用户身份的访问令牌，从而可以访问阿里云 API。
Scope	OAuth 范围	应用扮演登录用户之后可以访问的资源范畴。

从以上描述中可以看到，真正完成对 Resource Server 访问的不是 Resource Owner 本身，而是被 Resource Owner 授权的Client。在这种访问模式下，我们通常将应用获取代表用户身份的访问令牌这一过程叫做扮演用户 (User Impersonation)。

身份令牌与访问令牌

OAuth 服务可以给应用签发代表登录用户的令牌包含：

- 身份令牌：身份令牌只包含用户的身份信息，不能用于访问阿里云服务。
- 访问令牌：访问令牌包含了用户的身份信息以及应用的 OAuth 范围，可以用于访问 OAuth 范围内的阿里云服务。

应用类型

企业开发人员可以在阿里云 RAM 服务内注册 OAuth 应用。目前支持的应用类型包括：

- WebApp：指基于浏览器交互的网络应用。应用通过浏览器交互，获得用户授权访问用户的资源。用户资源可以是应用本身管理的用户资源，也可以是其他服务通过 API 暴露的用户资源。
- NativeApp：指操作系统中运行的本地应用。主要为运行在桌面操作系统，移动操作系统中的应用类型。这类应用获取用户授权访问的用户资源通常为其他服务通过 API 暴露的用户资源。

OAuth 范围

OAuth 2.0 框架通过 OAuth 范围（Scope）来控制应用扮演用户之后对资源服务的访问。应用注册的时候需要指定应用所需要的范围，应用运行时扮演用户后，访问的资源不能超过注册的范围。目前支持的范围包括：

名称	描述
openid	Open ID Connect 1.0 的默认 Scope。所有的应用默认具有这一范围，不需要额外注册。这一范围允许应用获取用户的身份令牌（ID Token））
aliuid	阿里云颁发的唯一用户标志符（ID Token）
profile	用户的名称等个人信息（ID Token）
/acs/ccc	阿里云呼叫中心服务 API
/acs/alidns	阿里云解析 API

说明:

除了`openid, aliuid, profile`这几个范围之外，所有其他范围都是与访问令牌（Access Token）关联，与身份令牌（ID Token）无关。

支持的协议场景

- *Web* 应用
- 桌面 / 移动应用

- *OIDC* 登录场景

控制台使用

1. 登录 *RAM* 控制台。
2. 单击 OAuth 应用管理 > 新建应用，可以创建 OAuth 应用并获取/编辑 OAuth 应用的 Scope，Secret，Token 有效期等信息。

3.3.2 Web 应用场景

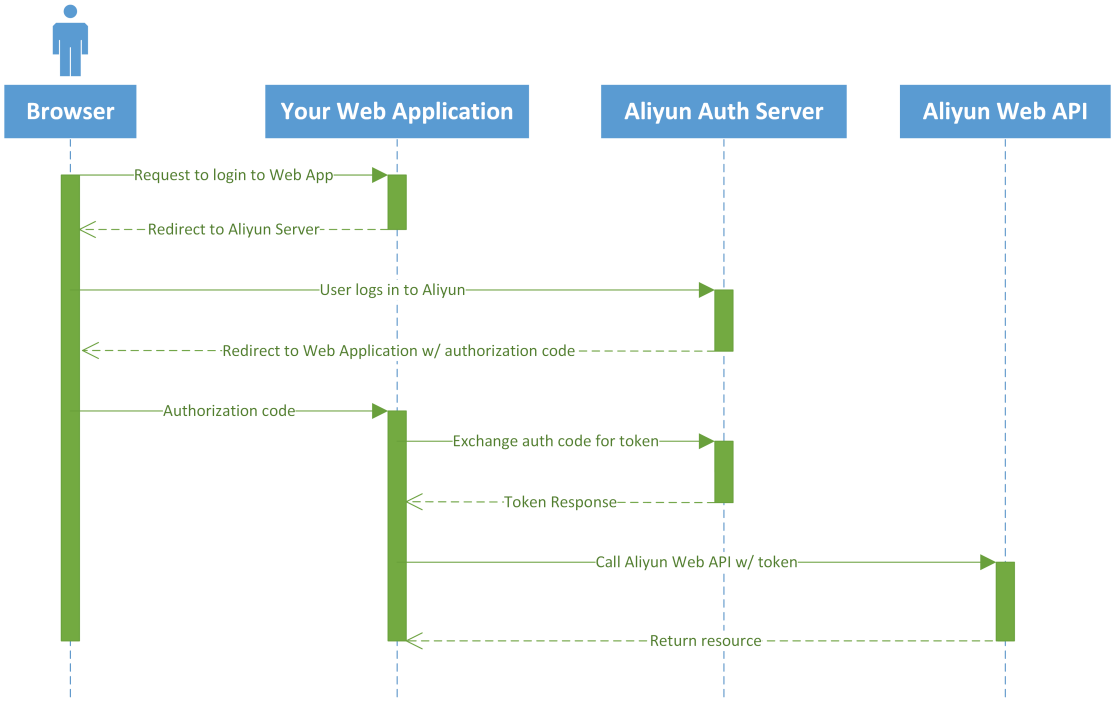
本文介绍 Web 应用如何通过 OAuth 2.0 扮演登录用户访问阿里云 API。

应用的注册

Web 应用扮演登录用户访问 Open API 是标准流程之一。应用开发方需要先在自己的云账号内注册一个 Web 应用，并且提供恰当的名称，访问的 API 范围（OAuth Scope），应用的回调地址等关键信息。并且需要为应用生成应用密码（App Secret）。

注册应用之后，可以在云账号内直接扮演用户。如果要扮演其他云账号的用户，则需要获得其他云账号的授权。

总体流程



获取访问令牌

1. 获取授权码（authorization code）

   获取访问令牌的第一步是要获取用户授权码。应用需要通过浏览器将用户重定向到阿里云 OAuth 2.0 服务。

   授权码发放 Endpoint：*https://signin.aliyun.com/oauth2/v1/auth*。

   接受以下 Query String 参数：

参数名	要求	描述
client_id	Required	应用的 Identifier。
redirect_uri	Required	应用注册的重定向 URI 之一。
response_type	Required	返回类型。根据 OAuth 2.0 标准，目前支持设置此参数的取值为`code`。
scope	Optional	空格分隔的 OAuth 范围列表。如不指定此参数取值，则默认为应用注册的全部 OAuth 范围，加上`openid`。
access_type	Optional	应用的访问类型。取值分为`online`（在线访问）和`offline`（离线访问）两种。Auth Server 针对离线访问类型的请求，会发放 refresh token，应用可以保持 refresh token，并根据需求持续刷新访问令牌。如果应用的使用场景不需要离线刷新访问令牌，则可以设置访问类型为`online`。默认取值为`online`。

参数名	要求	描述
state	Optional	设置为任意字符串，Auth Server 会将请求中的`state`参数以及取值放到返回中。应用通过`state`参数可以实现各种目的，比如状态保持，发送 nonce 从而减少 CSRF 威胁等。

请求样例：

```
https://signin.aliyun.com/oauth2/v1/auth?
client_id=123456&
redirect_uri=https%3A%2F%2Fyourwebapp.com%2Fauthcallback%2F&
response_type=code&
scope=openid%20%2Facs%2Fccc&
access_type=offline&
state=1234567890
```

响应样例：

```
GET HTTP/1.1 302 Found
Location: https://yourwebapp.com/authcallback/?code=ABAFDGDFXYZW888&
state=1234567890
```

2. 用授权码换取访问令牌

   一旦获得授权码，应用直接调用如下 Endpoint 换取访问令牌：*https://oauth.aliyun.com/v1/token*。

   接受以下 Query String 或 Form Post 参数：

参数名	要求	描述
code	Required	初始请求中获取的授权码。
client_id	Required	应用的 Identifier。
redirect_uri	Required	初始请求中设置的 redirect_uri参数。
grant_type	Required	根据 OAuth 2.0 标准， 取值为`authorization_code`。

参数名	要求	描述
client_secret	Optional	应用的 AppSecret，用作换取访问令牌时鉴定应用身份的密码。

请求样例：

```
POST /v1/token HTTP/1.1
Host: oauth.aliyun.com
Content-Type: application/x-www-form-urlencoded
code=ABAFDGDFXYZW888&
client_id=123456&
client_secret=`your_client_secret`&
redirect_uri=https://yourwebapp.com/authcallback/&
grant_type=authorization_code
```

响应中包含以下字段：

参数名	描述
access_token	代表用户身份的访问令牌，应用使用此访问令牌来访问阿里云 Open API。
expires_in	访问令牌的剩余有效时间，单位为秒。
token_type	访问令牌的类型，为`Bearer`，意味着客户端不要理解 access_token 的内容含义，只需直接使用即可。
id_token	如果初始请求的 scope 参数包含了`openid`，则返回 id_token。id_token 为 OAuth 服务签名的 JWT (Json Web Token)。
refresh_token	如果初始请求的 access_type 为`offline`，则返回 refresh_token。

响应样例：

```
{
  "access_token": "eyJraWQiOiJrMTIzNCIsImVuY...",
  "token_type": "Bearer",
  "expires_in": 3600,
  "refresh_token": "Ccx63VVeTn2dxV7ovXXfLtAqLLERAH1Bc",
  "id_token": "eyJhbGciOiJIUzI1N..."
}
```

刷新访问令牌 (Refresh Token)

刷新访问令牌 Endpoint： *https://oauth.aliyun.com/v1/token*。

刷新访问令牌需要的参数如下：

参数名	要求	描述
refresh_token	Required	用授权码换取 token 时获取到的 refresh_token。
client_id	Required	应用的 Identifier。
grant_type	Required	根据 OAuth 2.0 标准， 取值为 `refresh_token`。
client_secret	Optional	应用的 AppSecret，用作换取访问令牌时鉴定应用身份的密码。

请求样例：

```
POST /v1/token HTTP/1.1
Host: oauth.aliyun.com
Content-Type: application/x-www-form-urlencoded
refresh_token=Ccx63VVeTn2dxV7ovXXfLtAqLLERAH1Bc&
client_id=123456&
client_secret=`your_client_secret`&
grant_type=refresh_token
```

请求的返回值与用授权码换取访问令牌的返回值一致，但不包含 refresh_token 和 id_token。

令牌撤销

撤销令牌的 Endpoint：*https://oauth.aliyun.com/v1/revoke*。

如果应用获取了refresh_token，在特定的场景下也需要撤销 refresh_token，例如用户退出登录应用，或者用户将自己的账号从应用中移除等。撤销令牌需要的参数如下：

参数名	要求	描述
token	Required	需要撤销的 token
client_id	Required	应用的 Identifier
client_secret	Optional	应用的 AppSecret

3.3.3 桌面 / 移动应用场景

本文介绍桌面和移动端Native应用如何通过 OAuth 2.0 扮演登录用户访问阿里云 API。

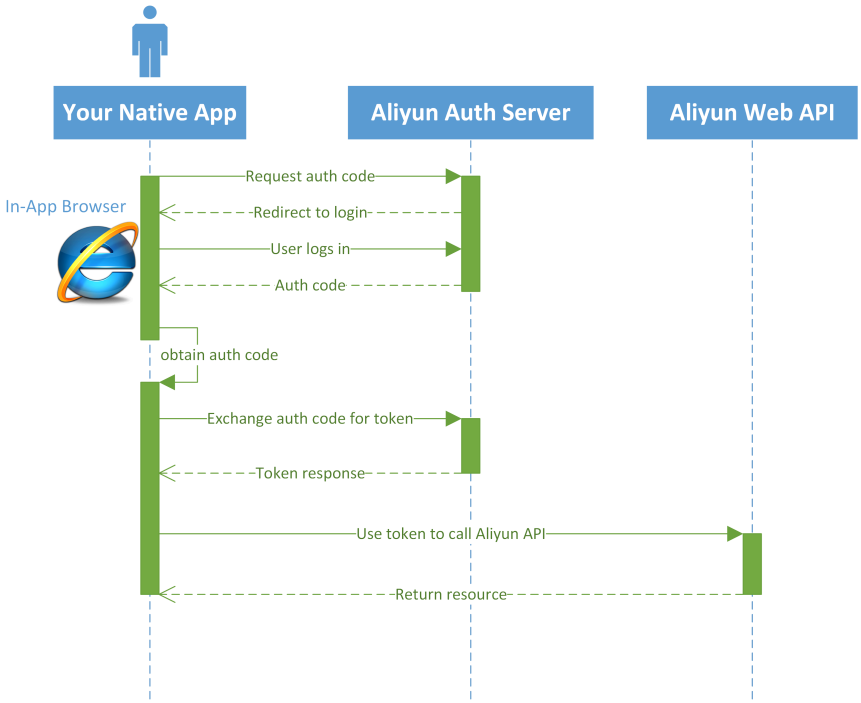
应用的注册

Native 应用扮演登录用户访问 Open API 是标准流程之一。应用开发方需要先在自己的云账号内注册一个 Native 应用，并且提供恰当的名称、访问的 API 范围（OAuth Scope）、应用的回调地

址等关键信息。由于 Native 应用运行在非可信环境，无法有效保护 App Secret，因此 Native 应用不使用 App Secret。

注册应用之后，可以在云账号内直接扮演用户。

总体流程



获取访问令牌

1. 生成Code Verifier和Code Challenge

   OAuth 2.0 对 Native 应用的场景增加了对 *Proof Key* 机制的支持，用于每次获取授权码以及用授权码换取 Access Token。这一机制可以减轻针对授权码截获的攻击。这一机制对于 Web 应

