Red Hat Single Sign-On 7.6

服务器管理指南

用于 Red Hat Single Sign-On 7.6

Last Updated: 2024-06-05

Red Hat Single Sign-On 7.6 服务器管理指南

用于 Red Hat Single Sign-On 7.6

法律通告

摘要

本指南由管理员配置 Red Hat Single Sign-On 7.6 的信息

目录

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。有关更多详情，请参阅我们的首席技术官 Chris Wright 提供的消息。

第 1 章 红帽单点登录功能和概念

Red Hat Single Sign-On 是适用于 Web 应用和 RESTful Web 服务的解决方案的单点登录。Red Hat Single Sign-On 的目标是使安全性变得简单，以便应用程序开发人员能够轻松保护他们在其组织中部署的应用和服务。开发人员通常必须自己写入的安全功能是开箱即用的，可轻松根据您的组织的特定需求进行定制。Red Hat Single Sign-On 为登录、注册、管理和帐户管理提供可自定义的用户界面。您还可以使用 Red Hat Single Sign-On 作为集成平台，将其转换为现有 LDAP 和 Active Directory 服务器。您还可以将身份验证委派给第三方身份提供程序，如 Facebook 和 Google。

1.1. 功能

Red Hat Single Sign-On 提供以下功能：

- 单Sign On 和 Single-Sign Out 用于浏览器应用程序。
- OpenID Connect 支持.
- OAuth 2.0 支持。
- SAML 支持。
- 身份代理 - 通过外部 OpenID Connect 或 SAML 身份提供程序进行身份验证。
- 社交登录 - 支持通过 Google、GitHub、Facebook、Twitter 和其他社交网络登录。
- 用户 Federation - 从 LDAP 和 Active Directory 服务器同步用户。
- Kerberos bridge - 自动验证登录到 Kerberos 服务器的用户。
- 用于集中管理用户、角色、角色映射、客户端和配置的管理控制台。
- 允许用户集中管理帐户的帐户管理控制台。
- 这个主题支持 - 自定义所有用户面向用户的页面，以便与您的应用程序和品牌集成。
- 双因素身份验证 - 通过 Google Authenticator 或 FreeOTP 支持 TOTP/HOTP。
- 登录流 - 可选用户自我注册、恢复密码、验证电子邮件、需要密码更新等。
- 会话管理 - 管理员和用户本身可以查看和管理用户会话。
- 令牌映射程序 - 映射用户属性、角色等。如何融入令牌和语句。
- 各个域、应用和用户前的撤销策略。
- CORS 支持 - 客户端适配器对 CORS 内置支持。
- JavaScript 应用程序、JBoss EAP 等的客户端适配器。
- 支持具有 OpenID Connect 独立库或 SAML 2.0 服务提供商库的任何平台/语言。

1.2. 基本 RED HAT SINGLE SIGN-ON 操作

Red Hat Single Sign-On 是您在网络上管理的独立服务器。应用程序被配置为指向 并受此服务器保护。Red Hat Single Sign-On 使用 OpenID Connect 或 SAML 2.0 等开放式协议标准来保护您的应用程序。浏览器应用将用户浏览器从应用程序重定向到 Red Hat Single Sign-On 身份验证服务器，在其中输入自己的

凭据。这个重定向很重要，因为用户完全与应用程序隔离，应用程序永远不会看到用户的凭据。相反，应用程序会被赋予一个以加密方式签名的身份令牌或断言。这些令牌可以有身份信息，如用户名、地址、电子邮件和其他配置集数据。它们也可以保存权限数据，以便应用程序可以做出授权决策。这些令牌也可用于在基于 REST 的服务上进行安全调用。

1.3. 核心概念和术语

在使用 Red Hat Single Sign-On 来保护 Web 应用程序和 REST 服务之前，请考虑这些核心概念和术语。

users

用户是能够登录到您的系统的实体。他们可以具有与自己关联的属性，如电子邮件、用户名、地址、电话号码和 birth 天。可以为用户分配组成员资格，并为它们分配特定角色。

身份验证

识别和验证用户的过程。

授权

授予用户访问权限的过程。

credentials

凭据是 Red Hat Single Sign-On 用来验证用户身份的数据片段。些示例包括密码、一次性密码、数字证书甚至指纹。

roles

角色标识用户的类型或类别。**Admin**, **user**, **manager**, 和 **employee** 是一个机构中的典型的职位。应用通常会将访问权限和权限分配给特定角色，而非处理用户的个人用户，而对于处理用户来说过于精细和难以管理。

用户角色映射

用户角色映射定义了角色和用户之间的映射。个用户可以与零个或多个角色关联。此角色映射信息可以封装到令牌和断言中，以便应用程序可以决定访问其管理的各种资源的权限。

复合角色

复合角色是可与其他角色关联的角色。例如，**超级用户** 复合角色可以关联 **sales-admin** 和 **order-entry-admin** 角色。如果用户映射到 **superuser** 角色，则他们也会继承 **sales-admin** 和 **order-entry-admin** 角色。

groups

组管理用户组。可以为组定义属性。您还可以将角色映射到组。成为组成员的用户继承组所定义的属性和角色映射。

realms

realm 管理一组用户、凭据、角色和组。用户从属于并登陆到的域。域彼此隔离，只能管理和验证其控制的用户。

客户端

客户端是可以请求 Red Hat Single Sign-On 对用户进行身份验证的实体。大多数情况下，客户都是希望使用 Red Hat Single Sign-On 的应用程序和服务来保护自身并提供单点登录解决方案。客户端也可以是仅请求身份信息或访问令牌的实体，以便它们能够安全地调用由 Red Hat Single Sign-On 保护的网络上的其他服务。

客户端适配器

客户端适配器是安装到应用程序环境的插件，以便可以通过 Red Hat Single Sign-On 进行通信并加以保护。Red Hat Single Sign-On 有许多适配器，供您下载的不同平台。此外，您还可以为我们所不涵盖的环境提供第三方适配器。

同意

当管理员希望用户向客户端授予其权限时，同意将权限授予客户端，然后客户端才能参与身份验证过程。用户提供凭证后，Red Hat Single Sign-On 将弹出一个屏幕，标识请求登录的客户端以及向用户请求哪些身份信息。用户可以决定是否授予请求。

客户端范围

注册客户端后，您必须为该客户端定义协议映射和角色范围映射。存储客户端范围通常很有用，通过共享一些常见设置来更轻松地创建新客户端。这也可用于根据 **scope** 参数的值来有条件地请求某些声明或角色。Red Hat Single Sign-On 提供了此客户端范围的概念。

客户端角色

客户端可以定义特定于它们的角色。这基本上是一个专用于客户端的角色命名空间。

身份令牌

此令牌提供与用户相关的身份信息。OpenID Connect 规范的一部分。

访问令牌

作为 HTTP 请求的一部分提供的令牌，用于授予对正在调用的服务的访问权限。这是 OpenID Connect 和 OAuth 2.0 规范的一部分。

assertion

有关用户的信息。这通常与 SAML 身份验证响应中包含的 XML blob 相关，后者提供有关经过身份验证的用户提供身份元数据。

服务帐户

每个客户端都有一个内置服务帐户，允许它获取访问令牌。

直接授权

客户端通过 REST 调用为用户授予访问令牌的方法。

协议映射程序

对于每个客户端，您可以定制将哪些声明和断言存储在 OIDC 令牌或 SAML 断言中。您可以通过创建和配置协议映射程序来为每个客户端执行此操作。

会话

当用户登录时，会创建一个会话来管理登录会话。会话包含用户登录以及在该会话中参与了什么应用程序时的信息。管理员和用户都可以查看会话信息。

用户联合供应商

红帽单点登录可存储和管理用户。通常，公司已经拥有用于存储用户和凭证信息的 LDAP 或 Active Directory 服务。您可以指出 Red Hat Single Sign-On 来验证来自这些外部存储的凭证，并拉取身份信息。

身份供应商

身份供应商(IDP)是能够验证用户身份的服务。红帽单点登录是一个 IDP。

身份提供程序联合

Red Hat Single Sign-On 可以被配置为将身份验证委派给一个或多个 IDP。通过 Facebook 或 Google+ 的社交登录是身份供应商联合的示例。您还可以 hook Red Hat Single Sign-On 将身份验证委派给任何其他 OpenID Connect 或 SAML 2.0 IDP。

身份提供程序映射器

在进行 IDP 联合时，您可以将传入的令牌和断言映射到用户和会话属性。这有助于将您的身份信息从外部 IDP 传播到请求身份验证的客户端。

所需的操作

所需操作是用户必须在身份验证过程中执行的操作。用户在完成这些操作之前无法完成身份验证过程。例如，管理员可以计划用户在每次月重置其密码。会为所有这些用户设置 **更新密码** 所需的操作。

身份验证流程

身份验证流是在与系统的某些方面交互时，用户必须执行的工作流。登录流程可以定义所需的凭证类型。注册流程定义用户必须输入的配置集信息，以及诸如 reCAPTCHA 的问题都必须用于过滤 bots。凭据重置流程定义用户必须先执行的操作，然后才能重置密码。

事件

事件是管理员可以在其中查看和 hook 的审核流。

themes

由 Red Hat Single Sign-On 提供的每个页面均由主题提供支持。主题定义了 HTML 模板和风格表，您可以根据需要进行覆盖。

第 2 章 创建第一个管理员

安装 Red Hat Single Sign-On 后，您需要管理员帐户作为 *超级* 管理员，并具有完全权限来管理 Red Hat Single Sign-On 的所有部分。使用这个帐户，您可以登录到 Red Hat Single Sign-On Admin Console，在其中创建域和用户，并注册由红帽单点登录保护的应用程序。

先决条件

- 执行 服务器安装与配置指南 中定义的安装和配置任务，使其指向 Red Hat Single Sign-On 服务器正在运行。

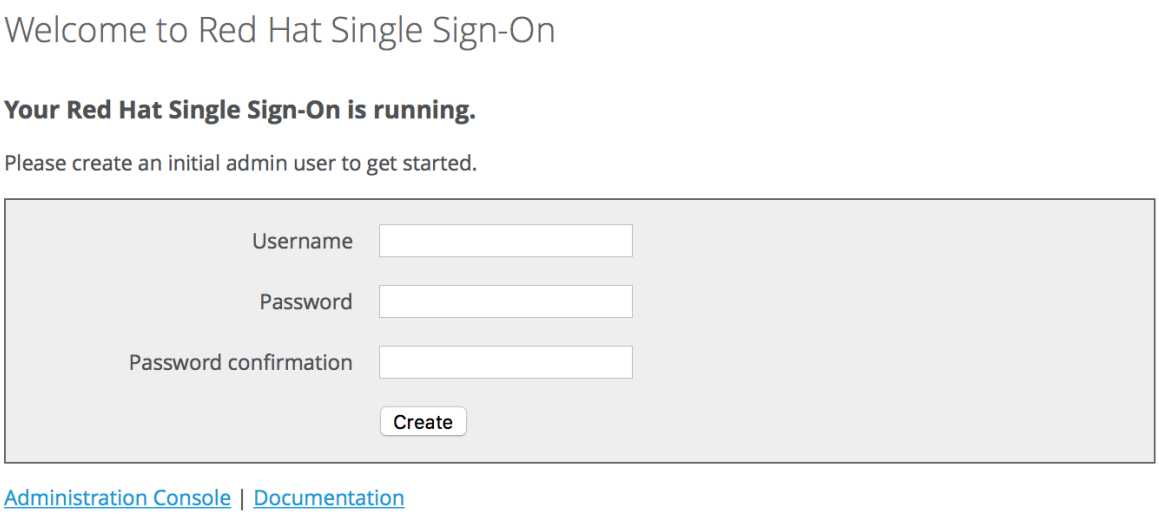
2.1. 在本地主机上创建帐户

如果可以从 **localhost** 访问您的服务器，请执行以下步骤。

流程

1. 在 Web 浏览器中，访问 http://localhost:8080/auth URL。
2. 提供您可以重新调用的用户名和密码。

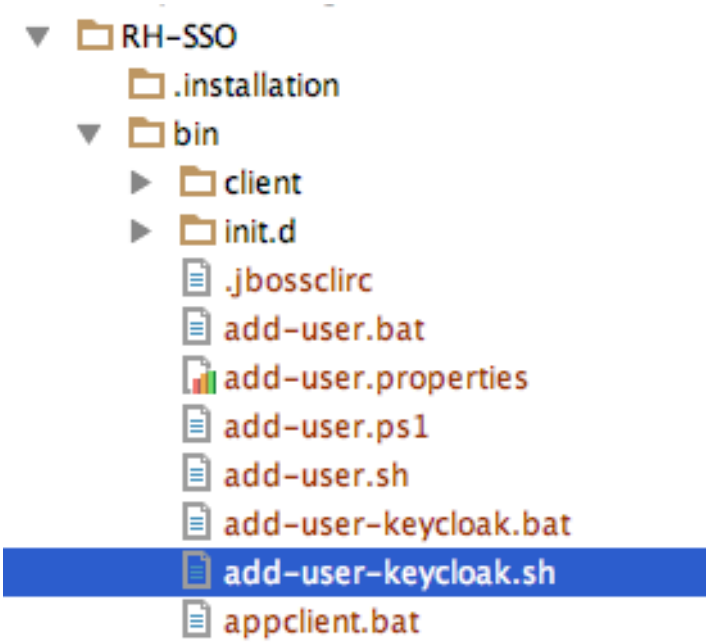
欢迎页面



2.2. 远程创建帐户

如果您无法从 **localhost** 地址访问服务器，或者只希望从命令行启动 Red Hat Single Sign-On，请使用 **.../bin/add-user-keycloak** 脚本。

add-user-keycloak 脚本



这些参数略有不同，具体取决于您使用独立操作模式或域操作模式。对于单机模式，以下是使用 脚本。

Linux/Unix

```
$ .../bin/add-user-keycloak.sh -r master -u <username> -p <password>
```

Windows

```
> ...\bin\add-user-keycloak.bat -r master -u <username> -p <password>
```

生成的文件由与运行 Red Hat Single Sign-On 的用户不同的用户所有。使用此命令设置权限，以便 Red Hat Single Sign-On 用户可以在重新启动服务器时读取该文件。

```
chgrp jboss /opt/rh/rh-sso7/root/usr/share/keycloak/standalone/configuration/keycloak-add-user.json
```

对于域模式，您必须使用 **-sc** 开关将脚本指向其中一个服务器主机。

Linux/Unix

```
$ .../bin/add-user-keycloak.sh --sc domain/servers/server-one/configuration -r master -u <username>
-p <password>
```

Windows

```
> ...\bin\add-user-keycloak.bat --sc domain/servers/server-one/configuration -r master -u
<username> -p <password>
```

第 3 章 配置域

具有 Admin Console 的管理帐户后，您可以配置 realms。realm 是您管理对象（包括用户、应用程序、角色和组）的空间。用户从属于并登陆到的域。一个 Red Hat Single Sign-On 部署可以在数据库中定义、存储和管理许多领域。

3.1. 使用管理控制台

您可以在 Red Hat Single Sign-On Admin 控制台中配置域并执行大多数管理任务。

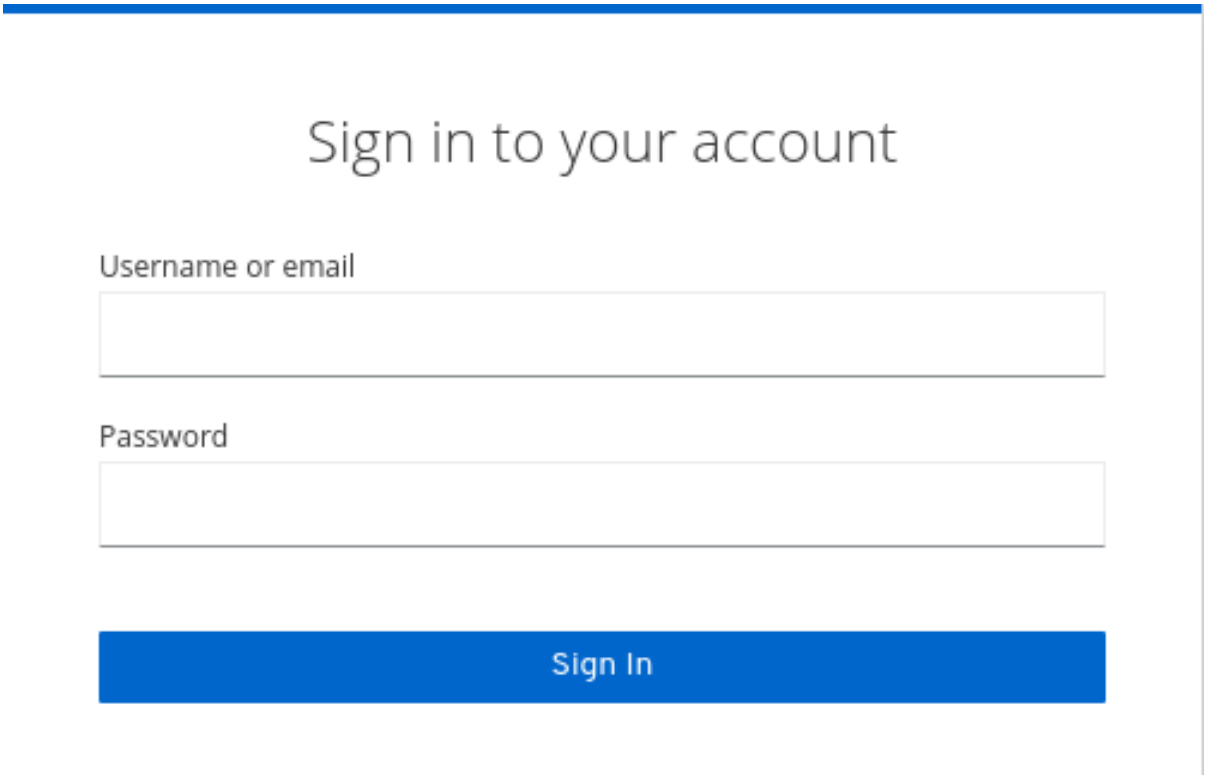
先决条件

- 您需要管理员帐户。请参阅 创建第一个管理员。

流程

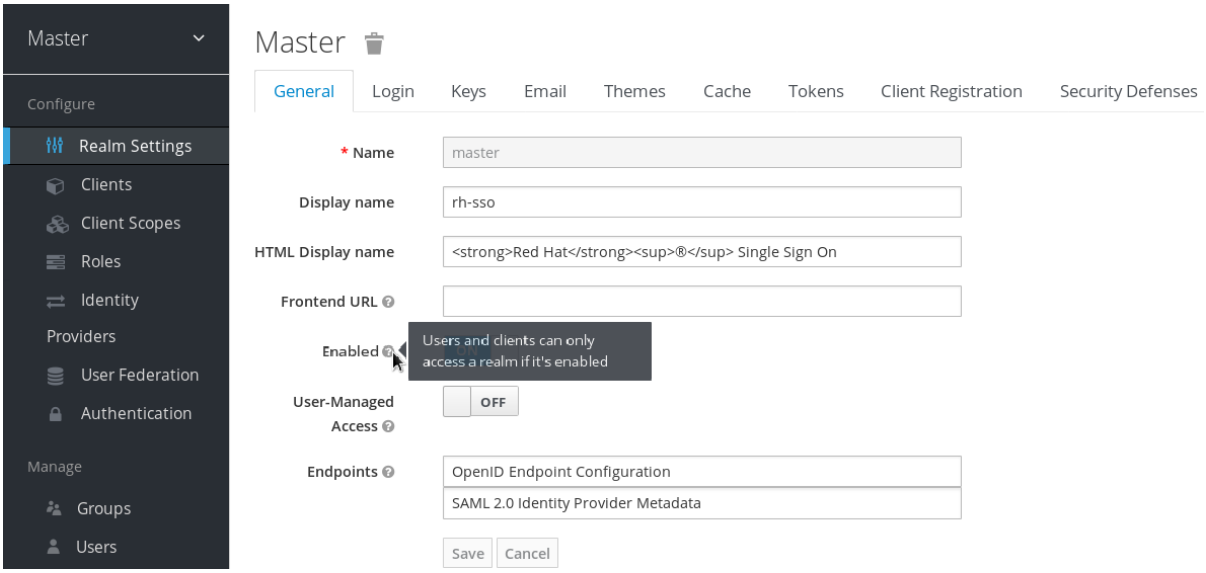
1. 前往 Admin Console 的 URL。
   例如，对于 localhost，使用这个 URL: http://localhost:8080/auth/admin/

   登录页面



2. 在 bin 目录中，输入您在 Welcome Page 或 **add-user-keycloak** 脚本中创建的用户名和密码。此操作显示 Admin Console。

   Admin Console



3. 请注意菜单以及您可以使用的其他选项：

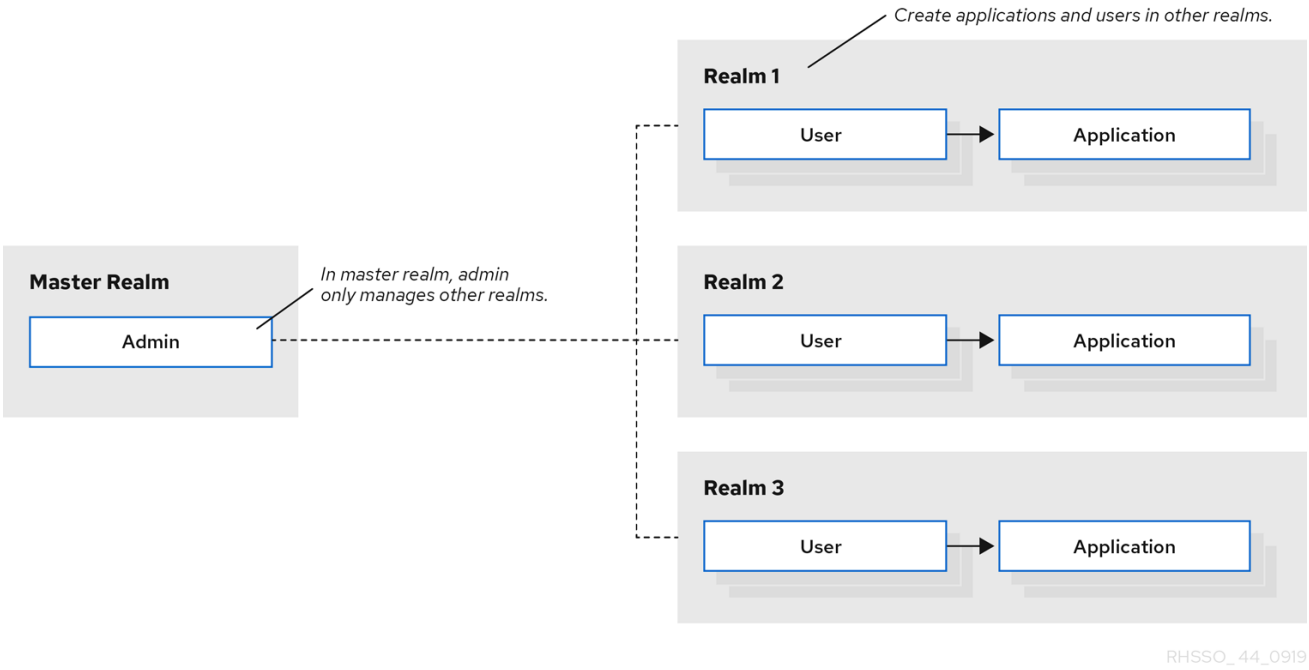
   - 点击标有 **Master** 的菜单选择您要管理的域或创建新域。
   - 点右列表查看您的帐户或注销。
   - 将鼠标悬停在问号 **?** 图标上，以显示描述该字段的工具提示文本。上图显示了操作提示。

3.2. 主域

在 Admin 控制台中，存在两种类型的域：

- **Master realm** - 首次启动 Red Hat Single Sign-On 时为您创建此域。它包含您在第一次登录时创建的管理员帐户。仅使用 *master* 域来创建和管理系统中的域。
- **其他域** - 这些域由主域中的管理员创建。在这些域中，管理员管理您的机构中的用户及其需要的应用程序。应用程序由用户所有。

realms 和应用程序



域彼此隔离，只能管理和验证其控制的用户。在此安全模型中，有助于防止意外更改并遵循允许用户帐户仅访问这些特权和完成当前任务所需的权限和电源。

其他资源

- 如果要禁用 *master* 域并在您创建的任何新域中定义管理员帐户，请参阅 Dedicated Realm Admin Consoles。每个 realm 具有自己的专用管理控制台，您可以使用本地帐户登录。

3.3. 创建域

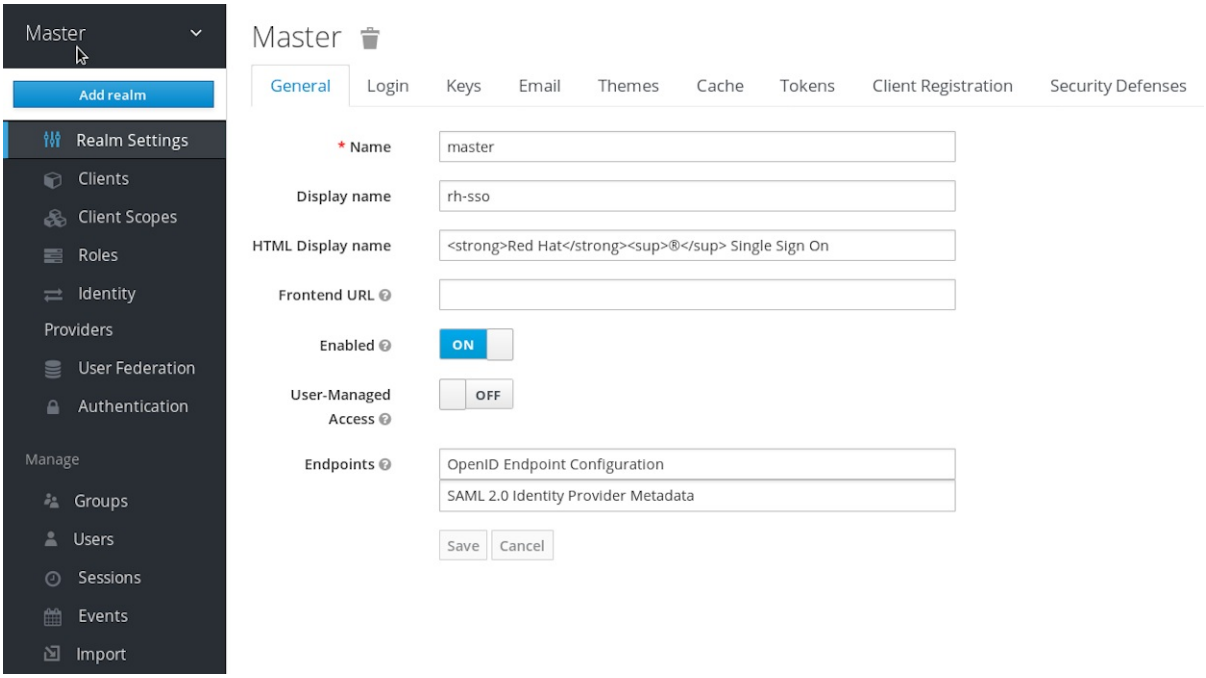
您可以创建一个域来提供管理空间，以便您可以创建用户并赋予他们使用应用程序的权限。首次登录时，您通常位于 *主域* 内，这是创建其他域的顶级域。

在决定您需要的域时，请考虑您要为用户和应用程序所需的隔离类型。例如，您可以为您的公司的员工创建一个域，并为您的客户创建一个单独的域。您的员工才能登录员工域，并只能访问内部公司应用程序。客户会登录客户域，并且只能与面向客户的应用程序进行交互。

流程

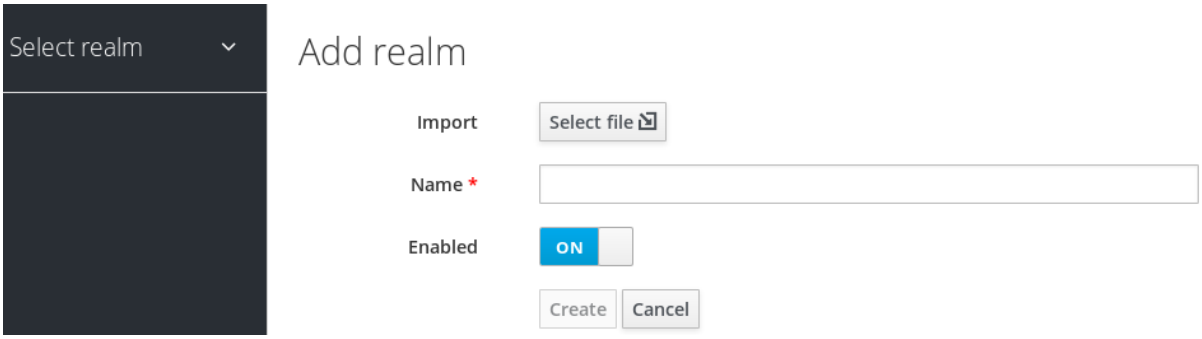
1. 指向左侧窗格的顶部。
2. 点 **Add Realm**。

添加 realm 菜单



3. 输入域的名称。
4. 点 **Create**。

创建域



当前域现在设置为您刚才创建的域。您可以通过指向上下角点 **Select Realm**，在管理不同域间切换。

3.4. 为域配置 SSL

每个域都有一个关联的 SSL 模式，用于定义与域交互的 SSL/HTTPS 要求。与域交互的浏览器和应用程序遵循 SSL 模式或者无法与服务器交互的 SSL/HTTPS 要求。

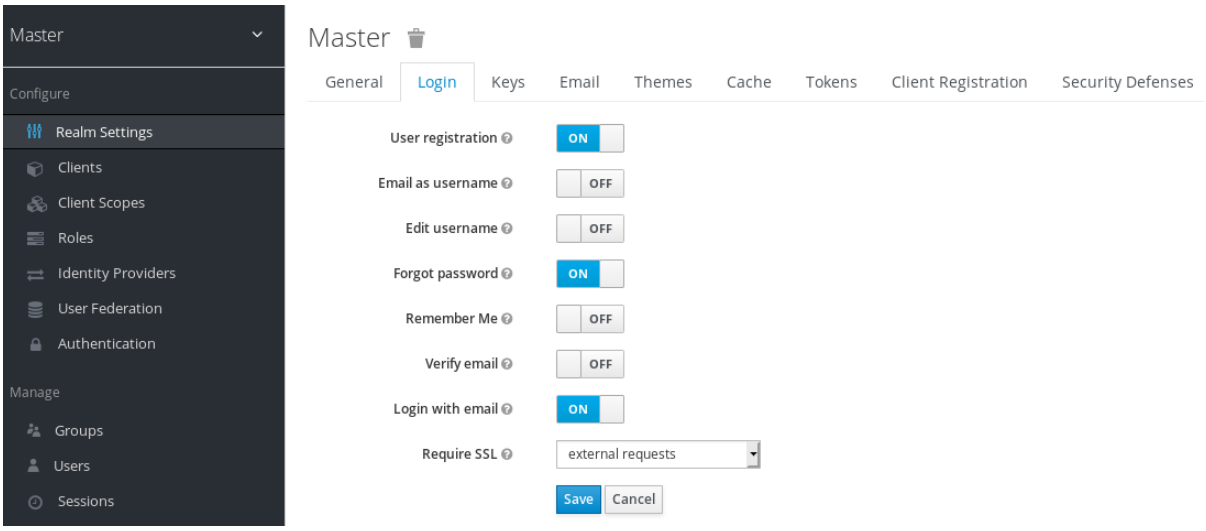
警告

Red Hat Single Sign-On 第一次生成自签名证书。请注意，自签名证书不安全，且只用于测试目的。强烈建议您在红帽单点登录服务器本身或在 Red Hat Single Sign-On 服务器前安装 CA 签名证书。请参阅" 服务器安装和配置指南 "。

流程

1. 点菜单中的 **Realm Settings**。
2. 点 **Login** 选项卡。

登录标签页



3. 将 **Require SSL** 设置为以下 SSL 模式之一：
   - 外部请求：用户可以在没有 SSL 的情况下与 Red Hat Single Sign-On 交互，只要它们坚持使用专用 IP 地址，如 **localhost**, **127.0.0.1**, **10.x.x.x**, **192.168.x.x**, 和 **172.16.x.x**。如果您试图从非私有 IP 地址在没有 SSL 的情况下访问 Red Hat Single Sign-On，则会出现错误。
   - 无：红帽单点登录不需要 SSL。只有当您进行试验且没有计划支持此部署时，这种选择才适用于开发。
   - 所有请求：红帽单点登录需要所有 IP 地址的 SSL。

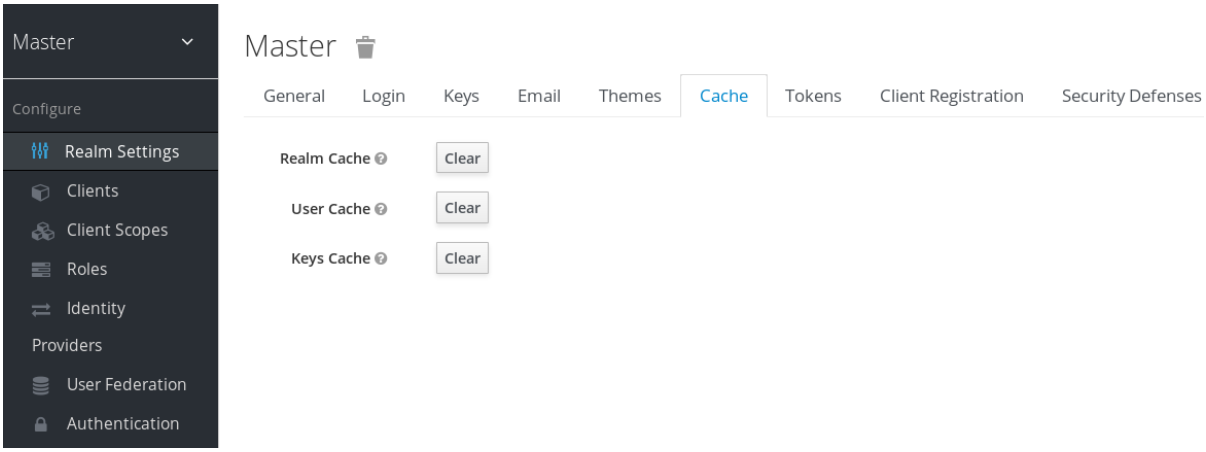
3.5. 清除服务器缓存

Red Hat Single Sign-On 会缓存它在 JVM 的限制以及您配置的限制时可以在内存中执行的所有内容。如果 Red Hat Single Sign-On 数据库由第三方（如 DBA）修改，超出服务器的 REST API 或管理控制台范围外，内存中缓存的部分可能过时。您可以清除外部公钥的域缓存、用户缓存或缓存，如外部客户端或身份提供程序的公钥，红帽单点登录可用于验证特定外部实体的签名。