用不适用（因为 Web 应用需要应用密码才能将授权码交换为 Access Token，截获了授权码并没有直接风险）。

这一机制的原理如下：

a. 客户端生成一个随机字符串：code verifier，并保存好这个随机字符串

```
code_challenge = transform(code_verifier, [Plain|S256])
```

A. 如果`transform method`是`plain`，那么`code challenge`等同于`code verifier`。

B. 如果`transform method`是`S256`，那么`code challenge`等于`code verifier`的 SHA256 哈希。

b. 在授权码请求中带上`code challenge`，以及生成`code challenge`的方法。这两者和服务端颁发的授权码被绑定了起来

c. 获取授权码之后，客户端在用授权码换取 Access Token 的时候，带上初始生成的`code verifier`。服务端按照绑定的`transform method`对`code verifier`进行计算，计算的结果与绑定的`code challenge`进行对比，如果一致则颁发 Access Token。

说明：

一个`code verifier`是一个高熵值的随机字符串，其合法的字符集为 [A-Z] / [a-z] / [0-9] / "-" / "." / "_" / "~"，长度介于 43 到 128 个字符之间。

`code challenge`的计算方法如下表所示：

code_challenge_method	code_challenge
plain	`code challenge`与`code verifier`的值相同`code_challenge = code_verifier`

code_challenge_method	code_challenge
S256	code challenge是code verifier的SHA256哈希值`code_challenge=BASE64URL-Encode(SHA256(ASCII(code_verifier)))`  说明: 哈希算法的输入为`code verifier`的ASCII 编码串，哈希算法的输出字符串需要进行 BASE64-URL 编码

示例：如果客户端生成如下的 code verifier: `dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk`，并采用 S256 的方式对 code verifier 进行 transform，那么获得的 code challenge 为`E9Melhoa2OwvFrEMTJguCHaoeK1t8URWbuGJSstw-cM`

2. 获取授权码（authorization code）

   授权码发放 Endpoint：*https://signin.aliyun.com/oauth2/v1/auth*

   获取授权码 API 接受以下query string 参数：

参数名	要求	描述
client_id	Required	应用的 Identifier
redirect_uri	Required	应用注册的重定向 URI 之一
response_type	Required	返回类型。根据 OAuth 2.0 标准，目前支持设置此参数的取值为`code`
scope	Optional	空格分隔的 OAuth 范围列表。如不指定此参数取值，则默认为应用注册的全部 OAuth 范围加上`openid`
state	Optional	设置为任意字符串，Auth Server 会将请求中的 state 参数以及取值放到返回结果中。应用通过 state 参数可以实现各种目的，比如状态保持，发送 nonce 从而减少 CSRF 威胁等。
code_challenge_method	Optional	推荐使用。如果不指定 code_challenge_method 参数，则取默认值plain

参数名	要求	描述
code_challenge	Optional	推荐使用。根据客户端指定的code_challenge_method，对随机生成的 code_verifier 进行转换和编码，转换的结果作为 code_challenge 参数的值在授权码请求中使用。  说明: 如果不指定此参数，则无法使用Proof Key 机制，在使用授权码换取令牌时也不需要指定相应的code_verifier。因此如果授权码被截获则可以直接被用于换取访问令牌。

请求样例:

```
https://signin.aliyun.com/oauth2/v1/authorize?
client_id=98989898&
redirect_uri=meeting%3A%2F%2Fauthorize%2F
&response_type=code
&scope=openid%20%2Fworksuite%2Fuseraccess
&state=1234567890
&code_challenge=E9Melhoa2OwvFrEMTJguCHaoeK1t8URWbuGJSstw-cM
&code_challenge_method=S256
```

响应样例:

```
GET HTTP/1.1 302 Found
Location: meeting://authorize/?code=ABAFDGDFXYZW888&state=1234567890
```

3. 用授权码换取访问令牌

一旦获得授权码，应用直接调用如下Endpoint换取访问令牌:

https://oauth.aliyun.com/v1/token。

用授权码换取访问令牌接受 Query String 或 FORM POST 参数，如下表所示:

参数名	要求	描述
code	Required	初始请求中获取的授权码
client_id	Required	应用的 Identifier

参数名	要求	描述
redirect_uri	Required	初始authorization请求中设置的redirect_uri参数
grant_type	Required	根据 OAuth 2.0 标准， 取值为`authorization_code`
code_verifier	Optional	第一步中生成的 code verifier，对应于授权码请求中使用的code_challenge参数。

请求样例：

```
POST /v1/token HTTP/1.1
Host: oauth.aliyun.com
Content-Type: application/x-www-form-urlencoded
code=ABAFDGDFXYZW888&
client_id=98989898&
redirect_uri=meeting://authorize/&
grant_type=authorization_code&
code_verifier=dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk
```

响应中包含以下字段：

参数名	描述
access_token	代表用户身份的访问令牌，应用使用此访问令牌来访问阿里云 Open API
expires_in	访问令牌的剩余有效时间，单位为秒
token_type	访问令牌的类型，为`Bearer`，意味着客户端不要理解 access_token 的内容含义，只需直接使用即可。
id_token	如果初始请求的 scope 参数包含了`openid`，则返回 id_token。id_token 为 OAuth 服务签名的 JWT (Json Web Token)。
refresh_token	客户端不要理解 refresh_token 的内容含义，只需直接使用即可。与 Web 应用不同的是，Native 应用不需要指定access_type 为 offline，也可以获取 refresh_token。

响应示例：

```
{
  "access_token": "eyJraWQiOiJrMTIzNCIsImVuYyI6I...",
```

```
  "token_type": "Bearer",
  "expires_in": 3600,
  "refresh_token": "Ccx63VVeTn2dxV7ovXXfLtAqLLERAH1Bc",
  "id_token": "eyJhbGciOiJIUzI1N..."
}
```

刷新访问令牌 (Refresh Token)

刷新访问令牌 Endpoint:

https://oauth.aliyun.com/v1/token

刷新访问令牌需要的参数如下:

参数名	要求	描述
refresh_token	Required	用授权码换取 token 时获取到的 refresh_token
client_id	Required	应用的 Identifier
grant_type	Required	根据 OAuth 2.0 标准， 取值为 `refresh_token`。

请求样例:

```
POST /v1/token HTTP/1.1
Host: oauth.aliyun.com
Content-Type: application/x-www-form-urlencoded
refresh_token=Ccx63VVeTn2dxV7ovXXfLtAqLLERAH1Bc&
client_id=98989898&
grant_type=refresh_token
```

请求的返回值与用授权码换取访问令牌的返回值一致，但不包含 refresh_token 和 id_token。

令牌撤销

撤销令牌的 Endpoint:

https://oauth.aliyun.com/v1/revoke

Native 应用获取了 refresh_token 后，在用户退出登录应用时或者用户将自己的账号从应用中移除时，必须撤销令牌。

撤销令牌需要的参数如下:

参数名	要求	描述
token	Required	需要撤销的token
client_id	Required	应用的Identifier

3.3.4 OIDC 登录场景

本文介绍应用如何通过 OIDC (Open ID Connect)登录阿里云但不访问阿里云 API 。

总体流程

Open ID Connect 1.0 是建立在 OAuth 2.0 协议之上的一个认证协议，因此其流程和 OAuth 2.0 的流程是一样的，根据不同的应用类型，其流程可以参考对应的 OAuth 2.0 流程。

ID Token

ID Token 包含了已认证用户的用户信息，与 Access Token 不同之处在于：

- 目的：ID Token 只是用于完成应用登录，不包含登录后扮演用户访问阿里云的 Open API。
- 内容：ID Token 用于表达用户信息，因此可以包含更多的用户属性，比如姓名，登录名，甚至头像，组织架构信息都属于 ID Token 的范畴。

ID Token 的用法通常为：

- 用于获取用户的信息，并在应用中进行展示。对于这种情况，由于 ID Token 是由应用直接从 OAuth 服务获取，因此应用可以不用验证 Token 的签名。
- 用于应用的各个不同模块之间通信。对于这种情况，建议应用接受 ID Token 的模块对 ID Token 进行验证。

ID Token 中包含的内容如下：

字段	含义
iat	Token 颁发时间（issued at）
exp	Token 过期时间（expiration）
iss	Token 颁发者（issuer）：取值为`https://oauth.aliyun.com`
aud	Token的接受者（audience）：为应用的 OAuth Client ID
sub	Token 的主体（subject）：为登录用户的 UID
upn	登录用户的 User Principal Name：比如 alice@demo.onaliyun.com
name	登录用户的显示名称
aid	登录用户的阿里云主账号 ID

ID Token的验证

阿里云颁发的 ID Token 为通过带有签名的 JWT (Json Web Token)，签名算法为 JWS 标准 RS256。ID Token 的验证，应至少包含以下几个方面：

- 签名验证：通过 OAuth 服务公布的签名公钥，验证 ID Token 的真实性（完整性）。
- 有效期验证：检查 iat 和 exp 字段的有效性，应考虑合理的时钟偏移。
- 检查 aud 是 client 自己：防止颁发给其他 client 的 ID Token 被恶意传递给本 client。

OIDC Discovery Endpoint

OpenID Connect 协议包含了不同的 Endpoint 用于不同的目的，为了简化实现和增加灵活性，协议建议 Open ID 服务提供者现一个 Discovery Endpoint，通过 JSON 文档提供一系列键值对，包含主要的提供者信息，例如协议支持的响应类型，issuer应该是什么取值，ID Token Signing Key 的地址，签名算法等。通过 Discovery Endpoint，很多库都可以在一定程度上自动化 OAuth 的流程。

阿里云 OIDC 服务的 Discovery Endpoint 为：

https://oauth.aliyun.com/.well-known/openid-configuration

上述 Discovery Endpoint 中包含的内容类似如下的 Sample

```
{
  "code_challenge_methods_supported": [
    "plain",
    "S256"
  ],
  "subject_types_supported": [
    "public"
  ],
  "response_types_supported": [
    "code"
  ],
  "issuer": "https://oauth.aliyun.com",
  "jwks_uri": "https://oauth.aliyun.com/v1/keys",
  "revocation_endpoint": "https://oauth.aliyun.com/v1/revoke",
  "token_endpoint": "https://oauth.aliyun.com/v1/token",
  "id_token_signing_alg_values_supported": [
    "RS256"
  ],
  "scopes_supported": [
    "openid",
    "aliuid",
    "profile"
  ],
  "authorization_endpoint": "https://signin.aliyun.com/oauth2/v1/auth"
```

```
}
```

附录

示例代码：获取 ID Token Signing Key

```
private List getSignPublicKey() {
  HttpResponse response = HttpClientUtils.doGet("https://oauth.aliyun.
com/v1/keys");
  List rsaKeyList = new ArrayList();
  if (response.getCode() == 200 && response.isSuccess()) {
    String keys = JSON.parseObject(response.getData()).getString("keys
");
    try {
      JSONArray publicKeyList = JSON.parseArray(keys);
      for (Object object : publicKeyList) {
        RSAKey rsaKey = RSAKey.parse(JSONObject.toJSONString(object));
        rsaKeyList.add(rsaKey);
      }
      return rsaKeyList;
    } catch (Exception e) {
      LOG.info(e.getMessage());
    }
  }
  LOG.info("GetSignPublicKey failed:{}", response.getData());
  throw new AuthenticationException(response.getData());
}
```

示例代码：验证 ID Token 的 JWS 签名

```
public boolean verifySign(SignedJWT signedJWT) {
  List publicKeyList = getSignPublicKey();
  RSAKey rsaKey = null;
  for (RSAKey key : publicKeyList) {
    if (signedJWT.getHeader().getKeyID().equals(key.getKeyID())) {
      rsaKey = key;
    }
  }
  if (rsaKey != null) {
    try {
      RSASSAVerifier verifier = new RSASSAVerifier(rsaKey.toRSAPubli
cKey());
      if (signedJWT.verify(verifier)) {
        return true;
      }
    } catch (Exception e) {
      LOG.info("Verify exception:{}", e.getMessage());
    }
  }
  throw new AuthenticationException("Can't verify signature for id
token");
}
```

ID Token Example

下面是一个不包含 Signature 的 JWT ID Token 的样例:

Header

```
{
  "alg": "RS256",
  "kid": "JC9wxzrhqJ0gtaCEt2QLUfevEUIwltFhui4O1bh67tU"
}
```

Body

```
{
  "exp": 1517539523,
  "sub": "aliyunuidxxxxxxxxx",
  "aud": "45678xxxxxxxx901234",
  "iss": "https:\/\/oauth.aliyun.com",
  "iat": 1517535923,
  "name": "alice", //显示名 需要profile scope
  "upn": "alice@demo.onaliyun.com",// 登录名 子账号是upn 需要profile
scope
  //"login_name":"abc@aliyun.com", // 登录名 主账号是 login_name 需要
profile scope
  "aid": "1937xxxxxxxxx9368", //对应的阿里云主账号id 需要aliuid scope
  "uid": "2133xxxxxxxxx4122", //对应的阿里云登录账号id 需要aliuid scope
}
```

UserInfo 接口

此外您还可以访问 userinfo 接口来获取用户信息。

Endpoint 为 *https://oauth.aliyun.com/v1/userinfo*

访问样例

```
GET /userinfo HTTP/1.1
Host: server.example.com
Authorization: Bearer SlAV32hkKG
```