流程

1. 点菜单中的 **Realm Settings**。
2. 点 **Cache** 选项卡。
3. 点击您要驱除的缓存的 **Clear**。

缓存标签页



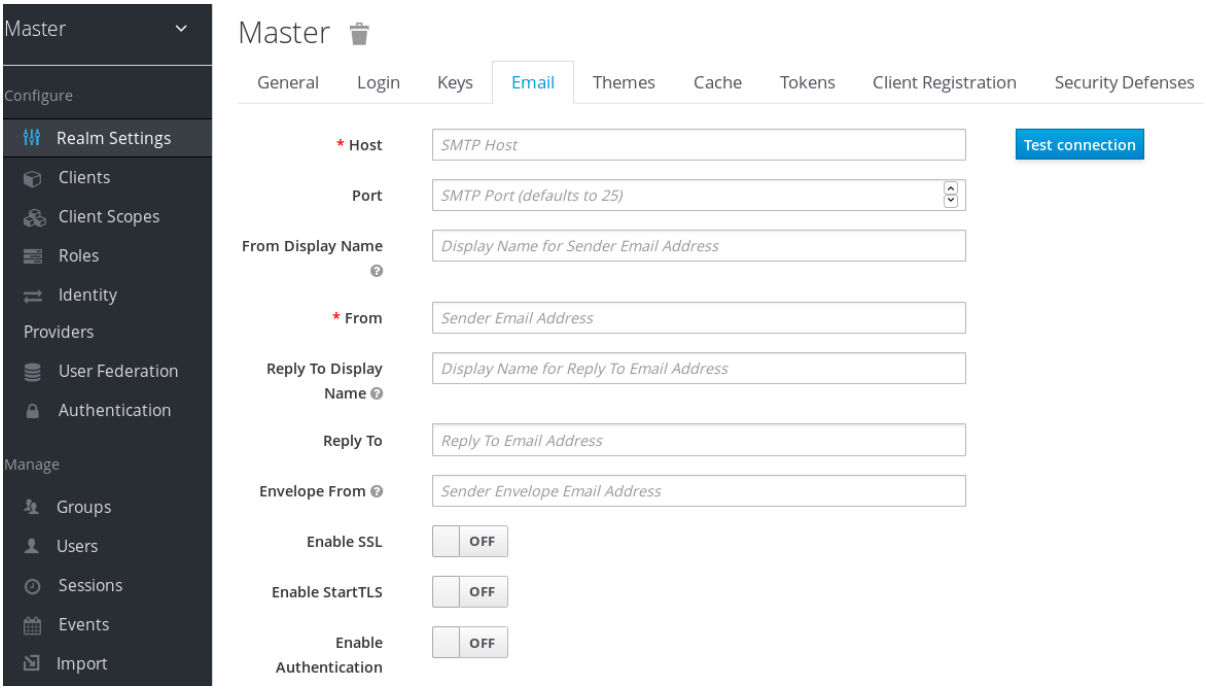
3.6. 为域配置电子邮件

Red Hat Single Sign-On 向用户发送电子邮件到用户，当他们忘记了密码时，或者管理员需要收到有关服务器事件的通知。要启用 Red Hat Single Sign-On 来发送电子邮件，您可向 Red Hat Single Sign-On 提供您的 SMTP 服务器设置。

流程

1. 点菜单中的 **Realm Settings**。
2. 单击 **电子邮件** 选项卡。

电子邮件标签页



3. 填写字段并根据需要切换切换。

主机

host 表示用于发送电子邮件的 SMTP 服务器主机名。

端口

端口 表示 SMTP 服务器端口。

从

From 表示用于发送的电子邮件的 **From** SMTP 标头使用的地址。

从 Display Name

在 Display Name 中，可以配置用户友好的电子邮件地址别名（可选）。如果没有设置纯 **From** 电子邮件地址，则会在电子邮件客户端中显示。

答复至

回复 表示用于接收邮件的 **Reply-To** SMTP-Header 的地址（可选）。如果没有设置普通 **From** 电子邮件地址。

回复显示名称

reply To Display Name 允许配置用户友好的电子邮件地址别名（可选）。如果没有设置普通的 **Reply To** 电子邮件地址，将会显示。

信封

envelope From 表示用于发送邮件的 return **-Path SMTP-** Header 的 Bounce Address （可选）。

启用 SSL 和启用 StartTSL

将其中一个交换机切换为 **ON**，以支持发送电子邮件以恢复用户名和密码，尤其是当 SMTP 服务器位于外部网络时。您可能需要将**端口**更改为 465，这是 SSL/TLS 的默认端口。

启用身份验证

如果您的 SMTP 服务器需要身份验证，请将此开关设置为 **ON**。出现提示时，提供 **Username** 和 **Password**。**Password** 字段的值可以从外部 密码库 引用值。

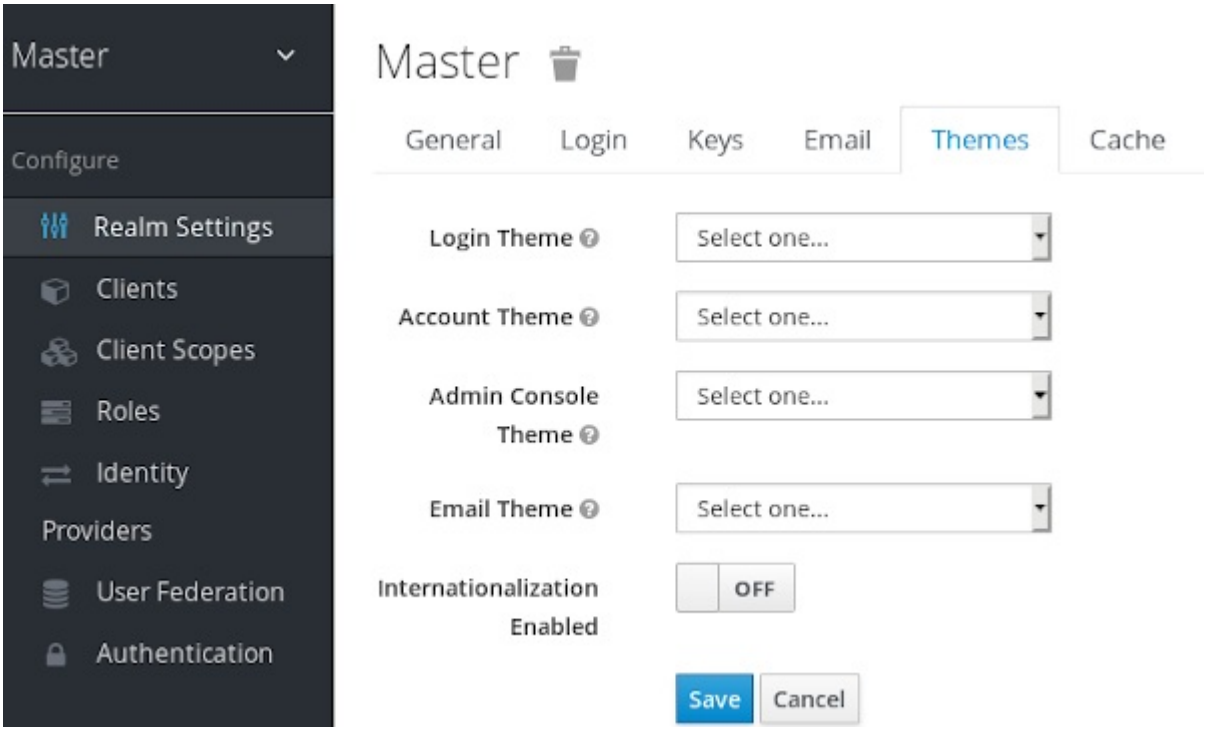
3.7. 配置它们和国际化

对于给定的域，您可以使用它们更改任何 UI 的外观，包括 Red Hat Sign-On 中的语言。

流程

1. 点菜单中的 **Realm Setting**。
2. 点 **Themes** 选项卡。

themes 选项卡



3. 选择每个 UI 类别所需的主题，然后单击 **Save**。

login Theme

用户名密码条目、OTP 条目、新用户注册和其他与登录相关的类似屏幕。

account Theme

每个用户都有一个用户帐户管理 UI。

admin Console Theme

Red Hat Single Sign-On Admin Console 的 skin。

电子邮件主题

每当 Red Hat Single Sign-On 必须收到一封电子邮件时，它将使用本主题中定义的模板制作电子邮件。

其他资源

- Server Developer Guide 描述了如何创建新主题或修改现有主题。

3.7.1. 启用国际化

每个 UI 屏幕在 Red Hat Single Sign-On 中进行国际化。默认语言为 English，但您可以选择要支持哪些区域设置以及默认的区域设置。

流程

1. 点菜单中的 **Realm Settings**。
2. 点 **Theme** 选项卡。
3. 将 **国际化** 设置为 **ON**。

用户下一次登录时，用户可以在登录页面上选择一个语言，以用于登录屏幕、帐户控制台和管理控制台。

其他资源

- 服务器开发人员指南 介绍了如何提供其他语言。由主题提供的所有国际化文本可由 **本地化** 选项卡中的特定域文本覆盖。

3.7.2. 用户区域设置

区域设置选择器供应商建议有关可用信息的最佳区域设置。但是，用户通常未知。因此，之前通过身份验证的用户区域设置被记住在一个持久的 Cookie 中。

选择区域设置的逻辑使用以下第一个可用：

- 用户选择 - 使用下拉区域设置来选择区域设置
- User profile - 当有经过身份验证的用户且用户具有首选区域设置时
- 客户端选择的客户端 - 使用示例 ui_locales 参数
- Cookie - 在浏览器中选择的最后区域设置
- 接受的语言 - **Accept-Language** 标头的区域设置
- realm default
- 如果以上都没有，则回退到英语

当用户通过身份验证后，会触发一个操作来更新之前提到的持久性 Cookie 中的区域设置。如果用户在登录页面上主动切换区域，此时还会更新用户区域设置。

如果要更改用于选择区域设置的逻辑，您可以选择创建自定义 **LocaleSelectorProvider**。详情请查看服务器开发人员指南。

3.8. 控制登录选项

Red Hat Single Sign-On 包括若干内置登录页面功能。

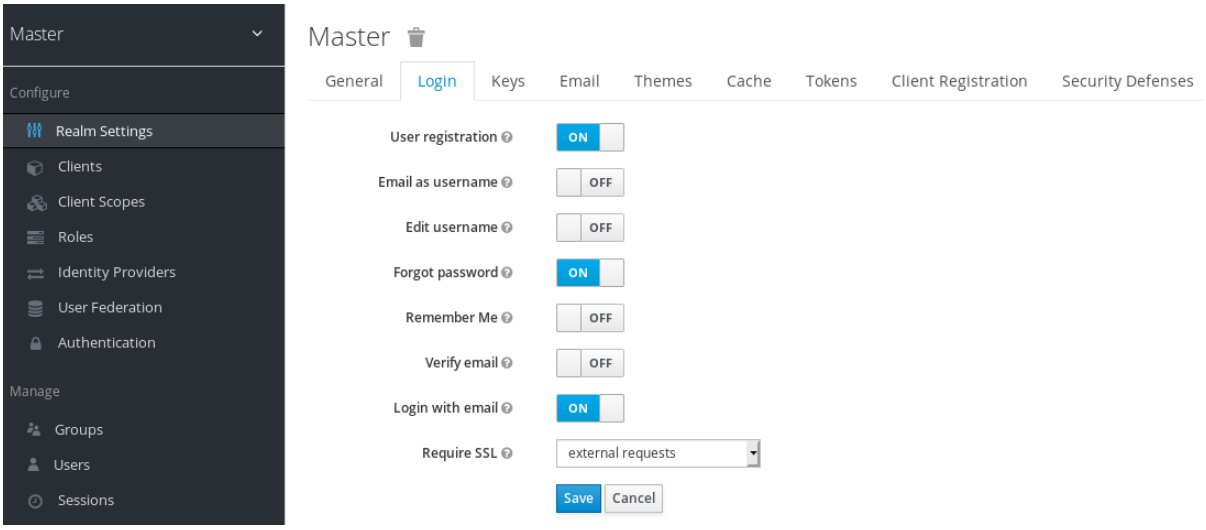
3.8.1. 启用忘记密码

如果您启用 **Forgot Password**，如果用户忘记了密码或丢失 OTP 生成器，可以重置其登录凭证。

流程

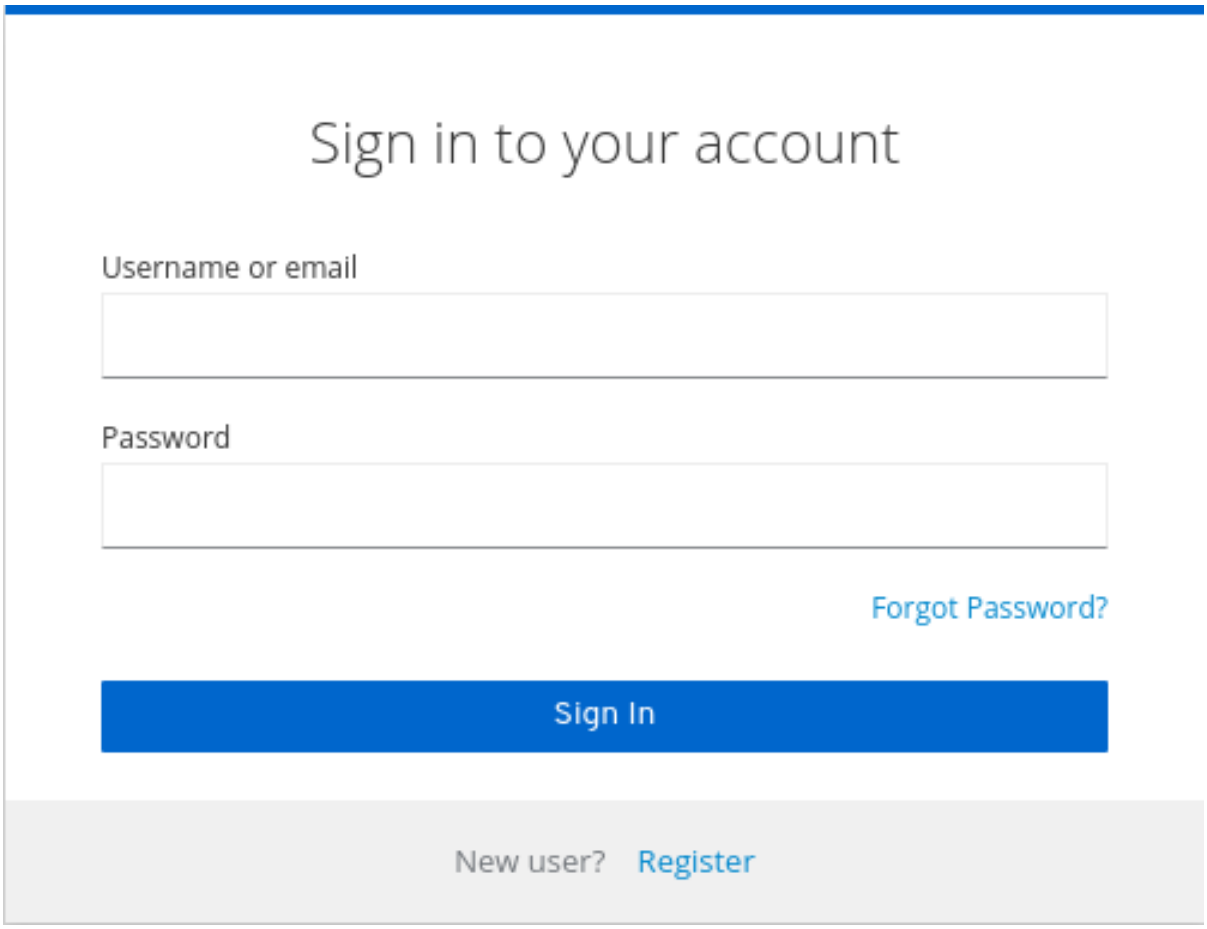
1. 点菜单中的 **Realm Settings**。
2. 点 **Login** 选项卡。

   登录标签页



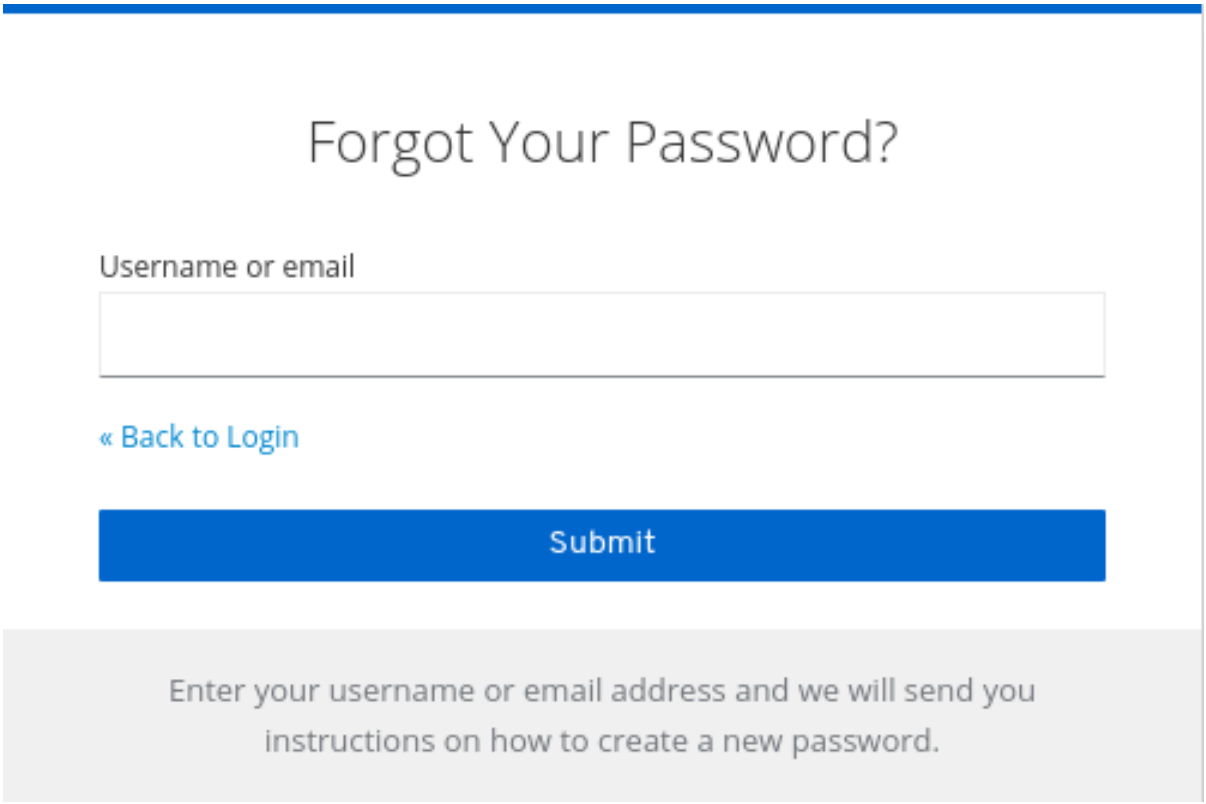
3. 将 **Forgot Password** 切换到 **ON**。
forgot password 链接会显示在您的登录页面中。

忘记密码链接



4. 单击此链接，方便用户输入用户名或电子邮件地址并接收包含链接的电子邮件，以重置其凭证。

Forgot password 页



电子邮件中发送的文本是可配置的。如需更多信息，请参阅 服务器开发人员指南。

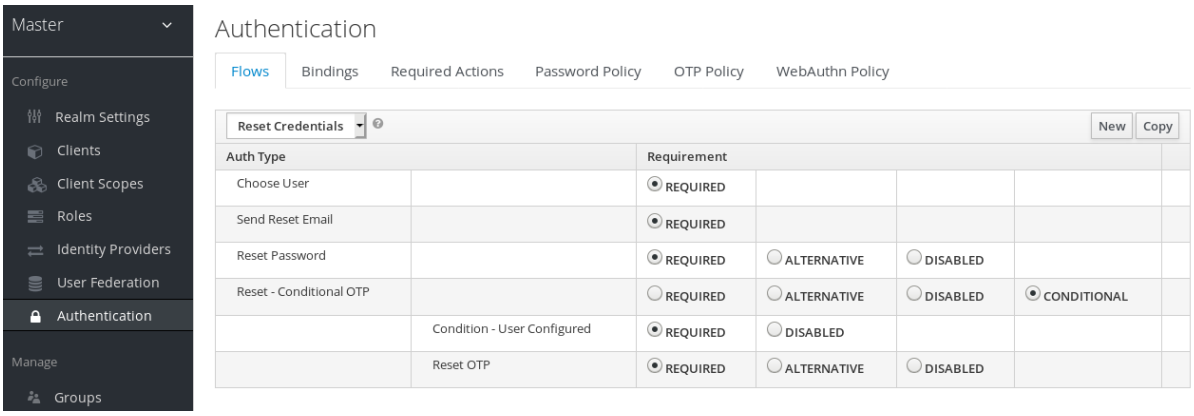
当用户点击电子邮件链接时，Red Hat Single Sign-On 会要求他们更新其密码，如果他们设置了 OTP 生成器，则 Red Hat Single Sign-On 会要求他们重新配置 OTP 生成器。根据机构的安全要求，您可能不希望用户通过电子邮件重置 OTP 生成器。

要更改此行为，请执行以下步骤：

流程

1. 在菜单中，单击 **Authentication**。
2. 点 **Flows** 选项卡。
3. 选择 **Reset Credentials** 流。

重置凭证流



如果您不想重置 OTP，请将 **Reset OTP** 要求设置为 **Disabled**。

4. 点 **所需的 Actions** 选项卡。确保启用了 *Update Password*。

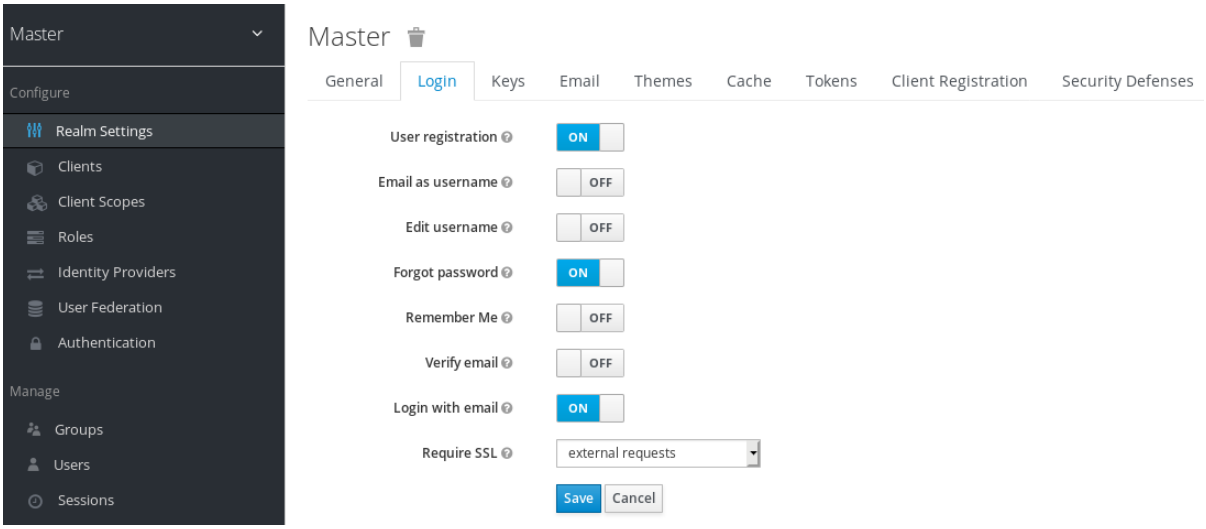
3.8.2. 启用 Remember Me

已登录的用户关闭其浏览器将销毁其会话，且该用户必须再次登录。如果在用户登录时选择了 *Remember Me*，则用户的 Red Hat Single Sign-On 登录会话保持有效状态。此操作会将仅会话 Cookie 的登录 Cookie 转变为持久 Cookie。

流程

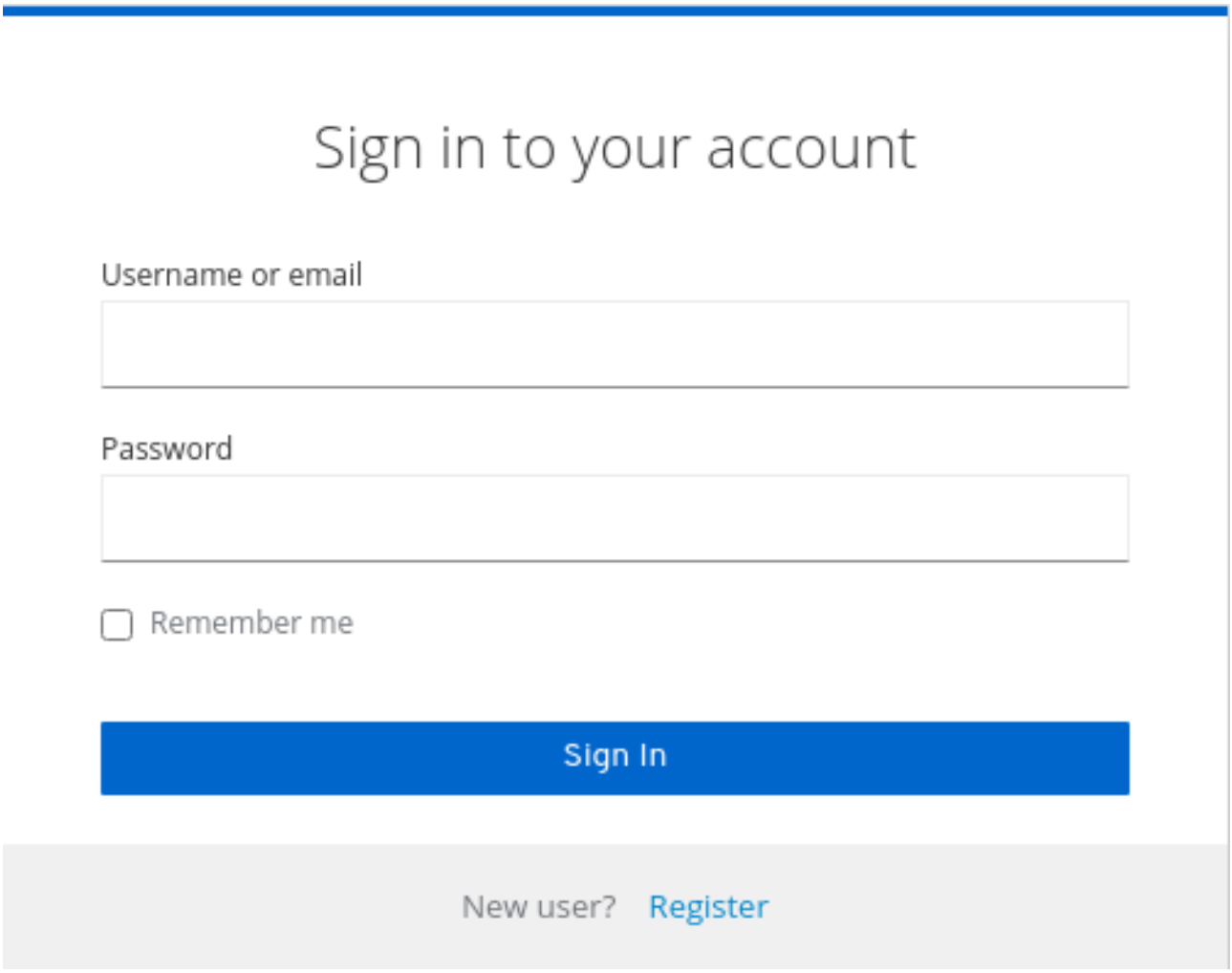
1. 点菜单中的 **Realm Settings**。
2. 点 **Login** 选项卡。
3. 将 **Remember Me** 开关切换为 **ON**。

登录标签页



在保存了此设置后，**remember me** 复选框显示在域的登录页面上。

记住我



3.8.3. ACR 到身份验证级别(LoA)映射

在域的登录设置中，您可以定义哪个 **身份验证上下文类参考(ACR)** 值映射到哪个 **级别身份验证(LoA**)。ACR 可以是任何值，而 LoA 必须是数字。在 OIDC 请求中发送的 **声明** 或 **acr_values** 参数中可以请求 Acr claim，它也包含在访问令牌和 ID 令牌中。映射的数字在身份验证流程条件中使用。

如果特定客户端需要使用不同于 realm 的值，可以在客户端级别上指定映射。但是，最佳做法是存储域映射。

详情请查看 Step-up Authentication 和 the offical OIDC specification。

3.8.3.1. 更新电子邮件工作流（更新电子邮件）

使用这个工作流，用户必须使用 UPDATE_EMAIL 操作来更改自己的电子邮件地址。

该操作与单个电子邮件输入表单关联。如果 realm 禁用了电子邮件验证，则此操作将允许在不验证的情况下更新电子邮件。如果 realm 启用了电子邮件验证，则操作会将电子邮件更新操作令牌发送到新电子邮件地址，而不更改帐户电子邮件。只有操作令牌触发才会完成电子邮件更新。

应用程序可以通过利用 UPDATE_EMAIL 作为 AIA （应用程序初始操作）将用户发送到电子邮件更新表单。

注意

UpdateEmail **是技术预览，不被完全支持。此功能默认为禁用。**

要使用 **-Dkeycloak.profile=preview** 或 **-Dkeycloak.profile.feature.update_email=enabled** 来启用服务器。如需了解更多详细信息，请参阅 配置文件。

注意

如果您启用此功能，且正在从之前的版本进行迁移，请在您的域中启用 **更新电子邮件** 所需的操作。否则，用户无法更新其电子邮件地址。

3.9. 配置域密钥

Red Hat Single Sign-On 使用的身份验证协议需要加密签名，有时也会加密。Red Hat Single Sign-On 使用非对称密钥对（私钥和公钥）来实现此目的。

Red Hat Single Sign-On 有一个活跃的密钥对，但也可以有多个被动键。主动密钥对用于创建新签名，而可以使用被动密钥对来验证之前的签名。这样可以定期轮转密钥，而不会对用户造成停机或中断。

创建域时，会自动生成密钥对和自签名证书。

流程

1. 在 Admin 控制台中选择 realm。
2. 单击 **Realm 设置**。
3. 点 **Keys**。
4. 点 **Passive** 查看被动键。
5. 点 **Disabled** 查看禁用的密钥。

密钥对可以具有 **Active** 状态，但仍然未选择为该域的当前活动密钥对。所选用于签名的活跃对根据根据优先级排序的第一个密钥供应商来选择，它们可以提供一个活跃的密钥对。

3.9.1. 轮转密钥

我们建议您定期轮转密钥。首先，创建优先级高于现有活跃密钥的新密钥。相反，您可以使用相同的优先级创建新密钥，并使之前的密钥被动。

新密钥可用后，所有新令牌和 Cookie 都用新密钥签名。当用户向应用进行身份验证时，SSO cookie 将使用新签名进行更新。刷新 OpenID Connect 令牌时，新的令牌将使用新密钥签名。最后，所有 Cookie 和令牌都使用新密钥，并在同时删除旧密钥之后。

删除旧密钥的频率在安全性之间是利弊，并确保所有 Cookie 和令牌都已更新。考虑在创建新的密钥后，每三至六个月创建新密钥，并在创建新密钥后将旧密钥删除一到两个月。如果用户在添加的新密钥和要删除的旧密钥间处于非活动状态，则用户必须重新进行身份验证。

轮转密钥也适用于离线令牌。为确保它们已更新，应用程序需要在删除旧密钥前刷新令牌。

3.9.2. 添加生成的密钥对

使用这个流程生成包含自签名证书的密钥对。

流程

1. 在 Admin 控制台中选择 realm。
2. 单击 **Realm 设置**。
3. 点 **Keys** 标签页。
4. 单击 **Providers** 选项卡。
5. 单击 **Add keystore**，再选择 **rsa-generated**。
6. 在 **优先级** 字段中输入数字。此数字决定了新密钥对是否成为活跃的密钥对。最高数字使密钥对处于活动状态。
7. 为 **keysize** 选择一个值。
8. 点 **Save**。

更改供应商的优先级不会生成密钥，但如果要更改 keysize，则可编辑提供程序和新密钥。

3.9.3. 通过提取证书来轮转密钥

您可以通过从 RSA 生成的密钥对中提取证书并在新密钥存储中使用该证书来轮转密钥。

先决条件

- 生成的密钥对

流程

1. 在 Admin 控制台中选择 realm。
2. 单击 **Realm Settings**。
3. 点 **Keys** 标签页。
   此时会出现 **Active** key 列表。
4. 在带有 RSA 密钥的行中，点 **Public Keys** 下的 **Certificate**。
   证书以文本形式出现。
5. 将证书保存到文件中，并将其放在这些行中。

```
----Begin Certificate----
<Output>
----End Certificate----
```

6. 使用 **keytool** 命令将密钥文件转换为 PEM 格式。
7. 从密钥存储中删除当前的 RSA 公钥证书。

```
keytool -delete -keystore <keystore>.jks -storepass <password> -alias <key>
```

8. 将新证书导入到密钥存储中

```
keytool -importcert -file domain.crt -keystore <keystore>.jks -storepass <password>  -alias
<key>
```

9. 取消部署并重新构建应用。

```
wildfly:undeploy
mvn clean install wildfly:deploy
```

3.9.4. 添加现有的密钥对和证书

要添加在其他位置获取的密钥对和证书， 并从下拉菜单中选择 **rsa**。您可以更改优先级，以确保新密钥对成为活跃的密钥对。

先决条件

- 私钥文件。该文件必须采用 PEM 格式。

流程

1. 在 Admin 控制台中选择 realm。
2. 单击 **Realm 设置**。
3. 点 **Keys** 标签页。
4. 单击 **Providers** 选项卡。
5. 单击 **Add keystore**，再选择 **rsa**。
6. 在 **优先级** 字段中输入数字。此数字决定了新密钥对是否成为活跃的密钥对。
7. 点 **Private RSA Key** 旁边的 **Select file** 来上传私钥文件。
8. 如果您的私钥有签名证书，点 **Select file** beside **X509 Certificate** to upload the certificate file。如果您尚未上传证书，红帽单点登录会生成自签名证书。
9. 点 **Save**。