返回样例

```
HTTP/1.1 200 OK
Content-Type: application/json
{
  "sub": "aliyunuidxxxxxxxxx",
  "name": "alice", //显示名 需要profile scope
  "upn": "alice@demo.onaliyun.com",// 登录名 子账号是upn 需要profile
scope
  //"login_name":"abc@aliyun.com", // 登录名 主账号是 login_name 需要
profile scope
  "aid": "1937xxxxxxxxx9368", //对应的阿里云主账号id 需要aliuid scope
  "uid": "2133xxxxxxxxx4122", //对应的阿里云登录账号id 需要aliuid scope
}
```

4 权限管理

4.1 权限策略概述

阿里云使用权限来描述内部身份（如用户、用户组、角色）对具体资源的访问能力。权限指在某种条件下允许或拒绝对某些资源执行某些操作，权限策略是一组访问权限的集合。

权限

- 主账户（资源 Owner）控制所有权限
  - 每个资源有且仅有一个属主（资源 Owner）。该属主必须是云账户，对资源拥有完全控制权限。
  - 资源属主不一定是资源创建者。比如，一个 RAM 用户被授予创建资源的权限，该用户创建的资源归属于主账户，该用户是资源创建者但不是资源属主。
- RAM 用户（操作员）默认无任何权限
  - RAM 用户代表的是操作员，其所有操作都需被显式授权。
  - 新建 RAM 用户默认没有任何操作权限，只有在被授权之后，才能通过控制台和 API 操作资源。
- 资源创建者（RAM 用户）不会自动拥有对所创建资源的任何权限
  - 如果 RAM 用户被授予创建资源的权限，用户将可以创建资源。
  - 但是 RAM 用户不会自动拥有对所创建资源的任何权限，除非资源 Owner 对他有显式的授权。

权限策略

权限策略（Policy）是用语法结构所描述的一组权限，它可以精确地描述被授权的资源集、操作集以及授权条件。通过给用户或用户组附加权限策略，用户或用户组中的所有用户就能获得权限策略中指定的访问权限，当权限策略中既有 Allow 又有 Deny 的授权语句时，遵循 Deny 优先的原则。

在 RAM 中，权限策略是一种资源实体，用户可以创建、更新、删除和查看权限策略。RAM 支持以下两种权限策略：

- 系统策略：系统策略是阿里云提供的一组通用权限策略，主要针对不同产品的只读权限或所有权限。阿里云会自动升级系统策略，用户不能修改。
- 自定义策略：由于系统策略的授权粒度比较粗，如果这种粗粒度权限策略不能满足您的需要，那么您可以创建自定义策略。比如，您想控制对某个具体的 ECS 实例的操作权限，或者您要求访

问者的资源操作请求必须来自于指定的 IP 地址，您必须使用自定义策略才能满足这种细粒度要求。

给 RAM 主体授权

给 RAM 主体授权，指给用户、用户组或角色绑定一个或多个权限策略。

- 绑定的权限策略可以是系统策略也可以是自定义策略。
- 如果绑定的权限策略被更新，更新后的权限策略自动生效，无需重新绑定权限策略。

4.2 权限策略管理

权限策略包含系统策略和自定义策略，其中系统策略只允许查看，不允许修改；自定义策略可以很方便的满足您的自定义需求。

创建自定义策略的前提条件

在创建自定义策略时，若您需要了解权限策略语言的基本结构和语法，请参考语法结构。

创建自定义策略

1. 登录 *RAM* 控制台。
2. 单击权限管理 > 权限策略管理。
3. 单击新建权限策略。
4. 填写策略名称和备注。
5. 配置模式选择可视化配置或脚本配置。
   - 若选择可视化配置：单击添加授权语句，根据界面提示，对权限效力、操作名称、资源等进行配置。
   - 若选择脚本配置，请参考语法结构进行编辑。

修改自定义策略

背景信息

当用户的权限发生变更时，例如新增或撤销权限，您需要修改权限策略。当您修改权限策略时可能会遇到以下问题：

- 希望一段时间后，老的权限策略还能继续使用。
- 修改完成后，您发现权限策略修改错了，需要返回修改前的策略。

权限策略具备版本管理机制：

- 可以为一个权限策略保留多个版本。如果超出限制，需要手动删除不需要的版本。
- 对于一个存在多版本的权限策略，只有一个版本是活跃的，即默认版本。

操作步骤

1. 在 RAM 控制台，单击权限管理 > 权限策略管理。
2. 通过权限策略名称找到需要管理的权限策略，单击其名称。

> 说明:
> 可使用关键字查询。

3. 单击版本管理。您可以：
   - 选择查看所有历史版本的策略内容。
   - 将非默认版本设为当前版本（即默认版本）。
   - 选择删除非默认版本。

删除自定义策略

您可以创建多个自定义策略，每个策略也可以维护多个版本。当您不再需要自定义策略时，您应该将权限策略删除。

前提条件

在删除某个权限策略前，应保证：

- 当前权限策略不存在多版本，只有一个默认版本。若该权限策略存在多个版本，您需要先删除除默认版本之外的所有版本。
- 当前权限策略未被引用（即授予用户、用户组或角色）。若该权限策略已被引用，您需要在该权限策略的引用记录中移除授权。

删除自定义策略的操作步骤

1. 在 RAM 控制台，单击权限管理。
2. 通过权限策略名称找到需要删除的权限策略，单击删除。

> 说明:
> 可使用关键字查询。

3. 在弹出的删除自定义策略对话框中，单击确定。

> 说明:
> 该策略有被引用记录时无法直接删除。

4.3 授权管理

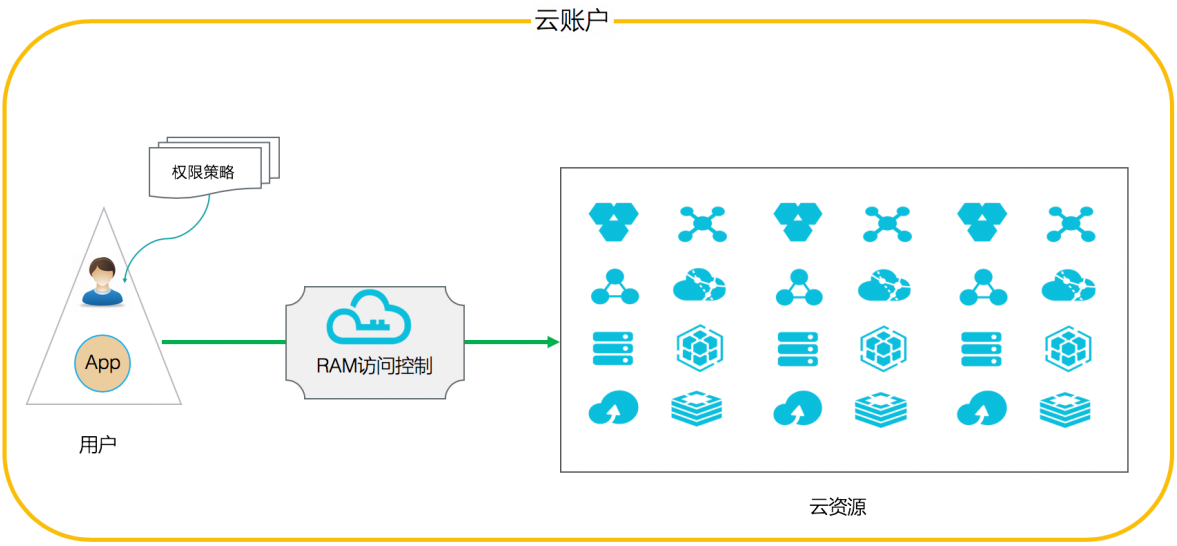
4.3.1 权限模型概述

阿里云为客户提供了云账户内授权和资源组内授权两级授权能力，请根据您的授权场景需要选择合理的授权模型。

云账户内授权

云账户内授权：对一个 RAM 身份主体添加权限策略时，该策略的可授权范围是云账户内的所有资源，这是最常见的一种权限模型。

图 4-1: 云账户内权限模型

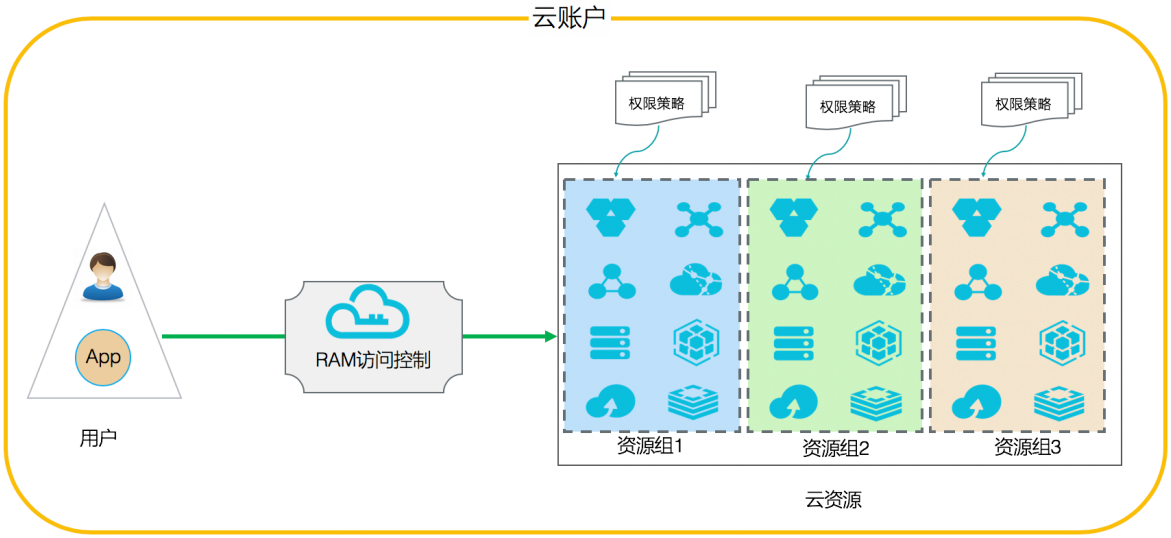


资源组内授权

资源组内授权：在某个资源组内对一个 RAM 身份主体添加权限策略时，该策略的可授权范围仅仅是该资源组内的资源。

管理员：在资源组内拥有`AdministratorAccess`系统策略的用户，资源组创建者默认为管理员。资源组管理员可以在资源组的成员管理中添加其他的 RAM 用户并在资源组内进行授权。

图 4-2: 资源组内权限模型



4.3.2 RAM 授权

在一个云账户内，RAM 授权指将一个或多个授权策略附加到 RAM 主体（用户、用户组或 RAM 角色）的过程。关于资源组内 RAM 主体授权，详见资源管理相关文档。

背景介绍

- 为用户授权：主要针对当前云账号下的 RAM 用户进行授权，RAM 用户可以访问相应资源。
- 为用户组授权：主要针对当前云账号下的 RAM 用户组进行授权，会应用到用户组下所有 RAM 用户，便于对资源访问需求类似的用户进行统一授权和管理。
- 为角色授权：主要针对更多复杂的云上应用场景，比如针对移动设备应用的临时授权、跨云账号的资源授权、云上应用的动态身份与授权管理、云服务之间的互操作授权等。

RAM 授权的前提条件

使用云账号（或拥有 RAM 操作权限的 RAM 用户）登录 *RAM* 控制台。

为用户授权

1. 登录 RAM 控制台，在人员管理菜单下，单击用户。
2. 通过用户登录名称/显示名称找到需要授权的用户，单击添加权限。
3. 从左侧权限策略名称列中勾选需要授予当前用户的权限策略，单击确认，完成给用户授权。

说明:

在右侧区域框，选择某条策略并单击 ×，可撤销该策略。

为用户组授权

1. 登录 RAM 控制台，在人员管理菜单下，单击用户组。
2. 通过用户组名称/显示名称找到需要授权的用户组，单击添加权限 。
3. 从左侧权限策略名称列中勾选需要授予当前用户组的权限策略，单击确认，完成给用户组授权。

说明:

在右侧区域框，选择某条策略并单击 ×，可撤销该策略。

为角色授权

新建角色时，可以选择新建阿里云账号角色或阿里云服务角色，并需选择相应的受信云账号或云服务（即允许其使用所创建的角色来访问您的云资源）。

- 对当前云账号阿里云账号角色授权，则当前云账号下的 RAM 用户可扮演角色并访问被授权的云资源。
- 对其他云账号阿里云账号角色授权，则指定的其他云账号下的 RAM 用户可扮演角色并访问被授权的云资源。
- 对阿里云服务角色授权，则受信的云服务可扮演角色并访问被授权的云资源。

1. 登录 RAM 控制台，单击 RAM 角色管理。
2. 通过 RAM 角色名称找到需要授权的角色，单击添加权限。
3. 从左侧权限策略名称列中勾选需要授予当前角色的权限策略，单击确认，完成给角色授权。

说明:

在右侧区域框，选择某条策略并单击 ×，可撤销该策略。

4.4 Policy 语言

4.4.1 Policy 基本元素

Policy 基本元素是权限策略的基本组成部分，RAM 中使用权限策略来描述授权的具体内容，掌握 Policy 基本元素的基本知识可以更合理的使用权限策略。

Policy 基本元素

使用权限策略之前需要了解 Policy 基本元素。

元素名称	描述
效力（Effect）	授权效力包括两种：允许（Allow）和拒绝（Deny）。
操作（Action）	操作是指对具体资源的操作。
资源（Resource）	资源是指被授权的具体对象。
限制条件（Condition）	限制条件是指授权生效的限制条件。

Policy 元素使用规则

- 效力（Effect）：

  取值 为 Allow 或 Deny，例如：`"Effect": "Allow"`。

- 操作（Action）：

  Action 支持多值，取值为云服务所定义的 API 操作名称。

  > 说明:
  > 操作是指对具体资源的操作，多数情况下 Action 与云产品的 API 一一对应，但也有例外。各产品支持的 Action 列表请参阅各产品关于 RAM 授权的文档，您可以在支持 RAM 的云服务中查询相关文档的链接。

  格式：

  `<service-name>:<action-name>`。

  - `service-name`: 阿里云产品名称。例如：ecs，rds，slb，oss，ots 等。
  - `action-name: service`：相关的 API 操作接口名称。

  描述样例：

  ```
  "Action": ["oss:ListBuckets", "ecs:Describe*", "rds:Describe*"]
  ```

- 资源（Resource）

  Resource 通常指资源，即操作对象。

  格式：

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

  - `acs`: Alibaba Cloud Service 的首字母缩写，表示阿里云的公有云平台。
  - `service-name`: 阿里云产品名称。例如：ecs，rds，slb，oss，ots 等。
  - `region`: 地域信息。如果不支持该项，可以使用通配符`*`来代替。
  - `account-id`: 账号 ID。例如：`1234567890123456`，可以用`*`代替。
  - `relative-id`: 与服务相关的资源描述部分，其语义由具体服务指定。这部分的格式支持树状结构（类似文件路径）。以 oss 为例，表示一个 OSS 对象的格式为：`relative-id = “mybucket/dir1/object1.jpg”`。

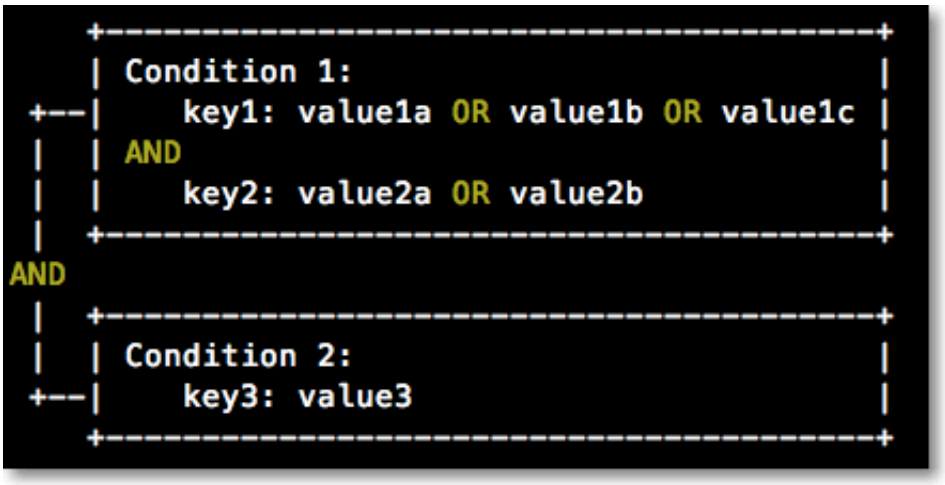
  描述样例：

  ```
  "Resource": ["acs:ecs:*:*:instance/inst-001", "acs:ecs:*:*:instance/inst-002", "acs:oss:*:*:mybucket", "acs:oss:*:*:mybucket/*"]
  ```

- 限制条件（Condition）：

  条件块（Condition Block）由一个或多个条件子句构成。一个条件子句由条件操作类型、条件关键字和条件值组成。

  图 4-3: 条件块判断逻辑



  逻辑说明：

  - 条件满足：一个条件关键字可以指定一个或多个值，在条件检查时，如果条件关键字的值与指定值中的某一个相同，即可判定条件满足。
  - 条件子句满足：同一条件操作类型的条件子句下，若有多个条件关键字，所有条件关键字必须同时满足，才能判定该条件子句满足。
  - 条件块满足：条件块下的所有条件子句同时满足的情况下，才能判定该条件块满足。

  条件操作类型

  条件操作类型包括：字符串类型（String）、数字类型（Numeric）、日期类型（Date and time）、布尔类型（Boolean）和 IP 地址类型（IP address）。

条件操作类型	支持类型
字符串类型（String）	- StringEquals - StringNotEquals - StringEqualsIgnoreCase - StringNotEqualsIgnoreCase - StringLike - StringNotLike

条件操作类型	支持类型
数字类型（Numeric）	- NumericEquals - NumericNotEquals - NumericLessThan - NumericLessThanEquals - NumericGreaterThan - NumericGreaterThanEquals
日期类型（Date and time）	- DateEquals - DateNotEquals - DateLessThan - DateLessThanEquals - DateGreaterThan - DateGreaterThanEquals
布尔类型（Boolean）	Bool
IP 地址类型（IP address）	- IpAddress - NotIpAddress

条件关键字

阿里云通用条件关键字命名格式：

```
acs:<condition-key>
```

阿里云产品级别条件关键字命名格式：

```
<service-name>:<condition-key>
```

表 4-1: 通用条件关键字

通用条件关键字	类型	说明
`acs:CurrentTime`	Date and time	Web Server 接收到请求的时间。以 ISO 8601 格式表示，例如：`2012-11-11T23:59:59Z`。
`acs:SecureTransport`	Boolean	发送请求是否使用了安全信道。例如：HTTPS。
`acs:SourceIp`	IP address	发送请求时的客户端 IP 地址。

通用条件关键字	类型	说明
acs:MFAPresent	Boolean	用户登录时是否使用了多因素认证。

表 4-2: 产品级别条件关键字

产品名称	条件关键字	类型	说明
ECS	ecs:tag/<tag-key>	String	ECS 资源的标签关键字，可自定义。
RDS	rds:ResourceTag/<tag-key>	String	RDS 资源的标签关键字，可自定义。
OSS	oss:Delimiter	String	OSS 对 Object 名字进行分组的分隔符。
OSS	oss:Prefix	String	OSS Object 名称的前缀。

权限策略样例

以下权限策略的含义：允许对 OSS 的 samplebucket 进行只读操作，限制条件：请求者的 IP 来源为 42.160.1.0。

```
{
      "Version": "1",
      "Statement":
        [{
          "Effect": "Allow",
            "Action": ["oss:List*", "oss:Get*"],
            "Resource": ["acs:oss:*:*:samplebucket", "acs:oss:*:*:
samplebucket/*"],
            "Condition":
             {
                "IpAddress":
                 {
                    "acs:SourceIp": "42.160.1.0"
                  }
              }
         }]
}
```

4.4.2 Policy 结构和语法

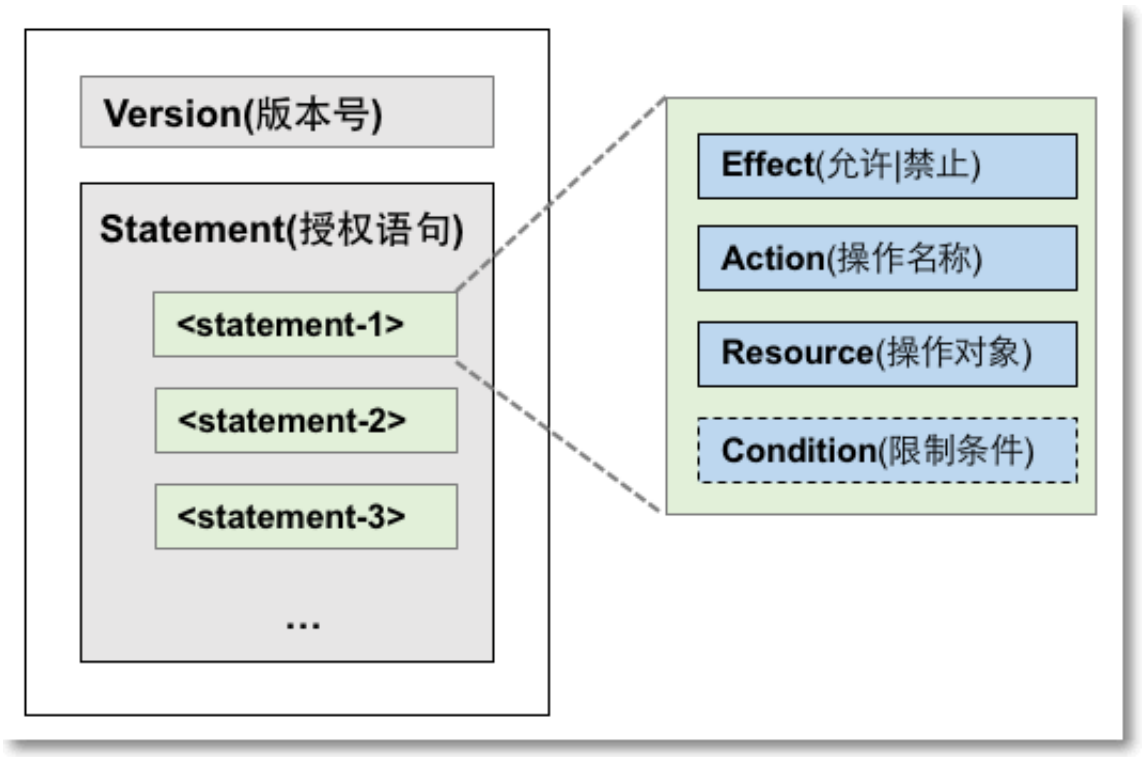
本文介绍 RAM 中权限策略的语法结构和规则，帮助您正确理解 Policy 语法，以完成创建或更新权限策略。

Policy 结构

Policy 结构包括：版本号及授权语句（Statement）列表。

每条授权语句包括：Effect（授权效力）、Action（操作列表）、Resource（资源列表）以及Condition（限制条件），其中限制条件是可选项。

图 4-4: Policy 结构



运用 Policy 语法的前提条件

运用 Policy 语法前，首先应了解 Policy 字符及其使用规则。

- Policy 字符：
  - Policy 中所包含的 JSON 字符：`{ } [ ] " , :`。
  - 描述语法使用的特殊字符：`= < > ( ) |`。
- Policy 字符使用规则：
  - 当一个元素允许多值时，使用逗号和省略号来表达。例如：`[ <action_string>, <action_string>, ...]`。

> 说明：

在所有支持多值的元素中，使用单值进行表达也是有效的，且两种表达方式效果相同。例如：`"Action": [<action_string>]` 和 `"Action": <action_string>`

- 元素带有问号表示此元素是一个可选元素。例如：`<condition_block?>`。
- 多值之间用竖线 `|` 隔开，表示取值只能选取这些值中的某一个。例如：`("Allow" | "Deny")`。
- 使用双引号的元素，表示此元素是文本串。例如：`<version_block> = "Version" : ("1")`。

Policy 语法

Policy 语法描述如下：

```
policy  = {
     <version_block>,
     <statement_block>
}
<version_block> = "Version" : ("1")
<statement_block> = "Statement" : [ <statement>, <statement>, ... ]
<statement> = {
    <effect_block>,
    <action_block>,
    <resource_block>,
    <condition_block?>
}
<effect_block> = "Effect" : ("Allow" | "Deny")
<action_block> = ("Action" | "NotAction") :
    ("*" | [<action_string>, <action_string>, ...])
<resource_block> = ("Resource" | "NotResource") :
    ("*" | [<resource_string>, <resource_string>, ...])
<condition_block> = "Condition" : <condition_map>
<condition_map> = {
  <condition_type_string> : {
      <condition_key_string> : <condition_value_list>,
      <condition_key_string> : <condition_value_list>,
      ...
  },
  <condition_type_string> : {
      <condition_key_string> : <condition_value_list>,
      <condition_key_string> : <condition_value_list>,
      ...
  }, ...
}
<condition_value_list> = [<condition_value>, <condition_value>, ...]
<condition_value> = ("String" | "Number" | "Boolean")
```

Policy 语法说明：

· 版本：当前支持的 Policy 版本为 1。

- 授权语句：一个 Policy 可以有多条授权语句。
  - 每条授权语句的效力是`Allow`或`Deny`。

    说明：
    一条授权语句中，Action（操作）和 Resource（资源）都支持多值。
  - 每条授权语句都支持独立的限制条件（Condition）。

    说明：
    一个条件块可以支持多种条件操作类型，以及多种条件的逻辑组合。
- Deny 优先原则：一个用户可以被授予多个权限策略，当这些权限策略同时包含`Allow`和`Deny`时，遵循 Deny 优先原则。
- 元素取值：
  - 当取值为数字（Number）或布尔值（Boolean）时，与字符串类似，需要使用双引号。
  - 当元素取值为字符串值（String）时，支持使用`*`和`?`进行模糊匹配。
    - `*`代表 0 个或多个任意的英文字母。

      说明：
      例如：`ecs:Describe*`表示 ecs 的所有以 Describe 开头的 action。
    - `?`代表 1 个任意的英文字母。