3.9.5. 从 Java 密钥存储加载密钥

要添加存储在主机上的 Java 密钥存储文件中的密钥对和证书，请选择 **Providers**，再从下拉菜单中选择 **java-keystore**。您可以更改优先级，以确保新密钥对成为活跃的密钥对。

要让相关的证书链被导入到 Java Keystore 文件中，必须使用相同的 **Key Alias** 来加载密钥对。

流程

1. 在 Admin 控制台中选择 realm。
2. 单击 **Realm 设置**。

3. 点 **Keys** 标签页。
4. 单击 **Providers** 选项卡。
5. 单击 **Add keystore**，再选择 **java-keystore**。
6. 在 **优先级** 字段中输入数字。此数字决定了新密钥对是否成为活跃的密钥对。
7. 为 **Keystore** 输入一个值。
8. 为" **密钥存储密码"输入值**。
9. 为 **Key Alias** 输入值。
10. 为"密钥密码 **"输入值**。
11. 点 **Save**。

3.9.6. 使键被动

流程

1. 在 Admin 控制台中选择 realm。
2. 点 Realm 设置。
3. 点 **Keys** 标签页。
4. 点 **Active** 标签页。
5. 点击您要进行被动的键的供应商。
6. 将 **Active** 切换到 **OFF**。
7. 点 **Save**。

3.9.7. 禁用密钥

流程

1. 在 Admin 控制台中选择 realm。
2. 点 Realm 设置。
3. 点 **Keys** 标签页。
4. 点 **Active** 标签页。
5. 点击您要进行被动的键的供应商。
6. 将 **Enabled** 切换到 **OFF**。
7. 点 **Save**。

3.9.8. 破坏的密钥

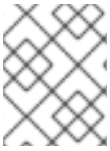
Red Hat Single Sign-On 具有仅存储在本地的签名密钥，它们永远不会与客户端应用程序、用户或其它实体共享。但是，如果您认为您的域签名密钥已被泄露，则应该首先生成新密钥对（如上所述），然后立即删除已被破坏的密钥对。

或者，您可以从 **Providers** 表中删除提供程序。

流程

1. 点菜单中的 **Clients**。
2. 单击 **security-admin-console**。
3. 单击 **Revocation** 选项卡。
4. 单击 **Set to now**。
5. 点 **Push**。

推送 not-before 策略可确保客户端应用程序不接受已被破坏的密钥签名的现有令牌。客户端应用程序被强制从 Red Hat Single Sign-On 下载新的密钥对，因此已被破坏的密钥签名的令牌将无效。



注意

REST 和机密客户端必须设置 **Admin URL**，以便 Red Hat Single Sign-On 可以在策略请求前发送推送后的客户端。

第 4 章 使用外部存储

组织可以包含信息、密码和其他凭证的数据库。通常，您无法将现有数据存储迁移到 Red Hat Single Sign-On 部署，以便 Red Hat Single Sign-On 可以联合现有的外部用户数据库。Red Hat Single Sign-On 支持 LDAP 和 Active Directory，但您也可以使用 Red Hat Single Sign-On User Storage SPI 为任何自定义用户数据库的代码扩展。

当用户尝试登录时，Red Hat Single Sign-On 会检查该用户的存储来查找该用户。如果 Red Hat Single Sign-On 没有找到用户，Red Hat Single Sign-On 会迭代该域的每个用户存储供应商，直到它找到匹配项。来自外部数据存储的数据然后映射到红帽单点登录运行时使用的标准用户模型。然后，这个用户模型映射到 OIDC 令牌声明和 SAML 断言属性。

外部用户数据库很少有必要数据来支持 Red Hat Single Sign-On 的所有功能，因此用户存储供应商可以选择在红帽单点登录用户数据存储中本地存储项目。提供商可以在本地导入用户，并定期与外部数据存储进行同步。这种方法取决于提供程序的功能以及提供程序的配置。例如，外部用户数据存储可能不支持 OTP。OTP 可以被 Red Hat Single Sign-On 处理和存储，具体取决于供应商。

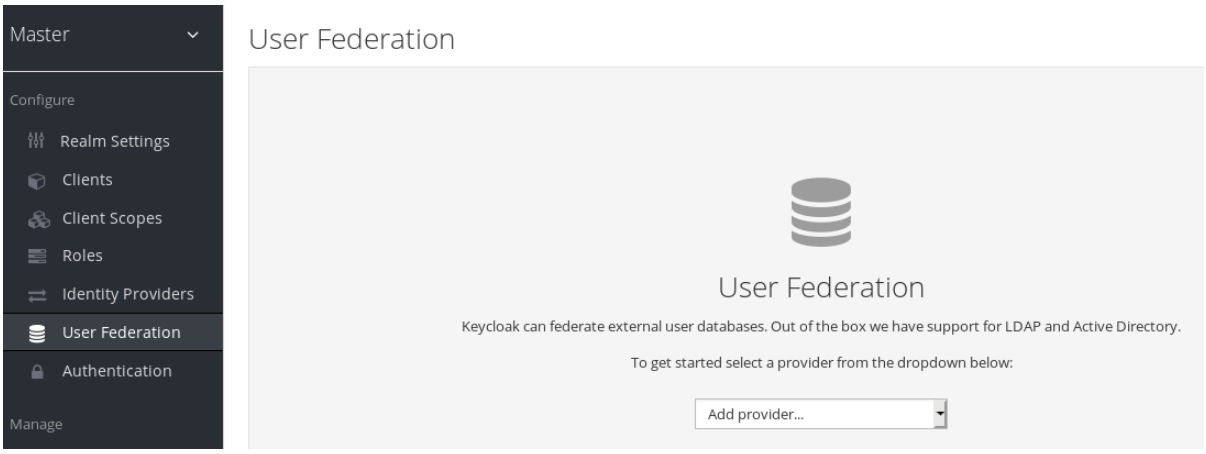
4.1. 添加供应商

要添加存储供应商，请执行以下步骤：

流程

1. 点菜单中的 **User Federation**。

用户联合



2. 从 **Add Provider** 列表选择提供程序类型。Red Hat Single Sign-On 为您提供了该提供程序的配置页面。

4.2. 处理供应商失败

如果用户存储供应商失败，您可能无法在 Admin 控制台中登录并查看用户。Red Hat Single Sign-On 不会检测使用存储提供程序查找用户时的故障，因此它会取消调用。如果您的存储提供程序具有高优先级，则用户查询在用户查找过程中失败，则登录或用户查询会失败，且不会切换到下一个配置的供应商。

Red Hat Single Sign-On 在任何 LDAP 或自定义用户供应商之前，首先搜索本地 Red Hat Single Sign-On 用户数据库来解析用户。考虑创建存储在本地 Red Hat Single Sign-On 用户数据库中管理员帐户，以防出现连接到 LDAP 和后端的问题。

每个 LDAP 和自定义 User Storage Provider 在其 Admin Console 页面上都有一个 **启用** 切换功能。禁用 User Storage Provider 会在执行查询时跳过提供程序，因此您可以使用优先级较低的不同提供程序查看和登录。如果您的供应商使用导入策略并被禁用，则导入的用户仍可以只读模式查找。

当存储提供程序查找失败时，Red Hat Single Sign-On 不会故障切换，因为用户数据库通常在它们之间有重复的用户名或重复电子邮件。重复的用户名和电子邮件可能会导致问题，因为当用户在管理员要求从另一个数据存储加载时从一个外部数据存储加载时。

4.3. 轻量级目录访问协议(LDAP)和 ACTIVE DIRECTORY

红帽单点登录包括 LDAP/AD 供应商。您可以在一个 Red Hat Single Sign-On realm 中联合多个不同的 LDAP 服务器，并将 LDAP 用户属性映射到 Red Hat Single Sign-On 通用用户模型中。

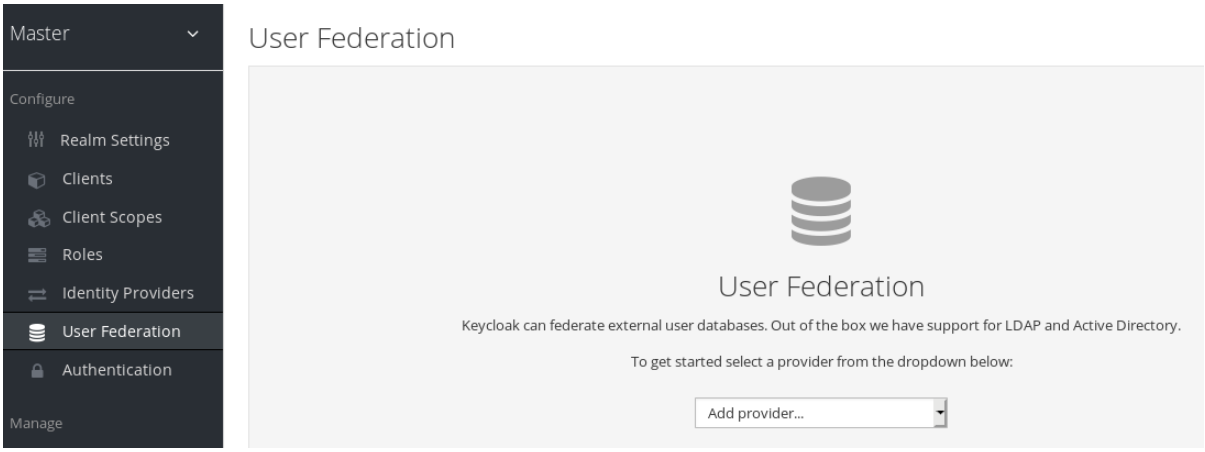
默认情况下，红帽单点登录映射用户帐户的用户名、电子邮件、名字和姓氏，但您也可以配置其他 映射。Red Hat Single Sign-On 的 LDAP/AD 供应商支持使用 LDAP/AD 协议和存储、编辑和同步模式验证密码验证。

4.3.1. 配置联合 LDAP 存储

流程

1. 点菜单中的 **User Federation**。

用户联合



2. 从 **Add Provider** 列表中选择 *ldap*。Red Hat Single Sign-On 为您提供了 LDAP 配置页面。

4.3.2. 存储模式

Red Hat Single Sign-On 将用户从 LDAP 导入到本地 Red Hat Single Sign-On 用户数据库。这个用户数据库的副本会按需同步，或者通过定期后台任务同步。同步密码存在异常。Red Hat Single Sign-On 不会导入密码。密码验证始终出现在 LDAP 服务器上。

同步的优点在于，所有红帽单点登录功能都高效工作，因为任何需要的额外用户数据都存储在本地。缺点是，每次 Red Hat Single Sign-On 首次查询特定用户时，Red Hat Sign-On 都会执行对应的数据库插入。

您可以将导入与 LDAP 服务器同步。当 LDAP 映射总是从 LDAP 而不是数据库读取特定属性时，导入同步是不必要的。

您可以将 LDAP 与 Red Hat Single Sign-On 一起使用，无需将用户导入到 Red Hat Single Sign-On 用户数据库中。LDAP 服务器备份 Red Hat Single Sign-On 运行时使用的通用用户模型。如果 LDAP 不支持 Red Hat Single Sign-On 功能需要的数据，则该功能将无法正常工作。这种方法的优势在于，您不能将

LDAP 用户副本导入和同步至 Red Hat Single Sign-On 用户数据库中的资源使用。

在 LDAP 配置页面中的 **Import Users** 切换会控制这个存储模式。要导入用户，请将此开关切换到 **ON**。

注意

如果您禁用 **Import Users**，则无法将用户配置集属性保存到 Red Hat Single Sign-On 数据库中。另外，除了映射到 LDAP 的用户配置集元数据外，您无法保存元数据。这个元数据可以根据 LDAP 映射、映射程序和其他元数据包含角色映射、组映射和其他元数据。

当您试图更改非LDAP 映射的用户数据时，无法更新用户更新。例如，除非用户的 **已启用** 标记映射到 LDAP 属性，否则您无法禁用 LDAP 映射用户。

4.3.3. 编辑模式

用户和管理员可以通过 帐户控制台和管理员修改用户元数据、用户，并通过管理控制台 修改用户元数据。LDAP 配置页面中的 **Edit Mode** 配置定义了用户的 LDAP 更新权限。

READONLY

您不能更改用户名、电子邮件、名字、姓氏和其他映射的属性。Red Hat Single Sign-On 向用户尝试更新这些字段时显示错误。不支持密码更新。

可写

您可以更改用户名、电子邮件、名字、姓氏和其他映射的属性和密码，并将它们与 LDAP 存储自动同步。

UNSYNCED

Red Hat Single Sign-On 存储了对 Red Hat Single Sign-On 本地存储的用户名、电子邮件、姓氏和密码的更改，因此管理员必须将此数据同步回 LDAP。在这个模式中，Red Hat Single Sign-On 部署可以更新只读 LDAP 服务器上的用户元数据。当将用户从 LDAP 导入到本地 Red Hat Single Sign-On 用户数据库时，此选项也适用。

注意

当 Red Hat Single Sign-On 创建 LDAP 提供程序时，红帽单点登录还会创建一组初始 LDAP 映射程序。Red Hat Single Sign-On 根据 **Vendor**, **Edit Mode**, and **Import Users** 的组合来配置这些映射程序。例如，在编辑模式为 UNSYNCED 时，Red Hat Single Sign-On 会将映射程序配置为从数据库读取特定用户属性，而不是从 LDAP 服务器中读取特定用户属性。但是，如果您稍后更改了编辑模式，则映射程序的配置不会改变，因为无法检测到 UNSYNCED 模式的配置是否已更改。在创建 LDAP 提供商时决定 **Edit Mode**此备注也适用于 **Import Users** 交换机。

4.3.4. 其他配置选项

控制台显示名称

要在 admin 控制台中显示的供应商名称。

优先级

查找用户或添加用户时的供应商的优先级。

同步注册

如果您希望由 Red Hat Single Sign-On 创建的新用户切换为 **ON**。

允许 Kerberos 身份验证

使用从 LDAP 置备的用户数据在域中启用 Kerberos/SPNEGO 身份验证。如需更多信息，请参阅 Kerberos 部分。

其他选项

将鼠标指针悬停在 Admin Console 中的工具提示上，以查看有关这些选项的更多详情。

4.3.5. 通过 SSL 连接到 LDAP

当您为 LDAP 存储配置安全连接 URL （例如**ldaps://myhost.com:636**）时，Red Hat Single Sign-On 使用 SSL 与 LDAP 服务器通信。在 Red Hat Single Sign-On 服务器端配置信任存储，以便 Red Hat Single Sign-On 可以信任到 LDAP 的 SSL 连接。

使用 Truststore SPI 为 Red Hat Single Sign-On 配置全局信任存储。有关配置全局信任存储的更多信息，请参阅 服务器安装和配置指南。如果您没有配置 Truststore SPI，则信任存储会回退到 Java 提供的默认机制，后者可以是 **javax.net.ssl.trustStore** 系统属性提供的文件，或者 JDK 中的 cacerts 文件（如果设置系统属性）。

使用 Truststore SPI 配置属性在 LDAP 联合供应商配置中，控制信任存储 SPI。默认情况下，Red Hat Single Sign-On 会将 属性设置为 **仅适用于 ldaps**，这对于大多数部署来说是足够的。如果连接到 LDAP 的连接 URL 仅以 **ldaps** 开始，Red Hat Single Sign-On 使用 Truststore SPI。

4.3.6. 将 LDAP 用户同步到红帽单点登录

如果您设置了 **Import Users** 选项，LDAP 提供程序将 LDAP 用户导入到 Red Hat Single Sign-On local 数据库中。当用户第一次登录时，LDAP 供应商将 LDAP 用户导入到 Red Hat Single Sign-On 数据库并验证 LDAP 密码。当用户第一次登录时，用户才是 Red Hat Single Sign-On 导入用户的唯一时间。如果单击 Admin Console 中的 **Users** 菜单并点击 **View all users** 按钮，则您只看到红帽单点登录至少验证的 LDAP 用户。Red Hat Single Sign-On 以这种方式导入用户，因此此操作不会触发整个 LDAP 用户数据库的导入。

如果要将所有 LDAP 用户同步到 Red Hat Single Sign-On 数据库，在 LDAP 提供程序配置页面中配置和启用 **Sync Settings**。

存在两种类型的同步：

定期完全同步

此类型将所有 LDAP 用户同步到 Red Hat Single Sign-On 数据库中。已存在于 Red Hat Single Sign-On 的 LDAP 用户，但在 LDAP 中有所不同，直接更新 Red Hat Sign-On 数据库中。

定期更改的用户同步

同步时，Red Hat Single Sign-On 会在最近一次同步后创建或更新用户。

您第一次创建 LDAP 供应商时，最好单击 **Synchronize all 用户**，然后设置更改用户的定期同步。

4.3.7. LDAP 映射器

LDAP 映射器是由 LDAP 提供程序触发 **的监听程序**。它们为 LDAP 集成提供了另一个扩展点。当以下情况时触发 LDAP 映射：

- 用户使用 LDAP 登录。
- 用户最初注册。
- Admin Console 查询用户。

当您创建 LDAP Federation 提供者时，Red Hat Single Sign-On 会自动为此提供程序提供一组 **映射程序**。用户可以更改此设置，也可以开发映射程序或更新/删除现有的项。

用户属性映射程序

此映射程序指定了哪些 LDAP 属性映射到 Red Hat Single Sign-On 用户的 属性。例如，您可以将 **mail** LDAP 属性配置为 Red Hat Single Sign-On 数据库中 **的电子邮件** 属性。对于这一映射程序实施，始终存在一对一的映射。

FullName Mapper

此映射程序指定用户的全名。Red Hat Single Sign-On 将名称保存在 LDAP 属性中（通常为 **cn**），并将名称映射到 Red Hat Single Sign-On 数据库中的 **firstName** 和 **lastname** 属性。在 LDAP 部署中，使用 **cn** 使其包含用户的全名是通用的。

注意

当您在 Red Hat Single Sign-On 和 **Sync Registrations** 中注册新用户时，对于 LDAP 供应商而言，fullName mapper 将允许回退到用户名。在使用 Microsoft Active Directory (MSAD)时，这个回退很有用。MSAD 的常见设置是将 **cn** LDAP 属性配置为 fullName，同时将 **cn** LDAP 属性用作 LDAP 提供程序配置中 **RDN LDAP** 属性。通过这个设置，Red Hat Single Sign-On 会回退到用户名。例如，如果您创建 Red Hat Single Sign-On 用户 "john123" 并保留 firstName 和 lastName 空，则 fullname mapper 会将 "john123" 保存为 LDAP 中的 **cn** 的值。当您为 firstName 和 lastName 输入 "John Doe" 时，全name mapper 会更新 LDAP **cn** 到 "John Doe" 值，因为回退到用户名是不必要的。

硬编码属性映射

此映射程序为与 LDAP 链接的每个 Red Hat Single Sign-On 用户添加了一个硬编码的属性值。此映射程序还可以强制 **启用** 或 **电子邮件用户** 属性的值。

角色映射程序

这个映射程序配置从 LDAP 到 Red Hat Single Sign-On 角色映射的角色映射。单个角色映射映射 LDAP 角色（通常是来自 LDAP 树的特定分支的组）到与指定客户端的域角色或客户端角色对应的角色。您可以为同一 LDAP 供应商配置更多角色映射程序。例如，您可以指定 **ou=main,dc=example,dc=org** map 下的组到 realm 角色映射，以及来自 **ou=finance,dc=example,dc=org** 下的组的 **角色映射**。

硬编码的角色映射程序

此映射程序为来自 LDAP 提供商的每个 Red Hat Single Sign-On 用户授予指定的红帽单点登录角色。

组群映射程序

此映射程序将 LDAP 组从 LDAP 树分支映射到 Red Hat Single Sign-On 中的组。此映射还会将用户组映射从 LDAP 传播到 Red Hat Single Sign-On 中的 user-group 映射。

MSAD 用户帐户映射程序

此映射程序专用于 Microsoft Active Directory (MSAD)。它可以将 MSAD 用户帐户状态集成到 Red Hat Single Sign-On 帐户状态，如启用帐户或过期密码。这个映射程序使用 **userAccountControl**、**pwdLastSet** LDAP 属性（特定于 MSAD），且不是 LDAP 标准。例如，如果 **pwdLastSet** 的值为 **0，** Red Hat Single Sign-On 用户必须更新其密码。其结果是添加至用户的 UPDATE_PASSWORD 必需操作。如果 **userAccountControl** 的值为 **514** （禁用帐户），则代表红帽单点登录用户被禁用。

证书映射程序

这个映射映射 X.509 证书。Red Hat Single Sign-On 与其与 X.509 身份验证和完整 **证书结合使用，采用 PEM 格式** 作为身份源。此映射的行为与用户属性 **映射程序** 类似，但红帽单点登录可以过滤存储 PEM 或 DER 格式的 LDAP 属性。启用 **Always Read Value from LDAP** with this mapper.

将基本红帽单点登录用户属性（如用户名、名字、姓氏和电子邮件）映射到对应的 LDAP 属性的用户属性。您可以扩展这些属性并提供自己的额外属性映射。Admin Console 提供工具提示，可帮助配置对应的映射程序。

4.3.8. 密码散列

当 Red Hat Single Sign-On 更新密码时，Red Hat Single Sign-On 以纯文本格式发送密码。此操作与更新内置 Red Hat Single Sign-On 数据库中的密码不同，其中 Red Hat Single Sign-On 哈希和 salt 在将密码发送到数据库之前。对于 LDAP，Red Hat Single Sign-On 依赖于 LDAP 服务器来哈希和 salt 密码。

默认情况下，LDAP 服务器（如 MSAD、RHDS 或 FreeIPA 哈希）和 salt 密码。其他 LDAP 服务器（如 OpenLDAP 或 ApacheDS）会以纯文本形式存储密码，除非您使用 *LDAPv3 Password Modify Extended Operation*，如 RFC3062 所述。在 LDAP 配置页面中启用 LDAPv3 Password Modify Extended Operation。如需了解更多详细信息，请参阅您的 LDAP 服务器文档。

警告

始终通过在使用 **ldapsearch** 和 base64 解码 **userPassword** 属性值来检查更改的目录条目，来验证用户密码是否正确进行哈希存储，而不是以明文形式存储。

4.3.9. 故障排除

将类别 **org.keycloak.storage.ldap** 的日志级别增加到 TRACE。通过这个设置，许多日志消息发送到 **TRACE** 级别中的服务器日志，包括所有对 LDAP 服务器和参数（用于发送查询）的日志记录。当您在用户论坛或 JIRA 上创建任何 LDAP 问题时，请考虑将服务器日志与已启用 TRACE 日志记录连接。如果情况太大，则好的替代方案是将服务器日志中的代码片段包含在操作期间添加到日志中，这会导致问题所在。

- 创建 LDAP 供应商时，服务器日志中会出现从以下内容开始的信息：

> When you create LDAP provider, message appear in the server log in the INFO level starting with:

> Creating new LDAP Store for the LDAP storage provider: ...

它显示了您的 LDAP 供应商的配置。在您提出问题或报告错误前，最好包含此消息来显示您的 LDAP 配置。最后，可以随意替换一些您不需要的配置更改，并附带一些占位符值。一个例子是 **bindDn=some-placeholder**。对于 **connectionUrl**，请随时替换它，但至少包含协议(**ldap** 和 **ldaps**)是非常有用的。同样，包含配置 LDAP 映射的详情（这些映射会在 DEBUG 级别显示）时很有用：

> Mapper for provider: XXX, Mapper name: YYY, Provider: ZZZ ...

请注意，这些消息只会显示启用的 DEBUG 日志记录。

- 要跟踪性能或连接池问题，请考虑设置属性 **Pool Debug Level** 的值。

为跟踪性能或连接池问题，请考虑将 LDAP 提供商 **的属性 Pool Debug Level** 的值设置为 **all**。这将添加很多额外的信息，以便记录 LDAP 连接池包含的日志。这可用于跟踪与连接池或性能相关的问题。

注意

更改连接池的配置后，您可能需要重启 Keycloak 服务器，以强制重新初始化 LDAP 供应商连接。

如果即使服务器重启后没有更多消息用于连接池，这可能表示连接池无法用于您的 LDAP 服务器。

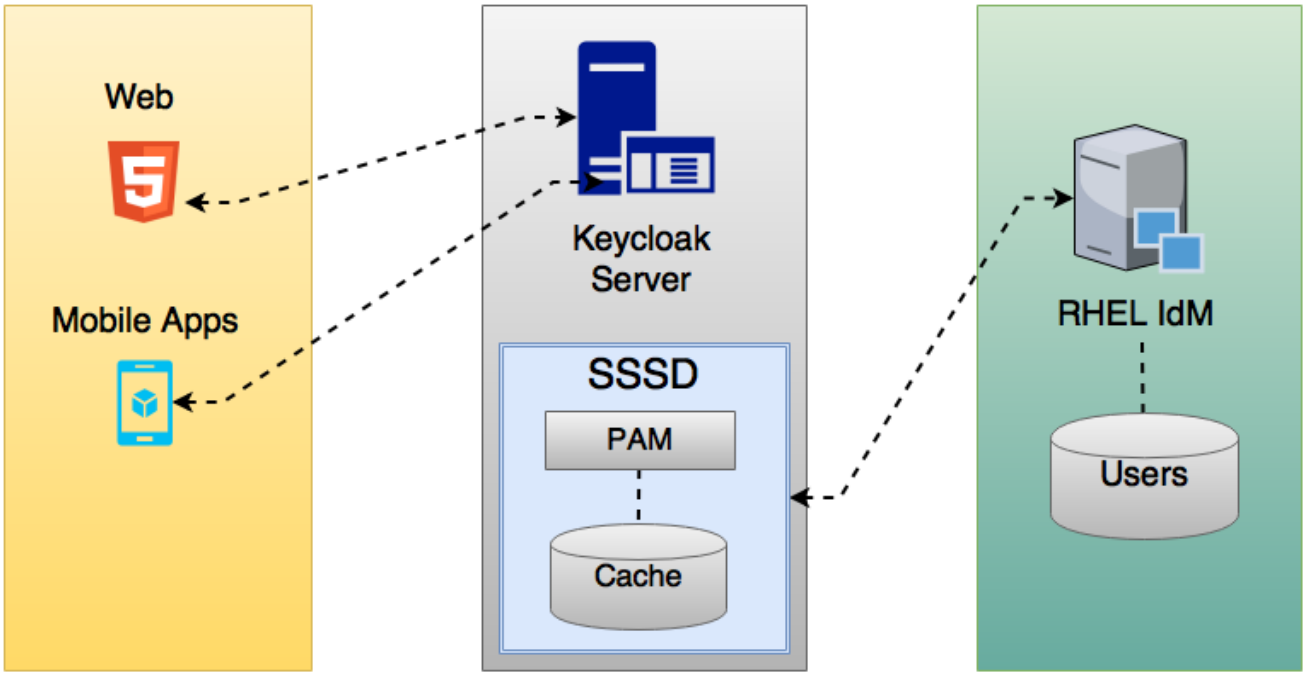
- 如需报告 LDAP 问题，您可能会考虑为目标数据附加部分 LDAP 树，这会导致您的环境出现问题。例如，如果某些用户登录时需要很多时间，您可以考虑附加显示各种"group"条目 **的成员** 属性计数的 LDAP 条目。在这种情况下，如果这些组条目映射到 Red Hat Single Sign-On 等某些组 LDAP 映射（或角色 LDAP 映射程序）时，添加可能很有用。

如需报告 LDAP 问题，您可能会考虑为目标数据附加部分 LDAP 树，这会导致您的环境出现问题。例如，如果某些用户登录时需要很多时间，您可以考虑附加显示各种"group"条目 **的成员** 属性计数的 LDAP 条目。在这种情况下，如果这些组条目映射到 Red Hat Single Sign-On 中的一些组 LDAP 映射（或角色 LDAP 映射程序），那么添加可能很有用。

4.4. SSSD 和 FREEIPA 身份管理集成

Red Hat Single Sign-On 包含 系统安全服务守护进程(SSSD) 插件。SSSD 是 Fedora 和 Red Hat Enterprise Linux (RHEL)的一部分，它提供了对多个身份和验证供应商的访问。SSSD 还提供诸多优点，如故障转移和离线支持。如需更多信息，请参阅 Red Hat Enterprise Linux Identity Management 文档 。

SSSD 与 FreeIPA 身份管理(IdM)服务器集成，提供身份验证和访问控制。通过这种集成，Red Hat Single Sign-On 可以根据特权访问管理(PAM)服务进行身份验证，并从 SSSD 检索用户数据。有关在 Linux 环境中使用红帽身份管理的更多信息，请参阅 Red Hat Enterprise Linux Identity Management 文档 。



Red Hat Single Sign-On 和 SSSD 通过只读 D-Bus 接口进行通信。因此，置备和更新用户的方式是使用 FreeIPA/IdM 管理界面。默认情况下，接口导入用户名、电子邮件、名字和姓氏。

注意

Red Hat Single Sign-On 会自动注册组和角色，但不会同步它们。红帽单点登录管理员对 Red Hat Single Sign-On 管理员所做的任何更改都不会与 SSSD 同步。

4.4.1. FreeIPA/IdM 服务器

FreeIPA Docker 镜像位于 Docker Hub 中。要设置 FreeIPA 服务器，请查看 FreeIPA 文档。

流程

1. 使用以下命令运行 FreeIPA 服务器：

```
docker run --name freeipa-server-container -it \
-h server.freeipa.local -e PASSWORD=YOUR_PASSWORD \
-v /sys/fs/cgroup:/sys/fs/cgroup:ro \
-v /var/lib/ipa-data:/data:Z freeipa/freeipa-server
```

带有 **server.freeipa.local** 的参数 **-h** 代表 FreeIPA/IdM 服务器主机名。将 your **R_PASSWORD** 更改为您自己的密码。

2. 容器启动后，将 **/etc/hosts** 文件更改为包括：

```
x.x.x.x     server.freeipa.local
```

如果没有进行此更改，您必须设置 DNS 服务器。

3. 使用以下命令在 IPA 域中注册 Linux 服务器，以便 SSSD 联合供应商在红帽单点登录上启动并运行：

```
ipa-client-install --mkhomedir -p admin -w password
```

4. 在客户端上运行以下命令验证安装是否正常工作：

```
kinit admin
```

5. 输入您的密码。

6. 使用以下命令将用户添加到 IPA 服务器：

```
$ ipa user-add <username> --first=<first name> --last=<surname> --email=<email address>
--phone=<telephoneNumber> --street=<street> \      --city=<city> --state=<state> --
postalcode=<postal code> --password
```

7. 使用 kinit 强制设置用户的密码。

```
kinit <username>
```

8. 输入以下内容恢复常规 IPA 操作：

```
kdestroy -A
kinit admin
```

4.4.2. SSSD 和 D-Bus

联合供应商使用 D-BUS 从 SSSD 获取数据。它使用 PAM 验证数据。

流程

1. 安装 sssd-dbus RPM。

```
$ sudo yum install sssd-dbus
```

2. 运行以下置备脚本：

```
$ bin/federation-sssd-setup.sh
```

此脚本对 **/etc/sssd/sssd.conf** 进行以下更改：

```
[domain/your-hostname.local]
...
ldap_user_extra_attrs = mail:mail, sn:sn, givenname:givenname,
telephoneNumber:telephoneNumber
...
[sssd]
services = nss, sudo, pam, ssh, ifp
...
[ifp]
allowed_uids = root, yourOSUsername
user_attributes = +mail, +telephoneNumber, +givenname, +sn
```

3. 运行 **dbus-send** 以确保设置成功。

```
sudo dbus-send --print-reply --system --dest=org.freedesktop.sssd.infopipe
/org/freedesktop/sssd/infopipe org.freedesktop.sssd.infopipe.GetUserGroups string:john
```

如果设置成功，您会看到用户的组。如果此命令返回超时或错误，则 Red Hat Single Sign-On 上运行的联合供应商无法检索任何数据。通常会出现这个错误，因为服务器没有在 FreeIPA IdM 服务器中注册，或者没有访问 SSSD 服务的权限。

如果您没有访问 SSSD 服务的权限，请确保运行 Red Hat Single Sign-On 服务器的用户位于以下部分的 **/etc/sssd/sssd.conf** 文件中：

```
[ifp]
allowed_uids = root, your_username
```

4.4.3. 启用 SSSD 联合供应商

红帽单点登录使用 DBus-Java 与 D-Bus 的低级别通信。D-Bus 依赖于 Unix 套接字库。

在启用 SSSD Federation 供应商前，为这个库安装 RPM：

```
$ sudo yum install rh-sso7-libunix-dbus-java
```

红帽单点登录使用 JNA 通过 PAM 进行身份验证。确保已安装 JAN 软件包。

```
$ sudo yum install jna
```

使用 **sssctl user-checks** 命令验证您的设置：

```
$ sudo sssctl user-checks admin -s keycloak
```

4.5. 配置联合 SSSD 存储

安装后，配置联合 SSSD 存储。

流程

1. 点菜单中的 **User Federation**。
2. 从 **Add Provider** 列表中，选择 *sssd*。Red Hat Single Sign-On 为您提供了 sssd 配置页面。
3. 点 **Save**。

现在，您可以使用 FreeIPA/IdM 凭证与 Red Hat Single Sign-On 进行身份验证。

4.6. 自定义供应商

Red Hat Single Sign-On 具有用于用户存储 Federation 的服务提供商接口(SPI)来开发自定义提供程序。您可以在 服务器开发人员指南 中找到有关开发客户供应商的文档。

第 5 章 管理用户

从管理控制台中，您可以执行的各种操作来管理用户。

5.1. 创建用户

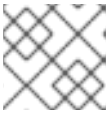
您可以在 realm 中创建这些用户所需的应用程序。避免在 master 域中创建用户，这只用于创建其他域。