Policy 格式检查

RAM 仅支持 JSON 格式。当创建或更新权限策略时，RAM 会首先检查 JSON 格式的正确性。

- 关于 JSON 的语法标准请参考 *RFC 7159*。
- 您也可以使用一些在线的 JSON 格式验证器和编辑器来校验 JSON 文本的有效性。

4.4.3 Policy 样例

本文介绍了解 Policy 基本元素、结构及语法后，如何创建 Policy 样例。

如下所示的 Policy 样例中，包含两条授权语句（Statement）：

- 第 1 条授权语句：允许对华东 1（杭州）地域的所有 ecs 资源授予查看权限`ecs:Describe*`。
- 第 2 条授权语句：允许对 oss 的 mybucket 存储桶中的对象授予只读访问权限`oss:ListObjects`和`oss:GetObject`，并限制请求者的 IP 来源必须是`42.120.88.10`或`42.120.66.0/24`。

```
{
```

```
    "Version": "1",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ecs:Describe*",
            "Resource": "acs:ecs:cn-hangzhou:*:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "oss:ListObjects",
                "oss:GetObject"
            ],
            "Resource": [
                "acs:oss:*:*:mybucket",
                "acs:oss:*:*:mybucket/*"
            ],
            "Condition":{
                "IpAddress": {
                    "acs:SourceIp": ["42.120.88.10", "42.120.66.0/24"]
                }
            }
        }
    ]
}
```

4.5 验权规则

本文介绍了在 RAM 中使用不同身份访问资源时的权限检查模型及规则，帮助您理解权限策略。

基本模型

在 RAM 中访问资源分为以主账号身份、以 RAM 用户身份和扮演 RAM 角色身份三种情形；针对每种情形，系统的授权判断条件如下表所示。

访问类型	允许访问条件（同时满足）
主账号身份访问资源	主账号是资源 Owner。  说明: 少数云产品（如日志服务）直接支持对跨云账号 ACL 授权，如果通过 ACL 授权检查，则允许访问。
RAM 用户身份访问资源	· RAM 用户所属的主账号对资源有访问权限。 · 主账号对 RAM 用户有显式的 Allow 权限策略。

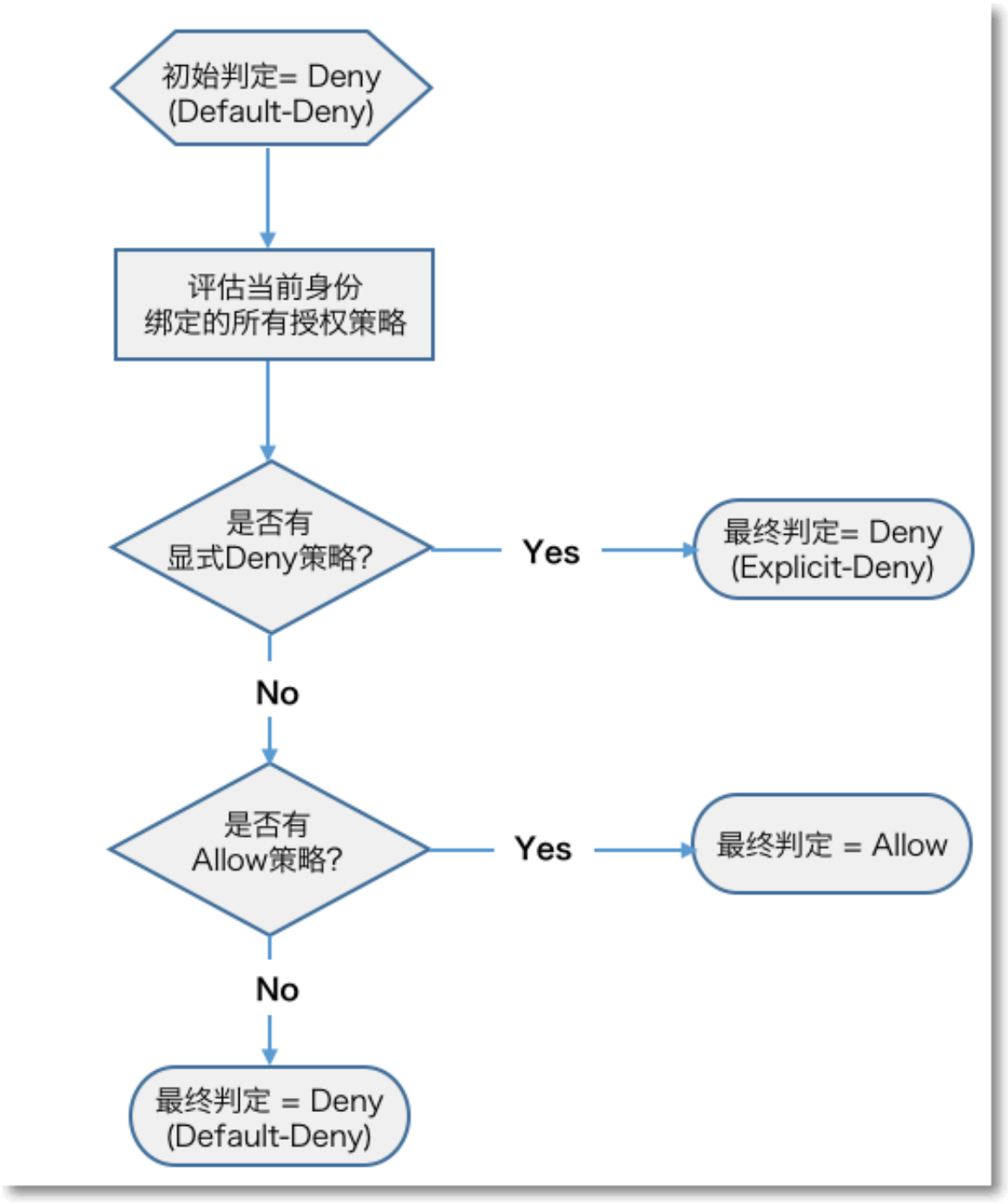
访问类型	允许访问条件（同时满足）
RAM 角色身份访问资源	· RAM 角色所属的主账号对资源有访问权限。 · 主账号对 RAM 角色有显式的 Allow 权限策略。 · RAM 角色令牌（STS token）有相应的权限策略。

RAM 用户身份的权限策略检查逻辑

RAM 用户访问资源时，默认没有任何权限，除非有进行显式的授权（给 RAM 用户绑定权限策略）。权限策略语句支持 Allow（允许）和 Deny（禁止）两种授权类型，当多个授权语句对一个资源操作分别出现 Allow 和 Deny 授权时，遵循 Deny 优先 原则。

权限策略检查逻辑如下图所示：

图 4-5: 权限策略检查逻辑



RAM 用户访问资源时，权限检查逻辑如下：

1. 检查 RAM 用户身份所绑定的权限策略是否有授权：
   - 如果是 Deny，则拒绝访问。
   - 否则进入下一步检查。

2. 检查 RAM 角色所属的主账号是否有访问权限：

   · 如果是资源 Owner，则允许访问。

   · 否则查看该资源是否有支持跨账号 ACL 许可：

     - 有则允许访问。

     - 否则拒绝访问。

RAM 角色身份的权限策略检查逻辑

RAM 角色（使用角色访问令牌）访问资源时，权限检查逻辑如下：

1. 如果当前访问令牌有指定权限策略（调用 AssumeRole 时所传入的权限策略参数），则按照上述权限策略检查逻辑进行判断：

   · 如果是 Deny，则拒绝访问。

   · 否则进入下一步检查。

   如果当前访问令牌没有指定权限策略，则直接进入下一步检查。

2. 检查 RAM 角色身份所绑定的权限策略是否有授权：

   · 如果是 Deny，则拒绝访问。

   · 否则进入下一步检查。

3. 检查 RAM 角色所属的主账号是否有访问权限：

   · 如果是资源 Owner，则允许访问。

   · 否则查看该资源是否有支持跨账号 ACL 许可：

     - 有则允许访问。

     - 否则拒绝访问。

5 典型场景

5.1 用户管理与分权

本文介绍了当企业有多种云资源，使用 RAM 的授权管理功能，实现用户分权及资源统一管理。

背景信息

企业 A 的某个项目（Project-X）上云，购买了多种阿里云产品，例如：ECS 实例、RDS 实例、SLB 实例、OSS 存储空间等。项目里有多个员工需要操作这些云资源，由于每个员工的工作职责不同，需要的权限也不一样。企业 A 希望能够达到以下要求：

- 出于安全或信任的考虑，A 不希望将云账号密钥直接透露给员工，而希望能给员工创建独立账号。
- 用户账号只能在授权的前提下操作资源。

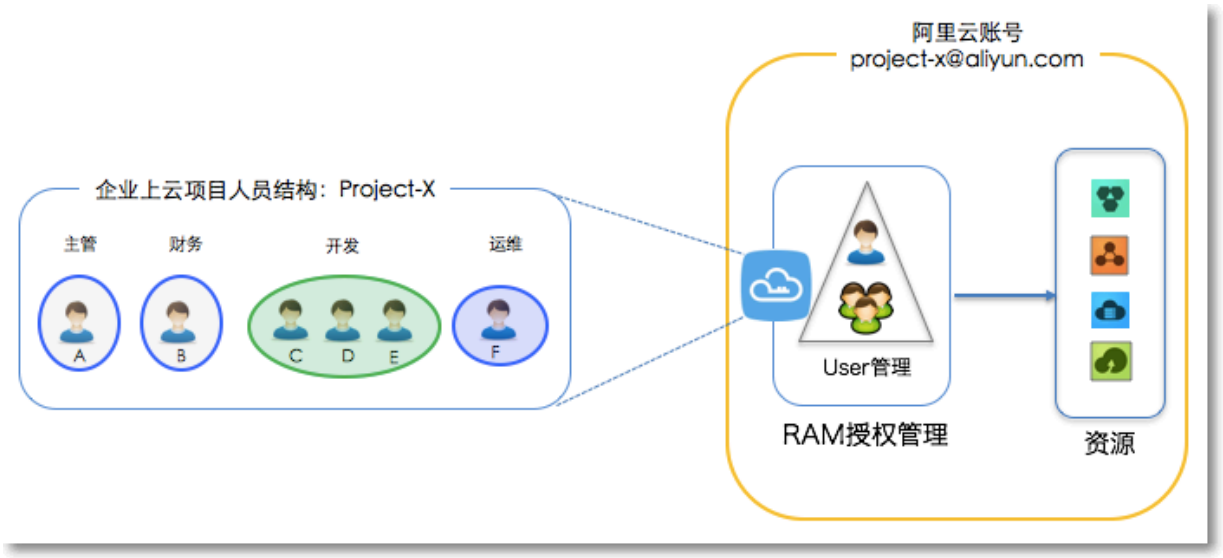
  A 随时可以撤销用户账号身上的权限，也可以随时删除其创建的用户账号。
- 不需要对用户账号进行独立的计量计费，所有开销都算在 A 的头上。

需求分析

- 杜绝多员工共享主账号，防止主账号密码或 AK 泄露导致风险不可控。
- 给不同员工分配独立的用户账号（或操作员账号）并独立分配权限，做到责权一致。
- 所有用户账号的所有操作行为可审计。
- 不需要分别核算每个操作人员的成本，所发生费用统一计入主账号账单。

企业场景解决方案

图 5-1: 企业场景解决方案



实现用户管理与分权的操作步骤

1. 设置 MFA#可选#，避免因主账号密码泄露导致风险。
2. 创建 RAM 用户。为不同员工（或应用系统）创建 RAM 用户账号，并按需设置登录密码或创建 AccessKey。
3. 创建 RAM 用户组#可选#。如果有多个员工的职责相同，建议创建用户组，并将用户添加到用户组。
4. RAM 授权。给用户或用户组添加一条或多条系统策略。如果需要更细粒度的授权，可以创建自定义策略，然后给用户或用户组授权。

5.2 针对移动设备应用的临时授权

本文介绍如何使用 RAM 角色安全令牌对移动设备进行临时授权，以实现使用临时安全令牌直接访问相关资源。

背景信息

企业 A 开发了一款移动 App，并购买了 OSS 服务。移动 App 需要直连 OSS 上传或下载数据，但是移动 App 运行在用户自己的终端设备上，这些设备并不受 A 的控制。

企业 A 有如下要求：

- 直传数据：企业 A不希望所有 App 都通过企业自己的服务端应用服务器（AppServer）来进行数据中转，而希望让 App 能直连 OSS 上传或下载数据。

- 安全考虑：企业 A 不能将访问密钥保存到移动 App 中，因为移动设备是归属于用户控制，属于不可信任的运行环境。
- 风险管控：企业 A 希望将安全风险控制到最小，每个移动 App 直连 OSS 时都必须使用最小权限的访问令牌且访问时效也要很短。

临时访问授权解决方案

- 云账号 A 在 RAM 中创建一个角色，给角色授予合适的权限，并允许 AppServer 以 RAM 用户身份使用该角色。

  操作流程见创建角色、用户及授权。
- 当 App 需要直连 OSS 上传或下载数据时，AppServer 可以扮演角色（调用 STS AssumeRole），获取角色的一个临时安全令牌并传送给 App，App 就可以使用临时安全令牌直接访问 OSS API。

  操作流程见获取、传递角色令牌及访问。
- AppServer 可以在使用角色时进一步限制临时安全令牌的资源操作权限，以更精细地控制每个 App 的权限。

  操作流程见 限制 STS token 权限。

创建角色、用户及授权

假设云账号 A 的 AccountID 为：11223344。

1. 云账号 A 创建 RAM 角色：`oss-readonly`，并选择当前云账号作为受信云账号，即只允许云账号 A 下的 RAM 用户来扮演该角色。

   具体操作请参考管理 RAM 角色。

   角色创建成功后，在角色详情中可以查看到该角色的基本信息：

   - 角色的 Arn 如下：

     ```
     acs:ram::11223344:role/oss-readonly
     ```

   - 角色的信任策略（只允许云账号 A 下的 RAM 角色来扮演角色）如下：

     ```
     {
     "Statement": [
     {
      "Action": "sts:AssumeRole",
      "Effect": "Allow",
      "Principal": {
        "RAM": [
          "acs:ram::11223344:root"// 当角色类型为阿里云账号时，此处固定是root
        ]
      }
     }
     ```

```
],
"Version": "1"
}
```