前提条件

- 您位于主域以外的域中。

流程

1. 点菜单中的 **Users**。
2. 单击 **添加用户**。
3. 输入新用户的详情。



注意

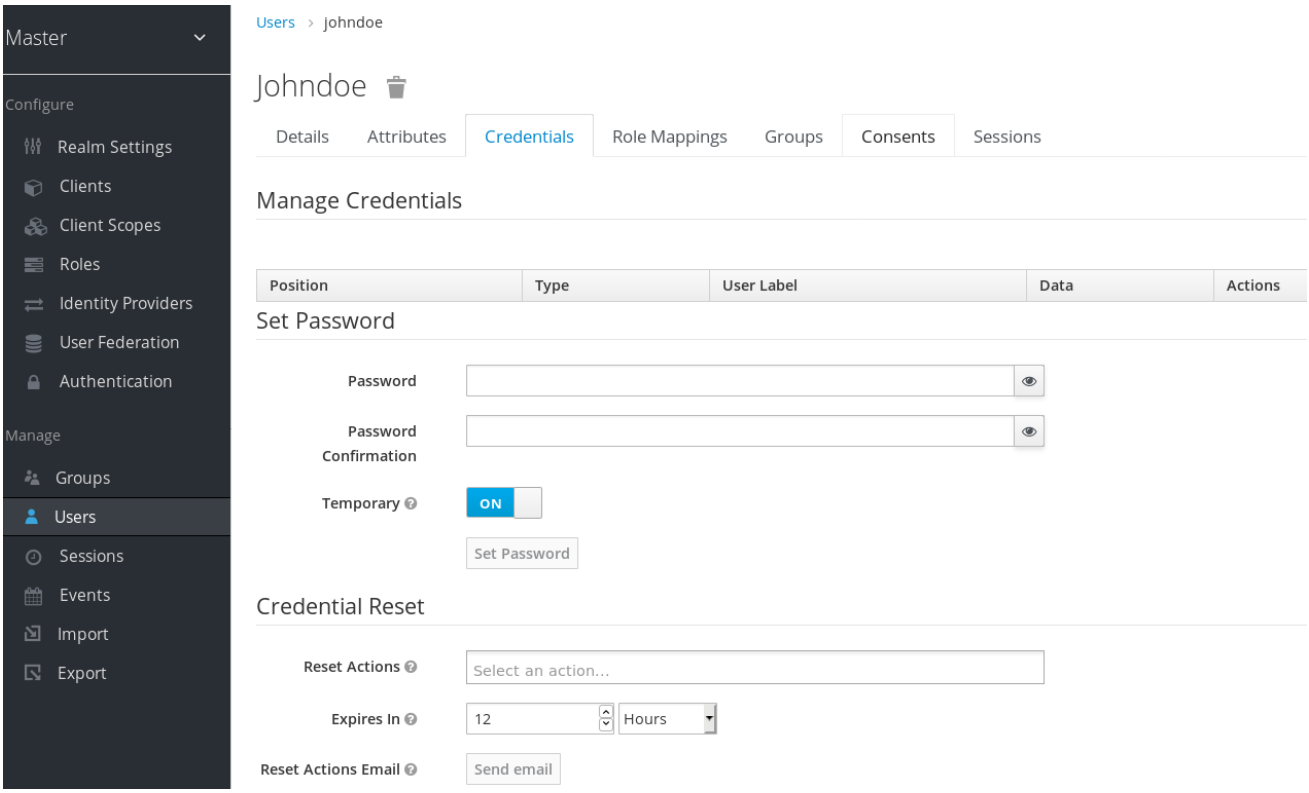
Username 是唯一的必填字段。

4. 点 **Save**。保存详情后，会显示新用户的 **Management** 页面。

5.2. 定义用户凭证

您可以在 **Credentials** 选项卡中管理用户的凭证。

凭证管理



此标签页包括以下字段：

位置

通过 Position **列中的** 箭头按钮，您可以转换用户的凭据的优先级。最顶层的凭证具有最高优先级。优先级决定了用户登录后首先显示哪些凭证。

类型

此列显示凭证类型，如**密码**或 **OTP**。

用户标签

这是在登录时作为选择选项来识别凭证的可分配标签。它可以设为任何值来描述凭据。

data

这是有关凭证的非机密技术信息。默认情况下是隐藏的。您可以点击 **Show data...** 以显示凭证的数据。

Actions

此列有两个操作：点 **Save** 记录该值或 user 字段。点 **Delete** 删除凭证。

您不能在 admin 控制台中为特定用户配置其他类型的凭证；该任务是用户的职责。

当用户丢失 OTP 设备或者凭证已被被破坏时，您可以删除用户的凭证。您只能在 **Credentials** 选项卡中删除用户的凭证。

5.2.1. 为用户设置密码

如果用户没有密码，或者密码已被删除，则会显示 **Set Password** 部分。

如果用户已拥有密码，可以在 **重置密码部分重置** 该密码。

流程

1. 点菜单中的 **Users**。此时会显示 **Users** 页面。
2. 选择用户。
3. 点 **Credentials** 选项卡。
4. 在 **Set Password** 部分中键入新密码。
5. 点 **Set Password**。

注意

如果 **Temporary** 是 **ON**，用户在首次登录时必须更改密码。要允许用户保存提供的密码，可将 **Temporary** 设置为 **OFF**。用户必须单击 **Set Password** 以更改密码。

6. 或者，您可以发送电子邮件到请求用户重置密码的用户。
   a. 导航到 **Credential Reset** 下的 **Reset Actions** 列表。
   b. 从列表中选择 **Update Password**。
   c. 单击 **Send Email**。发送的电子邮件包含一个将用户定向到 **Update Password** 窗口的链接。

d. 另外，您可以设置电子邮件链接的有效性。在 **Realm Settings** 中的 **Tokens** 选项卡中，这设置为默认的预设置。

5.2.2. 创建 OTP

如果您的域中有条件，用户必须进入 Red Hat Single Sign-On Account Console 来重新配置新的 OTP 生成器。如果需要 OTP，则用户在登录时必须重新配置新的 OTP 生成器。

另外，您可以发送电子邮件到请求用户重置 OTP 生成器的用户。如果用户已有 OTP 凭证，则以下步骤适用。

前提条件

- 登录到适当的域。

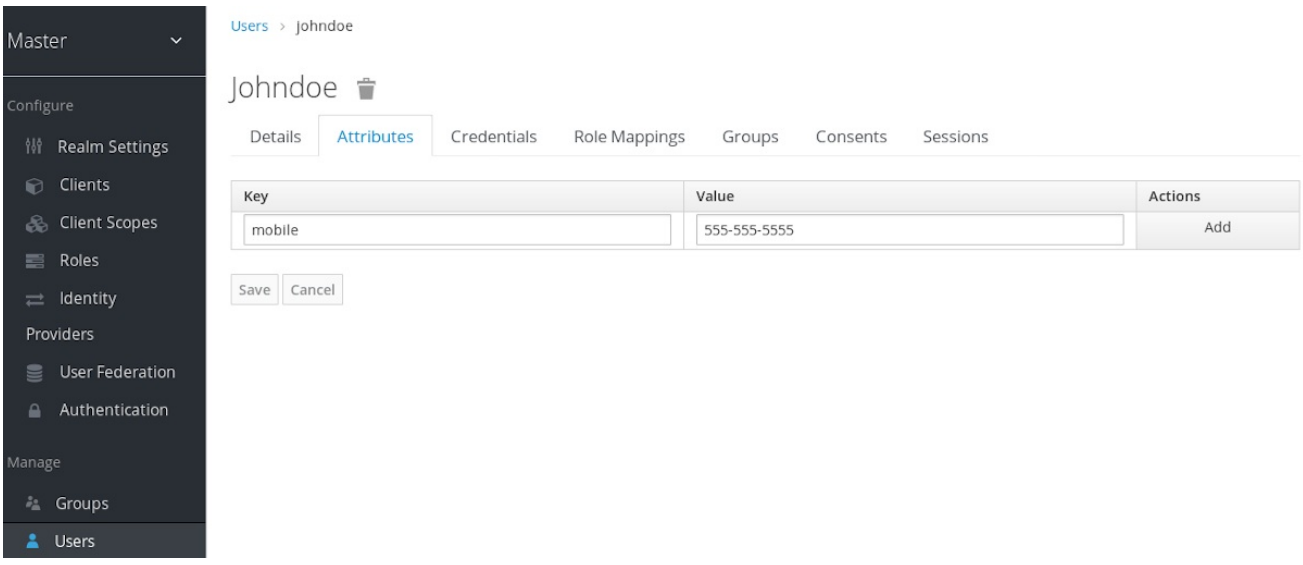
流程

1. 点主菜单中的 **Users**。此时会显示 **Users** 页面。
2. 选择用户。
3. 点 **Credentials** 选项卡。
4. 导航到 **Reset Actions** 列表。
5. 点 **Configure OTP**。
6. 单击 **Send Email**。发送的电子邮件包含一个链接，将用户定向到 **OTP 设置页面**。

5.3. 配置用户属性

用户属性为每个用户提供自定义体验。您可以通过配置用户属性来为控制台中的每个用户创建个性化身份。

用户



前提条件

- 您位于用户所在的域中。

流程

1. 点菜单中的 **Users**。
2. 选择要管理的用户。
3. 单击 **Attributes** 选项卡。
4. 在 **Key** 字段中输入属性名称。
5. 在 **Value** 字段中输入属性值。
6. 点 **Add**。
7. 点 **Save**。

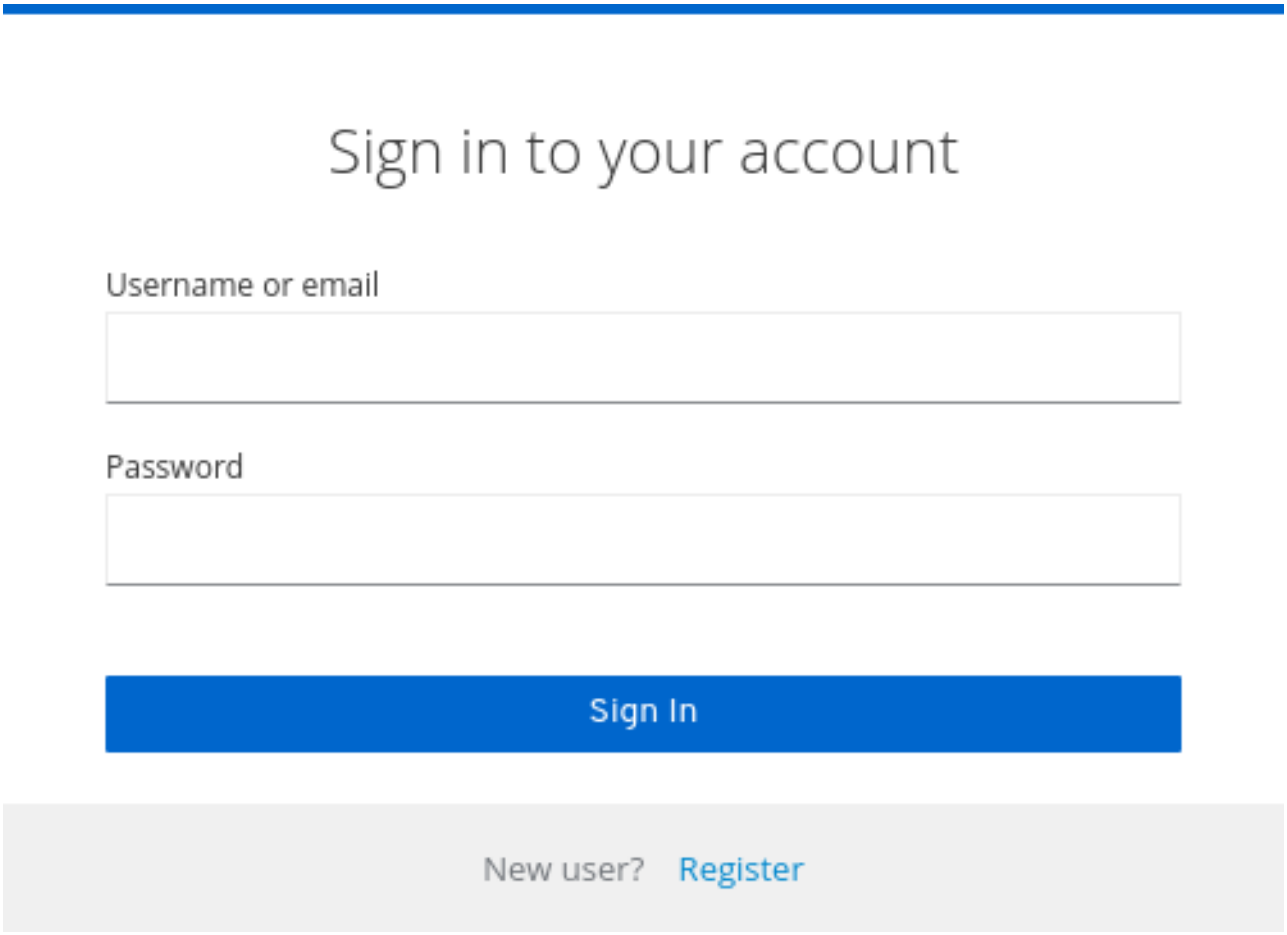
注意

管理员不应更新一些只读属性。其中包括像 **LDAP_ID** 一样以只读方式为只读的属性，这些属性由 LDAP 供应商自动填写。由于安全原因，对于典型的用户管理员，其他某些属性应为只读属性。请参阅 缓解安全威胁 一章中的详情。

5.4. 允许用户自助注册

您可以使用 Red Hat Single Sign-On 作为第三方授权服务器来管理应用程序用户，包括自助注册的用户。如果启用自我注册，则登录页面会显示一个注册链接，以便该用户能够创建帐户。

注册链接



用户必须在注册表单中添加配置集信息才能完成注册。可以通过删除或添加用户必须完成的字段来自定义注册表单。

其他资源

- 有关自定义用户注册的更多信息，请参阅 服务器开发人员指南。

5.4.1. 启用用户注册

允许用户自助注册。

流程

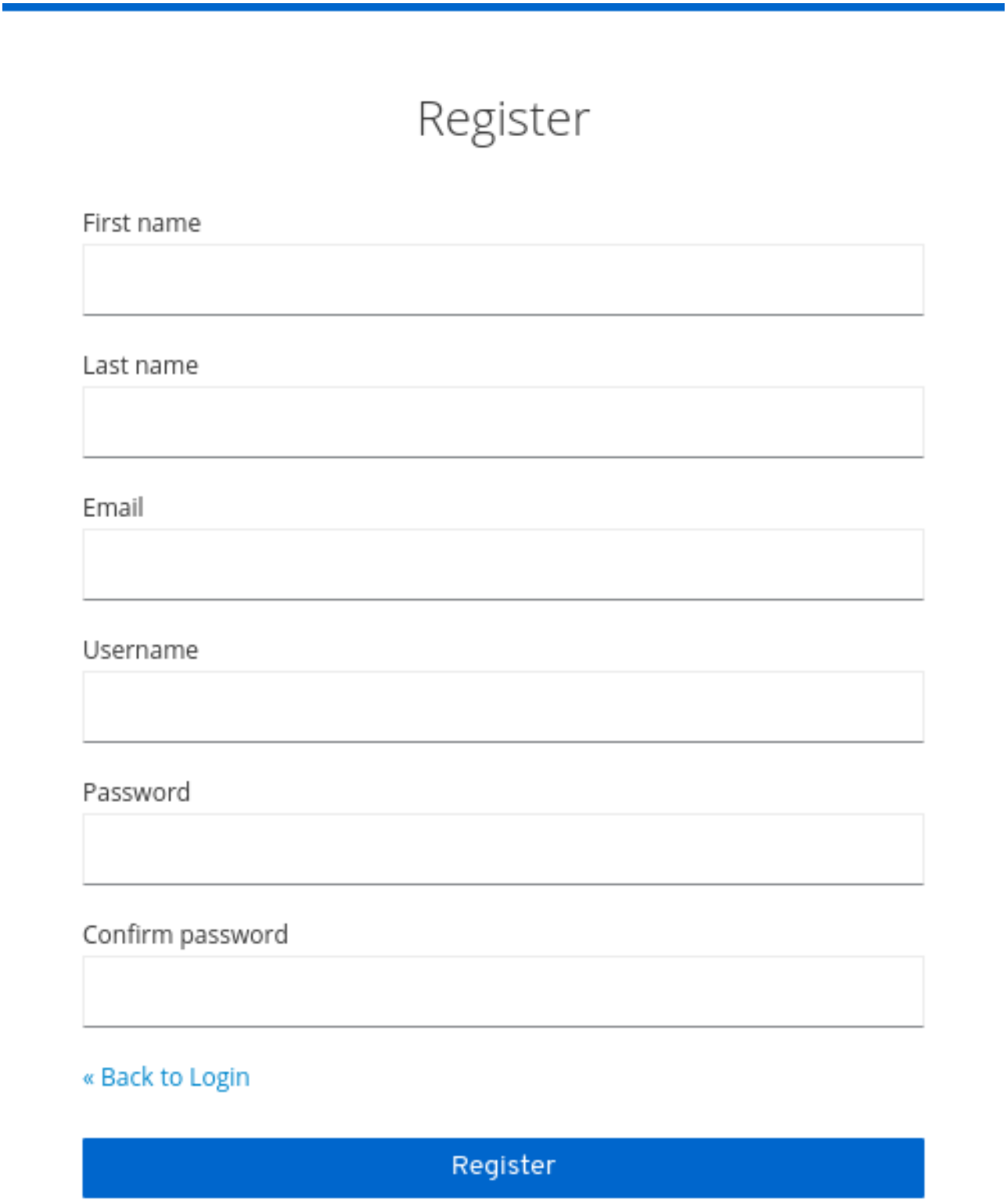
1. 点主菜单中的 **Realm Settings**。
2. 点 **Login** 选项卡。
3. 将用户 **注册** 切换为 **ON**。
4. 点 **Save**。