2. 云账号 A 给角色授权，向 RAM 角色（oss-readonly）授予`AliyunOSSReadOnlyAccess`（只读访问 OSS）的权限。

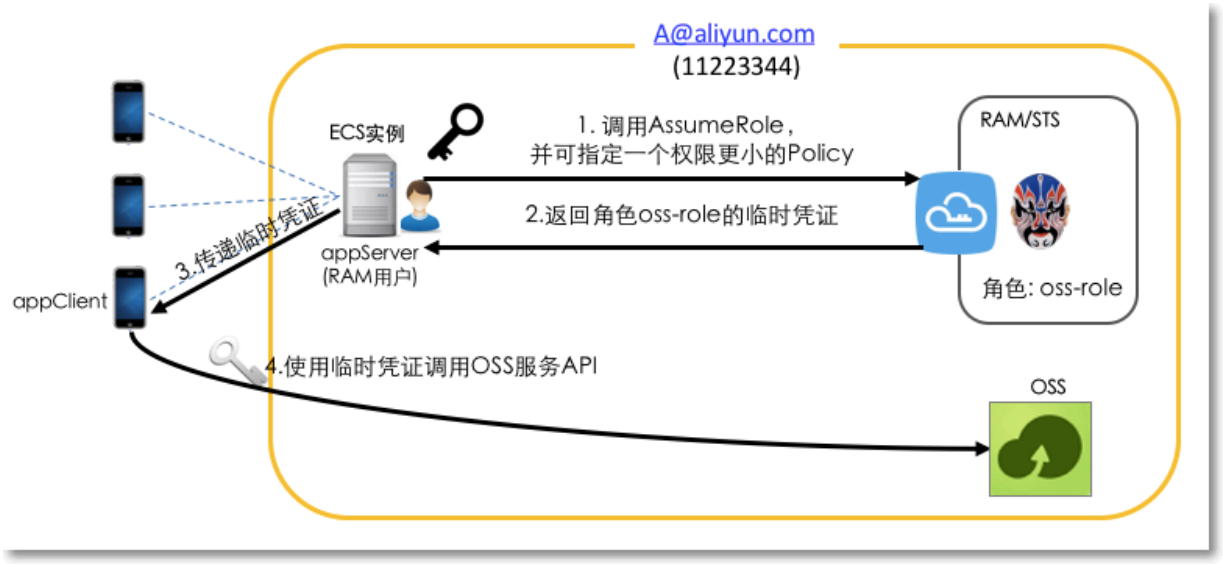
   具体操作请参考 *RAM 授权*。

3. 云账号 A 为 AppServer 创建 RAM 用户身份： Appserver，并为该 RAM 用户创建 AccessKey 并授予`AliyunSTSAssumeRoleAccess`（调用 STS AssumeRole 接口）的系统策略。

获取、传递角色令牌及访问 OSS

App 获取并使用角色令牌调用 OSS API 的操作示意图如下：

图 5-2: 操作流程



AppServer 使用已拥有的 RAM 用户的 AccessKey 调用 STS API *AssumeRole*。

> **说明:**
> 必须配置 AppServer 的 AccessKey，而非主账号 A 的 AccessKey。

使用 aliyuncli 来调用 AssumeRole 的命令示例如下：

```
$ aliyuncli sts AssumeRole --RoleArn acs:ram::11223344:role/oss-
readonly --RoleSessionName client-001
 {
     "AssumedRoleUser": {
         "AssumedRoleId": "391578752573972854:client-001",
         "Arn": "acs:ram::11223344:role/oss-readonly/client-001"
     },
     "Credentials": {
```

```
        "AccessKeySecret": "93ci2umK1QKNEja6WGqi1Ba7Q2Fv9PwxZqtVF2Vy
nUvz",
        "SecurityToken": "CAES6AIIARKAAUiwSHpkD3GXRMQk9stDr3YSVbyG
qanqkS+fPlEEkjZ+dlgFnGdCI2PV93jksole8ijH8dHJrHRA5JA1YCGsfX5hrzcNM3
7Vr4eVdWFVQhoCw0DXBpHv//ZcITp+ELRr4MHsnyGiErnDsXLkI7q/sbuWg6PACZ/
jzQfEWQb/f7Y1Gh1TVFMuRjEzR2pza1hUamszOGRCWTZZeEp0WEFaayISMzkxNTc4NzUy
NTczOTcyODU0KgpjbGllbnQtMDAxMKT+lIHBKjoGUnNhTUQ1QkoKATEaRQoFQW
xsb3cSGwoMQWN0aW9uRXF1YWxzEgZBY3Rpb24aAwoBKhIfCg5SZXNvdXJjZUVxdWFscxII
UmVzb3VyY2UaAwoBKkoFNDMyNzRSBTI2ODQyWg9Bc3N1bWVkUm9sZVVzZXJgAGoSMzkxNT
c4NzUyNTczOTcyODU0cgllY3MtYWRtaW544Mbewo/26AE=",
        "Expiration": "2016-01-13T15:02:37Z",
        "AccessKeyId": "STS.F13GjskXTjk38dBY6YxJtXAZk"
    },
    "RequestId": "E1779AAB-E7AF-47D6-A9A4-53128708B6CE"
}
```

限制 STS token 权限

1. **调用 AssumeRole 后授予更小的权限。**

   上述 AssumeRole 调用时没有指定 Policy 参数，意味着该 STS token 拥有 `oss-readonly` 的所有权限。如果需要进一步限制 STS token 的权限（例如：只允许访问 `sample-bucket/2015/01/01/*.jpg`），那么可以通过设置如下Policy 参数：

```
$ aliyuncli sts AssumeRole --RoleArn acs:ram::11223344:role/oss-
readonly --RoleSessionName client-002 --Policy "{\"Version\":\"1\",
 \"Statement\": [{\"Effect\":\"Allow\", \"Action\":\"oss:GetObject
\", \"Resource\":\"acs:oss:*:*:sample-bucket/2015/01/01/*.jpg\"}]}"
{
   "AssumedRoleUser": {
       "AssumedRoleId": "391578752573972854:client-002",
       "Arn": "acs:ram::11223344:role/oss-readonly/client-002"
   },
   "Credentials": {
       "AccessKeySecret": "28Co5Vyx2XhtTqj3RJgdud4ntyzrSNdUvNygAj7x
EMow",
       "SecurityToken": "CAESnQMIARKAASJgnzMzlXVyJn4KI+FsysaIpTGm
8ns8Y74HVEj0pOevO8ZWXrnnkz4a4rBEPBAdFkh3197GUsprujsiU78Fkszx
hnQPKkQKcyvPihoXqKvuukrQ/Uoudk31KAJEz5o2EjlNUREcxWjRDRSISMzkxNTc4
NzUyNTczOTcyODU0KgpjbGllbnQtMDAxMKmZxIHBKjoGUnNhTUQ1Qn8KATEa
egoFQWxsb3cSJwoMQWN0aW9uRXF1YWxzEgZBY3Rpb24aDwoNb3NzOkdldE9i
amVjdBJICg5SZXNvdXJjZUVxdWFscxIIUmVzb3VyY2UaLAoqYWNzOm9zczoq
Oio6c2FtcGxlLWJ1Y2tldC8yMDE1LzAxLzAxLyouanBnSgU0MzI3NFIFMjY4
NDJaD0Fzc3VtZWRSb2xlVXNlcmAAahIzOTE1Nzg3NTI1NzM5NzI4NTRyCWVj
cy1hZG1pbnjgxt7Cj/boAQ==",
       "Expiration": "2016-01-13T15:03:39Z",
       "AccessKeyId": "STS.FJ6EMcS1JLZgAcBJSTDG1Z4CE"
   },
   "RequestId": "98835D9B-86E5-4BB5-A6DF-9D3156ABA567"
}
```

说明:

上述 STS token 的默认过期时间为 3600 秒，用户还可以通过 DurationSeconds 参数来限制 STS token 的过期时间（最长不超过 3600 秒）。

2. AppServer 获取并解析临时凭证。
   - AppServer 从 AssumeRole 返回的临时凭证中获取 AccessKeyId、AccessKeySecret 和 STS token。
   - 考虑到 STS token 过期时间较短，如果应用业务需要一个较长的过期时间，需要 AppServer 重新颁发新的 STS token（比如每隔 1800 秒颁发一次 STS token）。
3. AppServer 将临时凭证传递给 App。
4. App 使用 STS token 直接访问云服务的 API（比如 OSS）。

   下面是 aliyuncli 使用 STS token 访问 OSS 对象的操作命令（颁发给 client-002 的 STS token）：

```
配置STS token语法: aliyuncli oss Config --host  --accessid  --
accesskey  --sts_token
$ aliyuncli oss Config --host oss.aliyuncs.com --accessid STS.
FJ6EMcS1JLZgAcBJSTDG1Z4CE --accesskey 28Co5Vyx2XhtTqj3RJgdud4ntyzrSN
dUvNygAj7xEMow --sts_token CAESnQMIARKAASJgnzMzlXVyJn4KI+FsysaIpTGm
8ns8Y74HVEj0pOevO8ZWXrnnkz4a4rBEPBAdFkh3197GUsprujsiU78Fkszx
hnQPKkQKcyvPihoXqKvuukrQ/Uoudk31KAJEz5o2EjlNUREcxWjRDRSISMzkxNTc4
NzUyNTczOTcyODU0KgpjbGllbnQtMDAxMKmZxIHBKjoGUnNhTUQ1Qn8KATEa
egoFQWxsb3cSJwoMQWN0aW9uRXF1YWxzEgZBY3Rpb24aDwoNb3NzOkdldE9i
amVjdBJICg5SZXNvdXJjZUVxdWFscxIIUmVzb3VyY2UaLAoqYWNzOm9zczoq
Oio6c2FtcGxlLWJ1Y2tldC8yMDE1LzAxLzAxLyouanBnSgU0MzI3NFIFMjY4
NDJaD0Fzc3VtZWRSb2xlVXNlcmAAahIzOTE1Nzg3NTI1NzM5NzI4NTRyCWVj
cy1hZG1pbnjgxt7Cj/boAQ==
访问OSS对象
$ aliyuncli oss Get oss://sample-bucket/2015/01/01/grass.jpg grass.
jpg
```

更多参考

关于移动应用直传场景，可参考以下文档：

- 快速搭建移动应用直传服务。
- 权限控制。
- 快速搭建移动应用上传回调服务。
- STS临时授权访问。

5.3 跨云账号的资源授权

本文介绍当一个企业希望如将部分业务授权给另一个企业时，使用 RAM 角色进行跨账号授权来管理资源的授权及访问。

背景信息

云账号 A 和云账号 B 分别代表不同的企业。A 购买了多种云资源来开展业务，例如：ECS 实例、RDS 实例、SLB 实例、OSS 存储空间等。

- 企业 A 希望能专注于业务系统，而将云资源运维、监控、管理等任务授权给企业 B。
- 企业 B 还可以进一步将 A 的资源访问权限分配给 B 的某一个或多个员工，B 可以精细控制其员工对 资源的操作权限。
- 如果 A 和 B 的这种运维合同关系终止，A 随时可以撤销对 B 的授权。

需求分析

- 账号 A 是资源 Owner，可以授权账号 B 来操作相应资源。
- 账号 B 给其名下的 RAM 用户（代表员工或应用）进行授权，限制 RAM 用户对资源的访问能力。当 B 的员工加入或离职时，A 无需做任何权限变更。
- 如果双方业务终止，A 随时可以撤销对 B 的授权。

解决方案

针对以上需求，使用 RAM 角色可以实现跨账号授权及资源访问的控制。

- 云账号 A 在 RAM 中创建一个角色，给角色授予合适的权限，并允许云账号 B 使用该角色。

  操作流程见跨账号授权。
- 如果云账号 B 下的某个员工（RAM 用户）需要使用该角色，那么云账号 B 可以自主进行授权控制。账号 B 下的 RAM 用户将使用被授予的角色身份来操作账号 A 的资源。

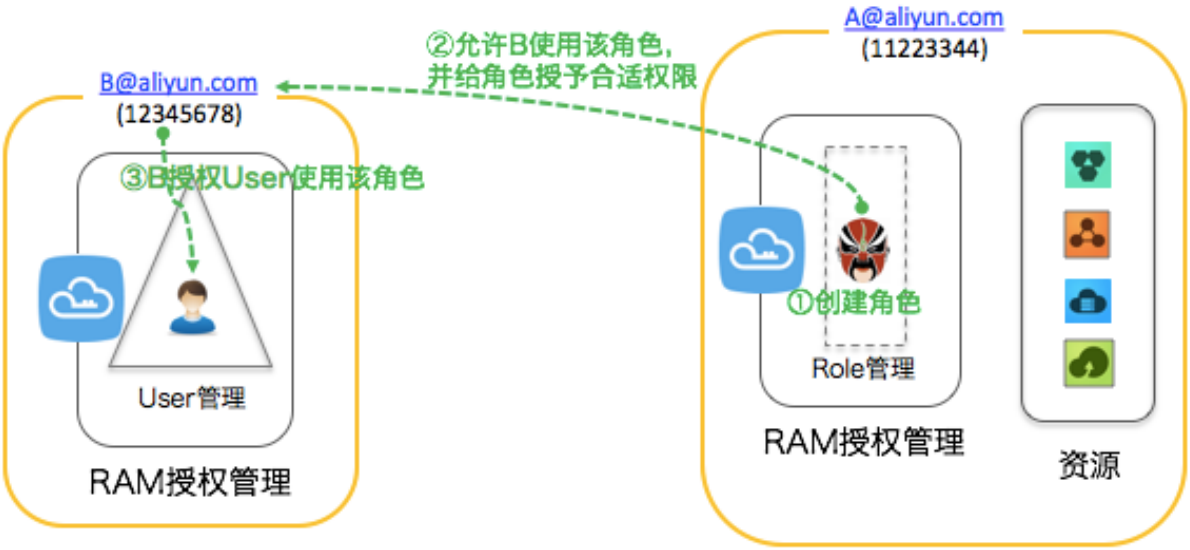
  操作流程见 跨账号资源访问。
- 如果账号 A 与账号 B 的合作终止，A 只需要撤销账号 B 对该角色的使用。此时账号 B 下的所有 RAM 用户对该角色的使用权限将被自动撤销。

  操作流程见撤销跨账号授权 。

跨账号授权

假设企业 A（AccountID=11223344，别名 company-a）需要授权企业 B（AccountID=12345678，别名 company-b）的员工对其 ECS 进行操作。

图 5-3: 使用 RAM 角色跨账号授权



1. 云账号 A 创建用户 RAM 角色 ecs-admin，并选择其他云账号：12345678 作为受信云账号。

   具体操作请参考管理 RAM 角色。

   角色创建成功后，在角色详情中可以查看到该角色的基本信息：

   - 角色的 ARN 如下：

     ```
     acs:ram::11223344:role/ecs-admin
     ```

   - 角色的信任策略（只允许企业 B 来扮演角色）如下：

     ```
     {
     "Statement": [
     {
      "Action": "sts:AssumeRole",
      "Effect": "Allow",
      "Principal": {
        "RAM": [
          "acs:ram::12345678:root"
        ]
      }
     }
     ],
     "Version": "1"
     }
     ```

2. 云账号 A 给角色 ecs-admin 授权`AliyunECSFullAccess`。

   具体操作请参考RAM 授权。

3. 云账号 B 为其员工创建 RAM 用户 zhangsan，并为该 RAM 用户设置登录密码及授予 `AliyunSTSAssumeRoleAccess`（调用 STS AssumeRole 接口）的系统策略。

跨账号资源访问

通过对云账户 B 的 RAM 用户 zhangsan 进行授权后，RAM 用户通过控制台可以访问云账号 A 的 ECS 资源。

1. 云账号 B 的 RAM 用户 zhangsan 登录 RAM 控制台。

   子用户登录时需输入账号别名：company-b、子用户名称：zhangsan 和子用户密码：123456。

2. 登录成功后，将鼠标悬停在右上角头像的位置，单击切换身份。

   进入身份切换页面，输入正确的账号别名：company-a 和角色名：ecs-admin，进行角色切换。

> 说明:
> 切换成功后，云账号 B 的 RAM 用户便可以操作云账号 A 下的 ECS 资源。

撤销跨账号授权

云账号 A 也可以撤销云账号 B 对角色 ecs-admin 的使用。

1. 云账号 A 登录 RAM 控制台，在RAM 角色管理 页面找到角色 ecs-admin，单击其角色名称。
2. 单击信任策略管理，删除 `acs:ram::12345678:root` 行。

> 说明:
> 撤销此权限也可以通过云账号 A 在 RAM 角色管理 页面删除 RAM 角色：ecs-admin；但在删除角色前，角色不能有任何权限策略。

5.4 云上应用的动态身份与授权管理

本文介绍了当用户购买阿里云产品后，通过 RAM 进行动态身份与授权管理，允许应用程序通过获取角色身份的动态令牌来访问云服务 API，以解决保密性和运维问题。

背景信息

用户购买了 ECS 实例，并且打算在 ECS 中部署企业的应用程序。这些应用程序需要使用 Access Key 访问其它云服务 API。有两种做法：

- 将 Access Key 直接嵌入在代码里。
- 将 Access Key 保存在应用程序的配置文件中。

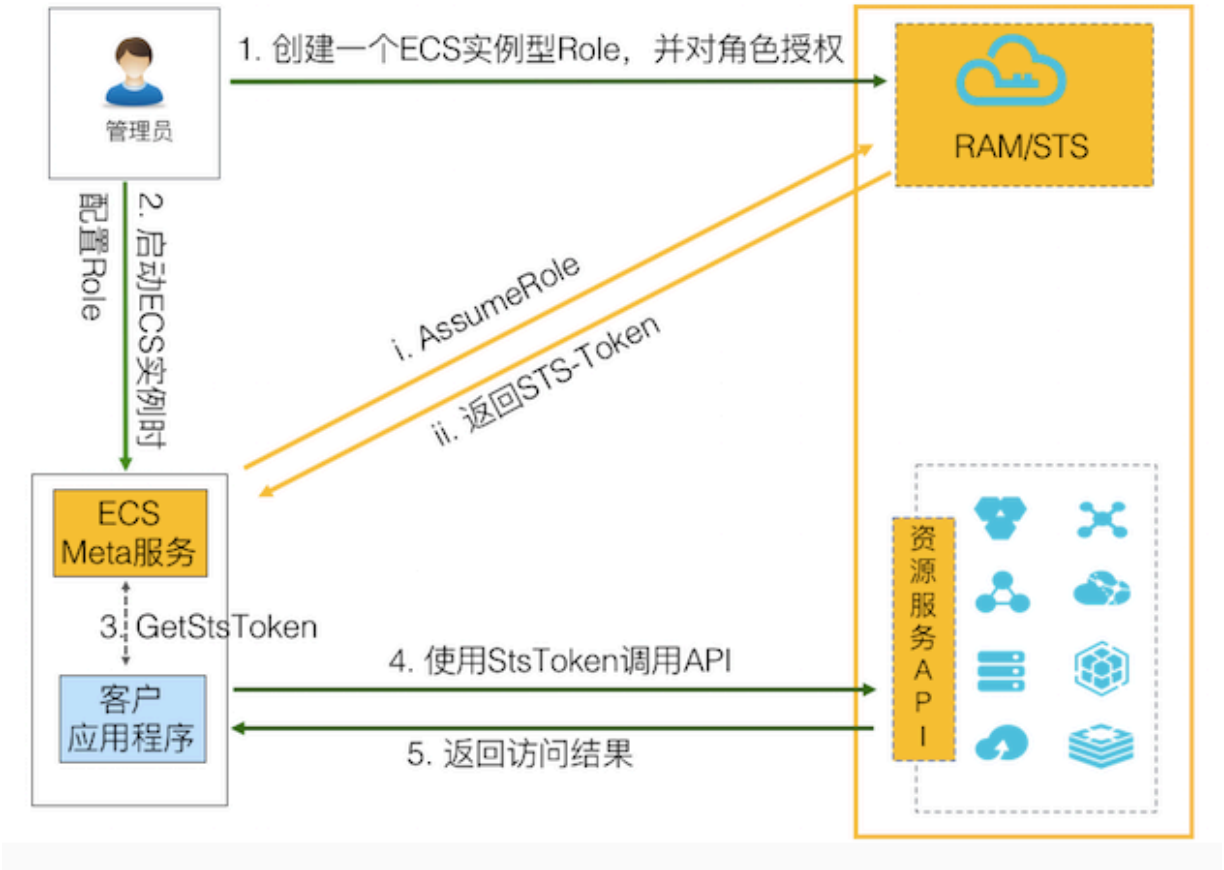
这样会带来两个问题：

- 保密性问题：如果 Access Key 以明文形式存在于 ECS 实例中，它都可能随着快照、镜像及镜像创建出来的实例被泄露。
- 难运维性问题：由于Access Key存在于实例中，如果要更换 Access Key（比如周期性轮转或切换用户身份），那么需要对每个实例和镜像进行更新并重新部署，这会增加对实例和镜像管理的复杂性。

动态身份与授权解决方案

ECS 服务结合 RAM 提供的访问控制能力，允许给每一个 ECS 实例（即用户应用程序的运行环境）配置一个拥有合适权限的 RAM 角色身份，应用程序通过获取该角色身份的动态令牌来访问云服务 API。

图 5-4: 工作原理



给 ECS 实例配置 RAM 角色的操作步骤

1. 云账号在 RAM 中创建一个 ECS 实例型 RAM 角色，并对角色授予合适的权限策略。

> 说明:

ECS 实例型 RAM 角色：是 RAM 服务角色中的一种类型，表示该角色是由用户创建并授权给该用户的 ECS 实例所使用。

2. 启动 ECS 实例时，配置创建好的 RAM 角色。
   - （i）ECS 服务根据所配置的 RAM 角色，调用 AssumeRole 去访问 STS 请求获取该角色的 STS token。
   - （ii）STS 服务会验证 ECS 服务身份及该角色的授权类型，验证通过后颁发 STS token，否则拒绝请求。

   详情请参考：*#unique_85* 或 *#unique_86*。
3. 获取到 STS token 后，ECS 将通过 Metadata 服务将 STS token 提供给实例中的应用程序访问。

   例如：在 Linux 中执行如下命令，即可获取 STS token 及过期时间等元数据信息。

```
$ curl http://100.100.100.200/latest/meta-data/ram/security-credentials/<roleName>
```

说明:

- STS token 过期时间通常为 1 小时，STS token 在有效期内都能正常访问云服务 API，在过期之前 ECS 服务会自动刷新 STS token。
- 如果 STS token 权限不足，那么需要找管理员给实例 RAM 角色添加足够的权限。
- 实例 RAM 角色的权限更新后，STS token 权限立即生效，无需重新启动 ECS 实例。

4. 使用 STS token 调用云服务 API。

说明:

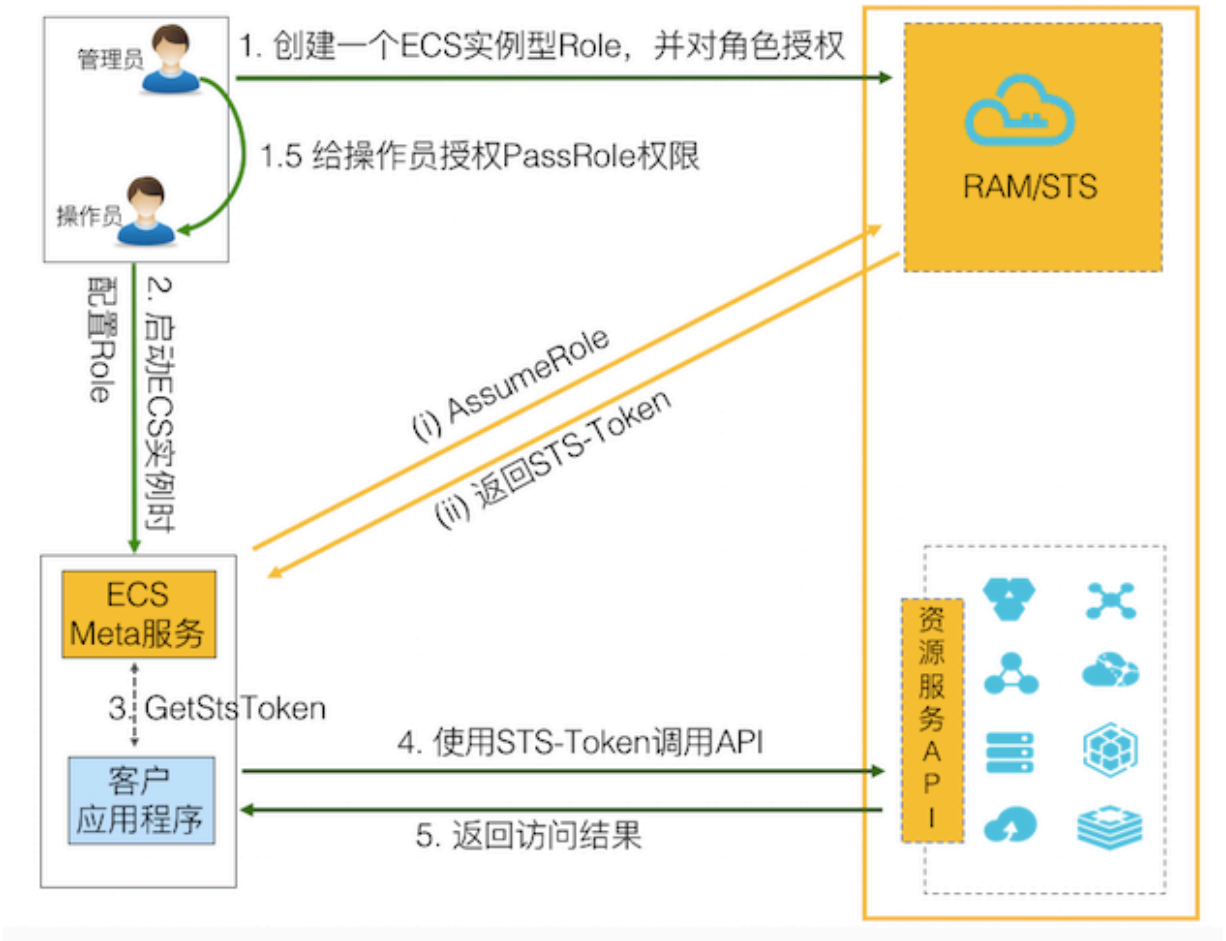
如果你的应用程序使用了阿里云 SDK，那么阿里云 SDK 将会自动从 ECS Metadata 服务中获取实例 RAM 角色的 STS token，开发者无需在 SDK 中配置任何 AccessKey 相关的信息。

详情请参考：*#unique_87*。

管理员与操作员职责分离的原理

对于很多企业用户，授权者和 ECS 实例操作者通常都是职责分离，是不同的 RAM 用户。针对管理员与操作员职责分离原理如下：

图 5-5: 管理员与操作员职责分离原理图



管理员与操作员职责分离的操作步骤

注意:

- 如果 RAM 用户不拥有管理员权限，仅有 ECS 权限。在创建 ECS 实例并配置 RAM 角色时，ECS服务会强制检查当前用户是否拥有指定 RAM 角色的 `ram:PassRole` 权限，否则无法成功创建 ECS 实例。
- 只有被授权用户才能为 ECS 实例配置 RAM 角色，避免 RAM 角色权限被滥用。

若想实现管理员与操作员职责分离，只需在给 *ECS* 实例配置 *RAM* 角色的操作步骤的基础上，根据 Step 1.5 使管理员给操作员增加一个 `PassRole` 权限即可。

管理员可以通过 RAM 按如下 Policy 示例创建一个自定义策略，然后将这个自定义策略授权给操作员。

注意:

替换 `rolename` 为自己的 RAM 角色名称。

```
{
   "Statement": [
    {
      "Effect": "Allow",
      "Action": "ram:PassRole",
      "Resource": "acs:ram:*:*:role/<rolename>"
    }
  ],
  "Version": "1"
}
```

更多信息

除了 ECS 服务之外，阿里云其它计算类服务如函数计算、MaxCompute 也都提供了类似的 RAM 角色访问能力，以帮助用户解决云上身份密钥管理的问题。

5.5 RAM 资源分组与授权

若您的企业存在多种云资源，通过创建资源组可以统一进行云资源分组，以及组内成员、权限和资源的独立管理。

背景信息

某游戏公司 A 正在开发 3 款游戏项目，每个游戏项目都会用到多种云资源。目前公司 A 只有 1 个账号，该账号下有超过 100 个 ECS 实例。

需要满足以下几个要求：

- 项目独立管理：每个管理员各自能够独立管理项目人员及其访问权限。
- 按项目分账：财务部门希望能够根据项目进行出账，以解决财务成本分摊的问题。
- 共享底层网络：客户希望云资源的底层网络默认共享。

需求分析

- 若选择多账号方案
  - 可以满足项目独立管理：公司 A 注册 3 个账号（对应 3 个项目），每个账号有对应项目管理员可以独立管理成员及其访问权限。
  - 可以满足按项目分账：每个账号有默认账单，可以利用阿里云提供的多账号合并记账能力（consolidated billing）来解决统一账单和发票问题。
  - 无法满足共享底层网络：账号之间是有安全边界的，不同账号之间的资源是 100% 隔离的，网络之间默认不通。虽然可以通过 VPC-Peering 来打通跨账号的 VPC 网络，但会带来较高的管理成本。
- 若选择单账号给资源打标签方案
  - 无法满足项目独立管理：给资源打标签可以模拟项目分组，但无法解决项目管理员独立管理项目成员及其访问权限。
  - 可以满足按项目分账：按照项目组给资源打上对应标签，根据标签实现分账。
  - 可以满足共享底层网络：公司 A 只用 1 个账号，根据项目打不同的项目标签，结合 RAM 提供的基于标签的条件授权能力，可以将一组资源授权给某些 RAM 用户，不存在打通网络所需的额外管理成本。

综合考虑，资源组是阿里云提供的一种资源分组管理能力，资源组管理方案可以满足上述所有要求。

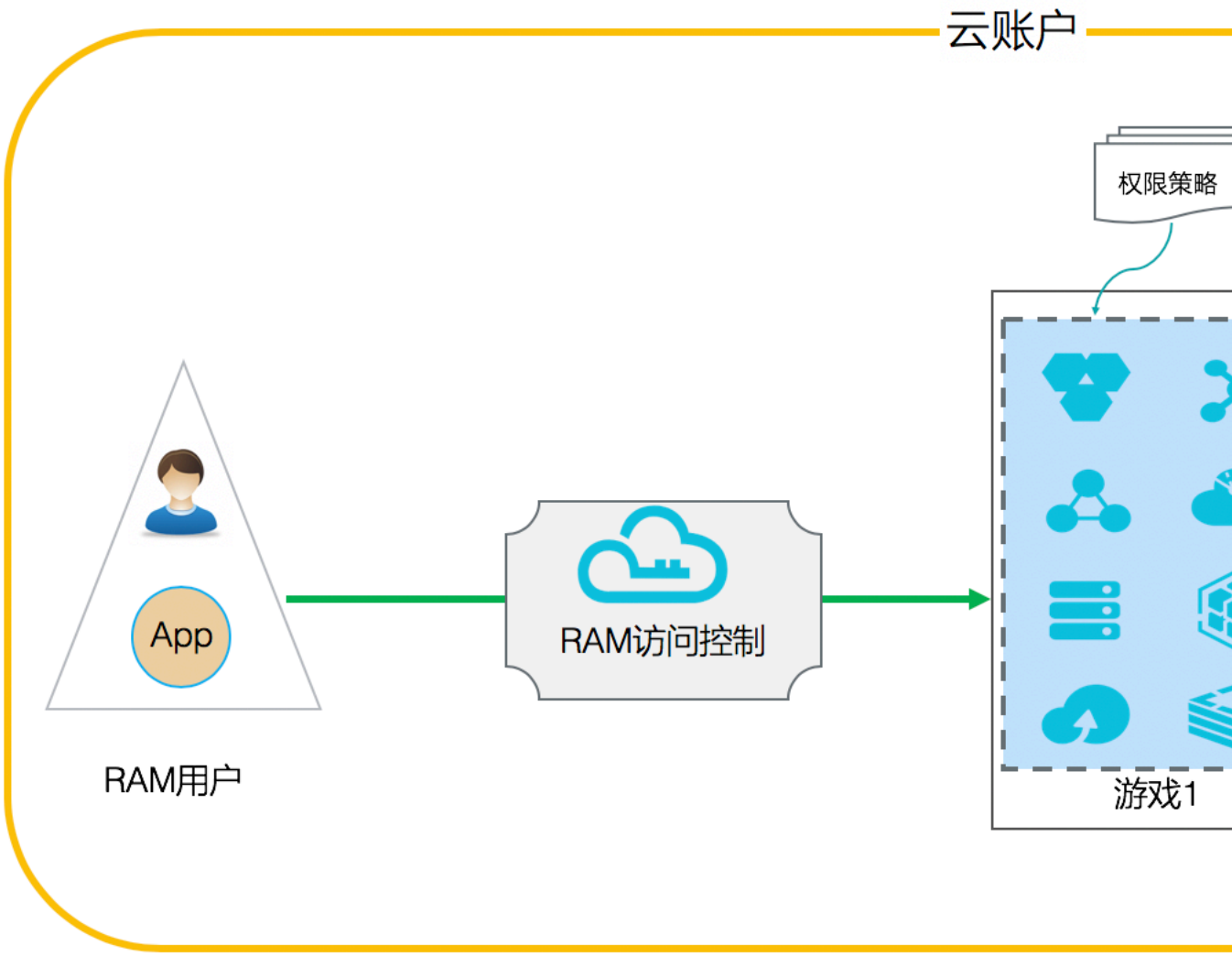
资源组管理方案

此案例中公司 A 只使用 1 个账号，创建 3 个资源组（对应 3 个项目）。

- 可以满足项目独立管理：每个资源组有对应的管理员，资源组管理员可以独立管理成员及其访问权限。
- 可以满足按项目分账：账单管理功能支持按资源组进行分账，解决财务成本分摊的问题。

- 可以满足共享底层网络：资源组属于账号内部的分组功能，同一账号下的不同资源组可以共享同一个 VPC 网络，节约管理成本。

图 5-6: 资源组内授权模型



设置管理员的操作步骤

前提条件：

- 请确保已拥有 RAM 账号并登录 RAM 控制台。
- 为 3 个管理员创建 RAM 账号：`alice@secloud.onaliyun.com`、`bob@secloud.onaliyun.com`和`charlie@secloud.onaliyun.com`。

  详情请参考：创建 RAM 用户。

下面将介绍如何 Alice 设置为项目的管理员并授予`AdministratorAccess`的系统策略。

1. 在企业控制台的概览页面下，单击资源管理下的立即使用。
2. 在资源组界面，单击创建资源组。
3. 输入标识和显示名后，单击确定。

> 说明:
> 创建 3 个资源组：游戏 1、游戏 2、游戏 3。

4. 单击管理，在成员页签下，单击新增成员。
5. 在成员处输入 Alice，权限策略处单击选择，单击`AdministratorAccess`，单击确定。

> 说明:
> 如何将 Bob、Charlie进行设置为管理员请参考上述步骤进行操作。

设置为管理员后的操作结果

以 Alice 为例：由于 Alice 是游戏 1 的资源组管理员，有以下权限：

- 登录 ECS 控制台，可以看到游戏 1 资源组，并可以创建和管理 ECS 实例。
- 登录企业控制台，可以在成员管理中添加其它 RAM 用户并授权资源访问权限。

5.6 使用企业本地账号登录阿里云

本文主要介绍如何在阿里云上使用企业内部账号进行统一的身份认证，实现使用企业本地账号登录阿里云才能访问相应资源。

背景信息

企业 A 有两个部门上云，每个部门各自独立创建了云账号 A1和 A2。企业 A 有自己的本地域账号系统（假设是使用 Microsoft AD 和 AD FS 服务）。

企业 A 对 IAM 安全管理有非常严苛的合规要求：

- 所有云上操作都必须使用企业内部域账号系统进行统一的身份认证，禁止任何企业成员使用独立用户账号和密码直接操作云资源。
- 企业 A 希望云上用户与企业本地账号系统集成，所有员工必须使用企业本地账号登录然后才能访问被授权的云资源。

解决思路

前提条件：

进行操作前需先了解本次操作涉及到的相关术语：

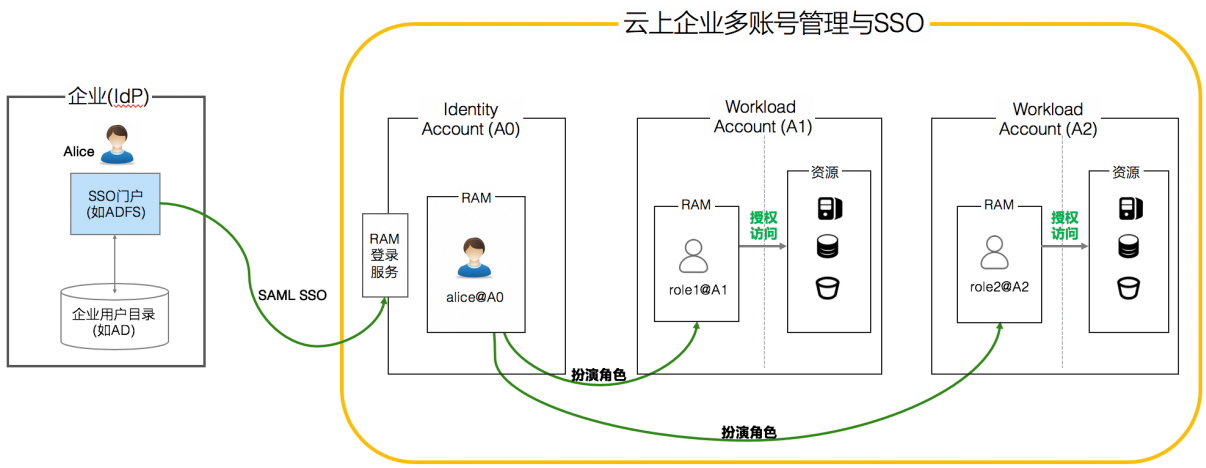
术语	描述
资源云账号（Workload Account）	云账号下可以购买 ECS 实例、OSS 实例、RDS 实例等云资源。
身份云账号（Identity Account）	云账号下只创建 RAM 用户。
服务提供商（Service Provider）	利用 IdP 的身份管理功能，为用户提供具体服务的应用，SP 会使用 IdP 提供的用户信息。

基本思路：

如果企业的两个云账号都属于资源云账号，请按照如下思路进行操作：

1. **创建一个独立的身份云账号。**
2. **将身份云账号当做服务提供商与企业本地 IdP 进行 SSO 集成。**
3. **利用阿里云 RAM 提供的跨账号 RAM 角色的授权访问能力进行跨账号访问其他云账号资源。**

图 5-7: 基本原理



云上多账号管理的操作步骤

1. **注册一个新的云账号 A0 作为身份云账号。**

   说明：

   A0 主要用于解决用户同步及配置 SSO 单点登录，用以区别于资源云账号。

2. **使用 A0 登录 RAM 控制台配置 SSO 单点登录。**
3. **在企业本地 AD FS 服务中配置 A0 为服务提供商。**
4. **同步本地用户到 A0：将需要访问云资源的本地部门用户同步到 RAM。**
5. **在资源云账号A1 中创建跨账号的 RAM 角色 1，给 RAM 角色 1 设置合适的资源访问授权，并设置 A0 为受信云账号。**

6. 在资源云账号A2 中创建跨账号的 RAM 角色 2，给 RAM 角色 2 设置合适的资源访问授权，并设置 A0 为受信云账号。
7. A0 授权 RAM 用户 Alice 拥有扮演 A1 或 A2 中对应角色的权限。