启用这个设置后，Admin Console 的登录页面上会显示 **Register** 链接。

5.4.2. 作为新用户注册

作为新用户，您必须完成注册表单，以便第一次登录。您可以添加配置集信息和要注册的密码。

注册表单



前提条件

- 启用用户注册。

流程

1. 点登录页面上的 **Register** 链接。此时会显示注册页面。
2. 输入用户配置集信息。

3. 输入新密码。

4. 点 **Save**。

5.5. 定义登录时所需的操作

您可以设置用户在第一次登录时必须执行的操作。用户提供凭证后需要这些操作。第一次登录时，不再需要这些操作。您可以在该用户的 **Details** 标签页中添加所需的操作。

以下是所需操作类型的示例：

更新密码

用户必须更改其密码。

配置 OTP

用户必须使用 Free OTP 或 Google Authenticator 应用程序在移动设备中配置一次性密码生成器。

验证电子邮件

用户必须验证他们的电子邮件帐户。将向用户发送一封电子邮件，其中包含必须点击的验证链接。成功完成此工作流后，用户就可以登录。

更新配置集

用户必须更新配置文件信息，如名称、地址、电子邮件和电话号码。

5.5.1. 为一个用户设置所需操作

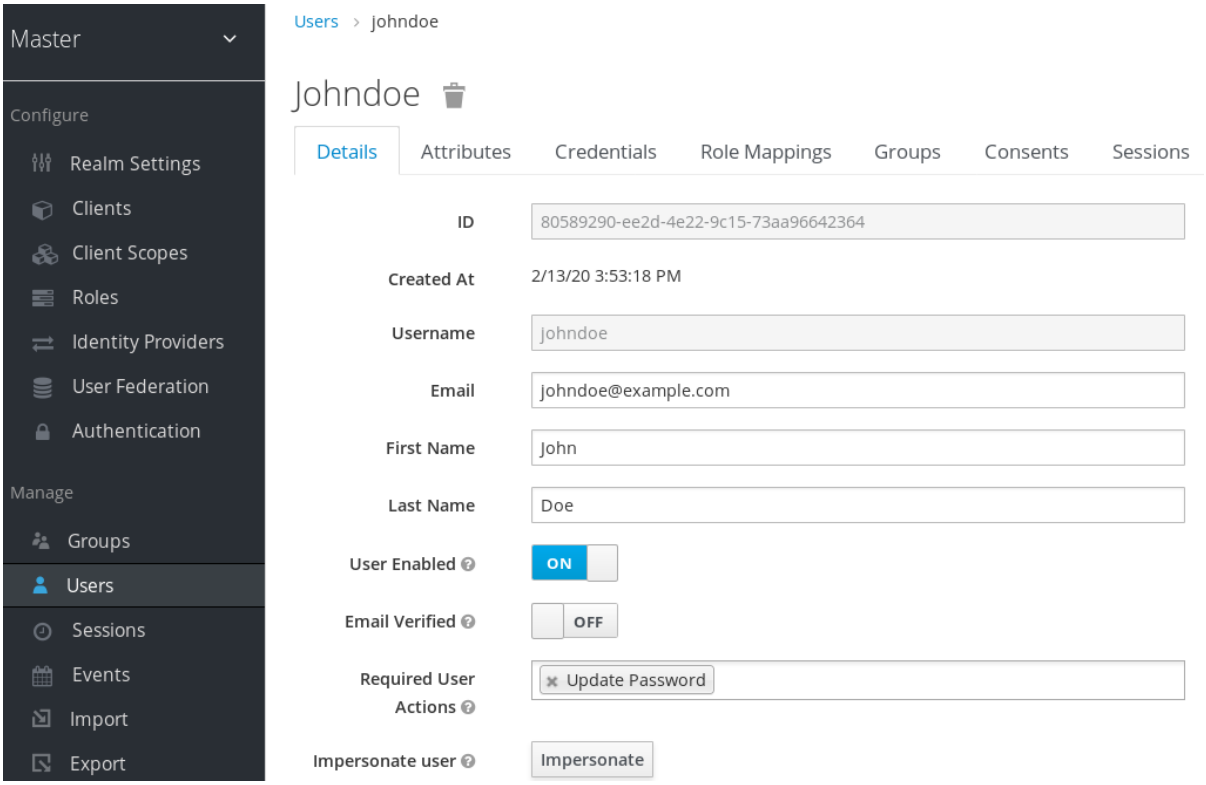
您可以设置任何用户所需的操作。

流程

1. 点菜单中的 **Users**。

2. 从列表中选择一个用户。

3. 导航到 **Required User Actions** 列表。



4. 选择您要添加到帐户中的所有操作。
5. 单击操作名称旁边的 **X** 以将它删除。
6. 选择要添加的操作后点 **Save**。

5.5.2. 为所有用户设置所需操作

您可以在所有新用户首次登录前指定所需的操作。要求应用到 **Users** 页面中由 **Add User** 按钮或登录页面上的 **Register** 链接创建的用户。

流程

1. 在菜单中，单击 **Authentication**。
2. 点 **所需的 Actions** 选项卡。
3. 在 **Default Action** 列中，单击一个或多个所需操作的复选框。当新用户第一次登录时，必须执行选定的操作。

5.5.3. 作为所需操作启用条款和条件

您可以启用新用户第一次登录 Red Hat Single Sign-On 前，必须接受条款和条件。

流程

1. 在菜单中，单击 **Authentication**。
2. 点 **所需的 Actions** 选项卡。
3. 启用 **条款和条件** 操作。
4. 编辑基本登录主题中的 **terms.ftl** 文件。

其他资源

- 有关扩展和创建它们的更多信息，请参阅 服务器开发人员指南。

5.6. 搜索用户

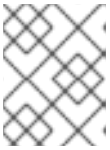
搜索用户以查看有关用户的详细信息，如用户的组和角色。

前提条件

- 您位于用户所在的域中。

流程

1. 点主菜单中的 **Users**。此时会显示这个 **Users** 页面。
2. 在搜索框中输入您要搜索的用户的完整名称、姓、名字或电子邮件地址。搜索返回与您的条件匹配的所有用户。
3. 或者，您可以点击 **View all users** 列出系统中的每个用户。



注意

此操作仅搜索本地 Red Hat Single Sign-On 数据库，而不是联合数据库，如 LDAP。federated 数据库的后端没有启用搜索用户的分页机制。

a. 要从联合后端搜索用户，必须将用户列表同步到 Red Hat Single Sign-On 数据库。调整搜索条件，将后端用户同步到 Red Hat Single Sign-On 数据库。

b. 或者，单击左侧菜单中的 **User Federation**。

i. 要将更改应用到所选用户，请点击页面中的 **Sync changed users** with your federation provider。

ii. 要将更改应用到数据库中的所有用户，请点击您的联合供应商页面上 **同步所有用户**。

其他资源

- 有关用户联邦的更多信息，请参阅用户联邦。

5.7. 删除用户

您可以删除不再需要访问应用程序的用户。如果删除了用户，用户配置文件和数据也会被删除。

流程

1. 点菜单中的 **Users**。此时会显示 **Users** 页面。
2. 单击 **View all users** 以查找要删除的用户。

注意

或者，您可以使用搜索栏来查找用户。

3. 点您要删除的用户旁边的 **Delete** 进行确认。

5.8. 启用用户删除帐户

如果您在 Admin 控制台中启用了此功能，最终用户和应用可以在帐户控制台中删除其帐户。启用此功能后，您可以为特定用户提供该功能。

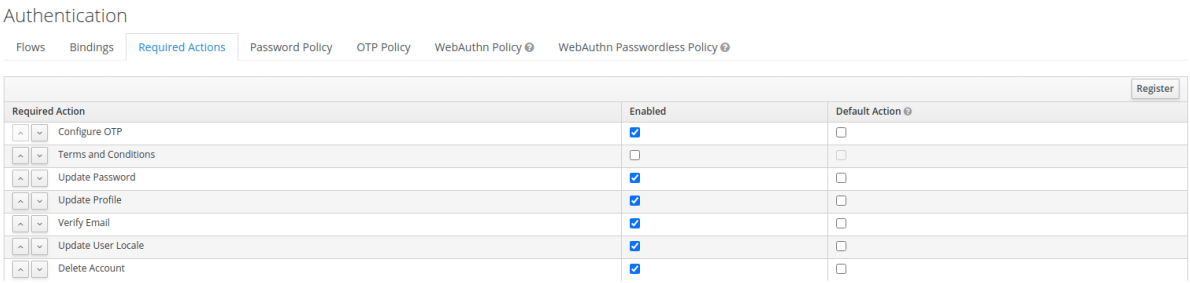
5.8.1. 启用删除帐户功能

您可以在 **所需的 Actions** 选项卡中启用此功能。

流程

1. 在菜单中，单击 **Authentication**。
2. 点 **所需的 Actions** 选项卡。
3. 在 **Delete Account** 行中，选择 **Enabled**。

删除所需操作标签的帐户



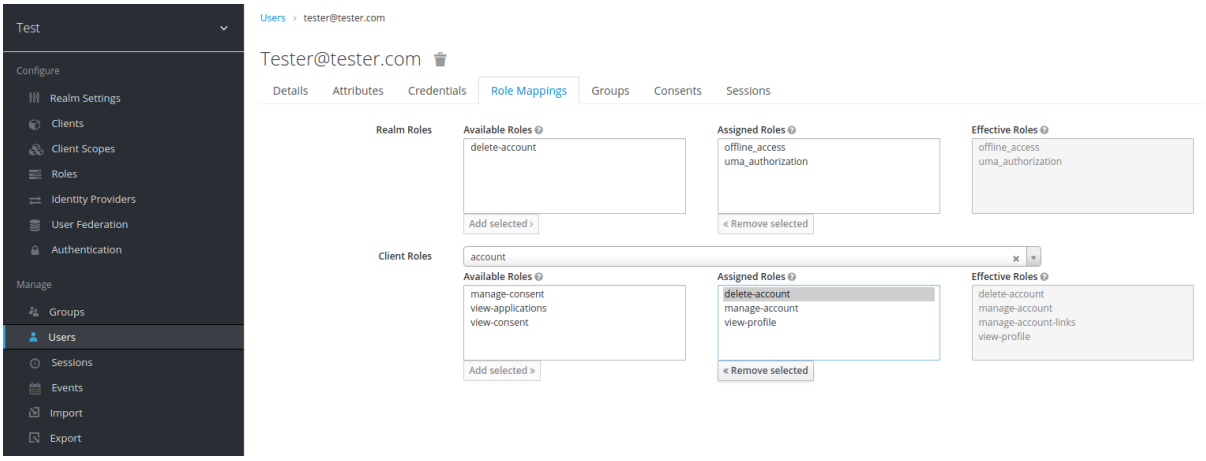
5.8.2. 授予用户 delete-account 角色

您可以为特定用户赋予允许删除帐户的角色。

流程

1. 点菜单中的 **Users**。
2. 选择用户。
3. 点 **Role Mappings** 选项卡。
4. 从 **Client Roles** 列表中，选择 **account**。
5. 在 **Available Roles** 下，选择 **delete-account**。
6. 点 **Add selected**。

delete-account 角色

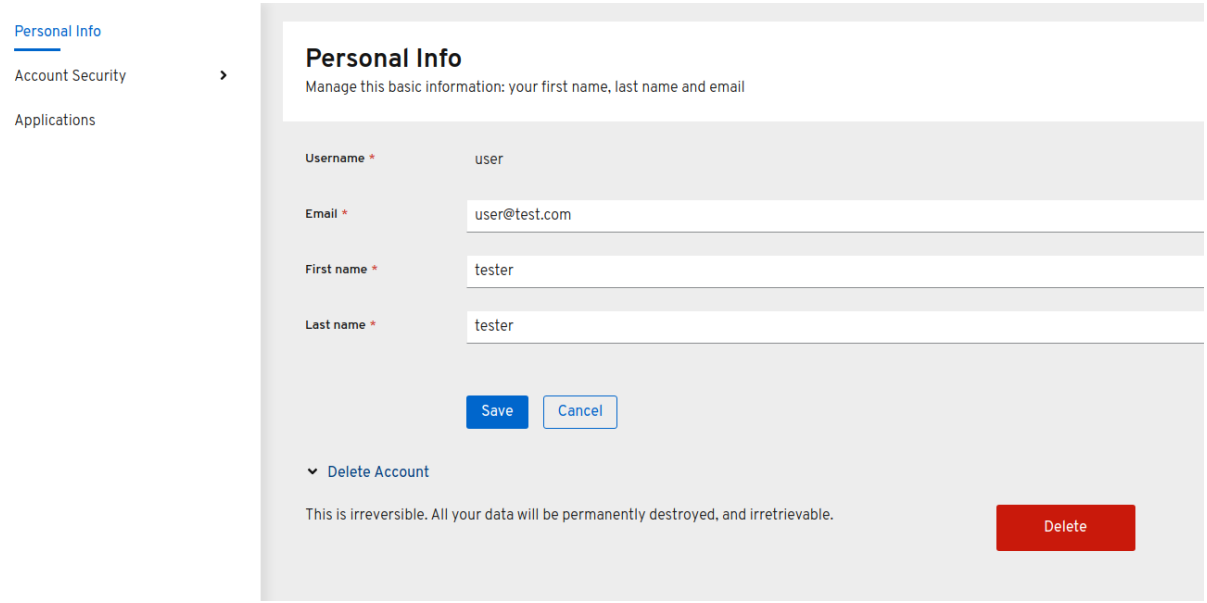


5.8.3. 删除您的帐户

拥有 **delete-account** 角色后，您可以删除自己的帐户。

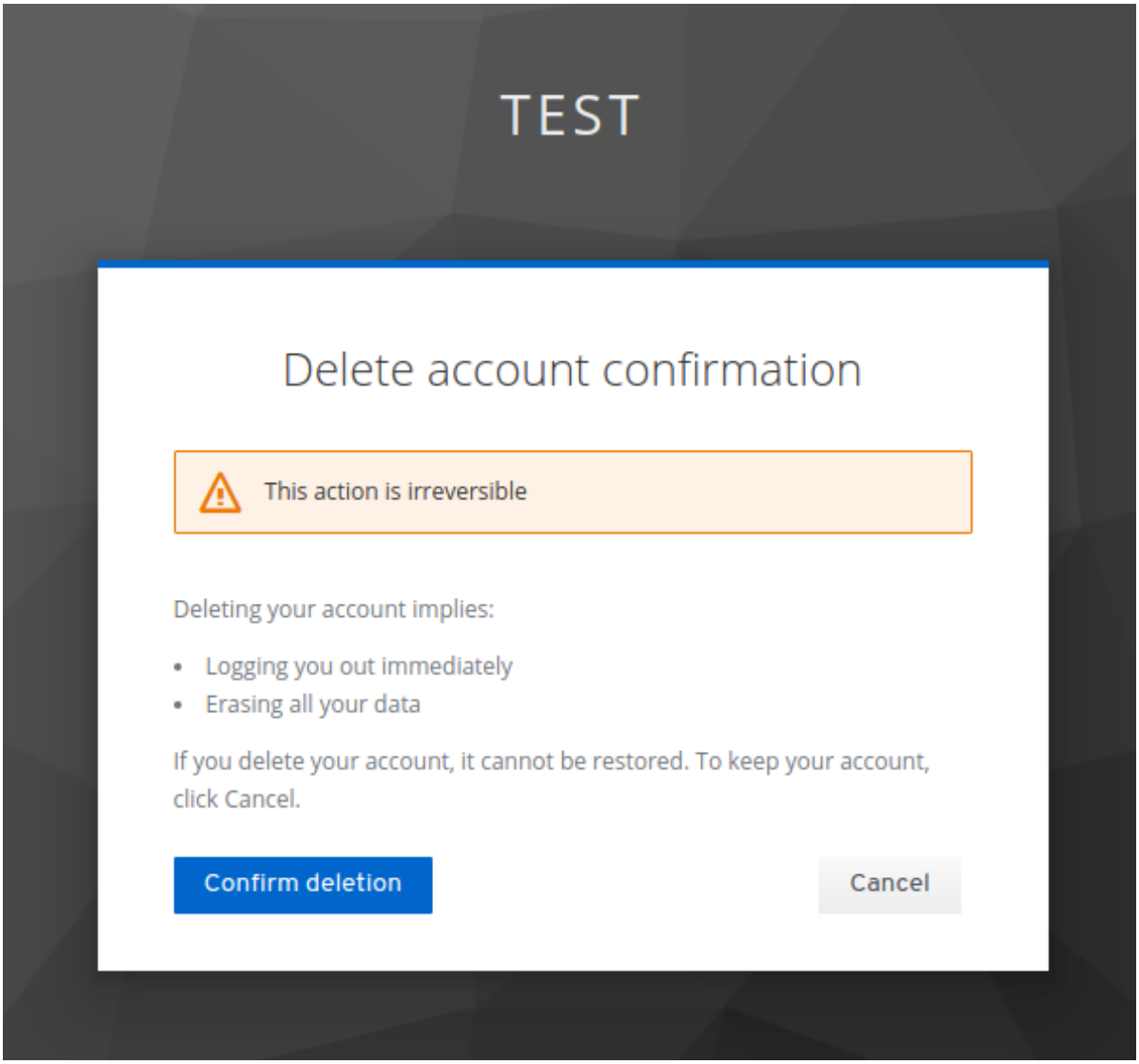
1. 登录到帐户控制台。
2. 在 **Personal Info** 页面的底部，点 **Delete Account**。

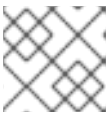
删除帐户页面



3. 输入您的凭证并确认删除。

删除确认





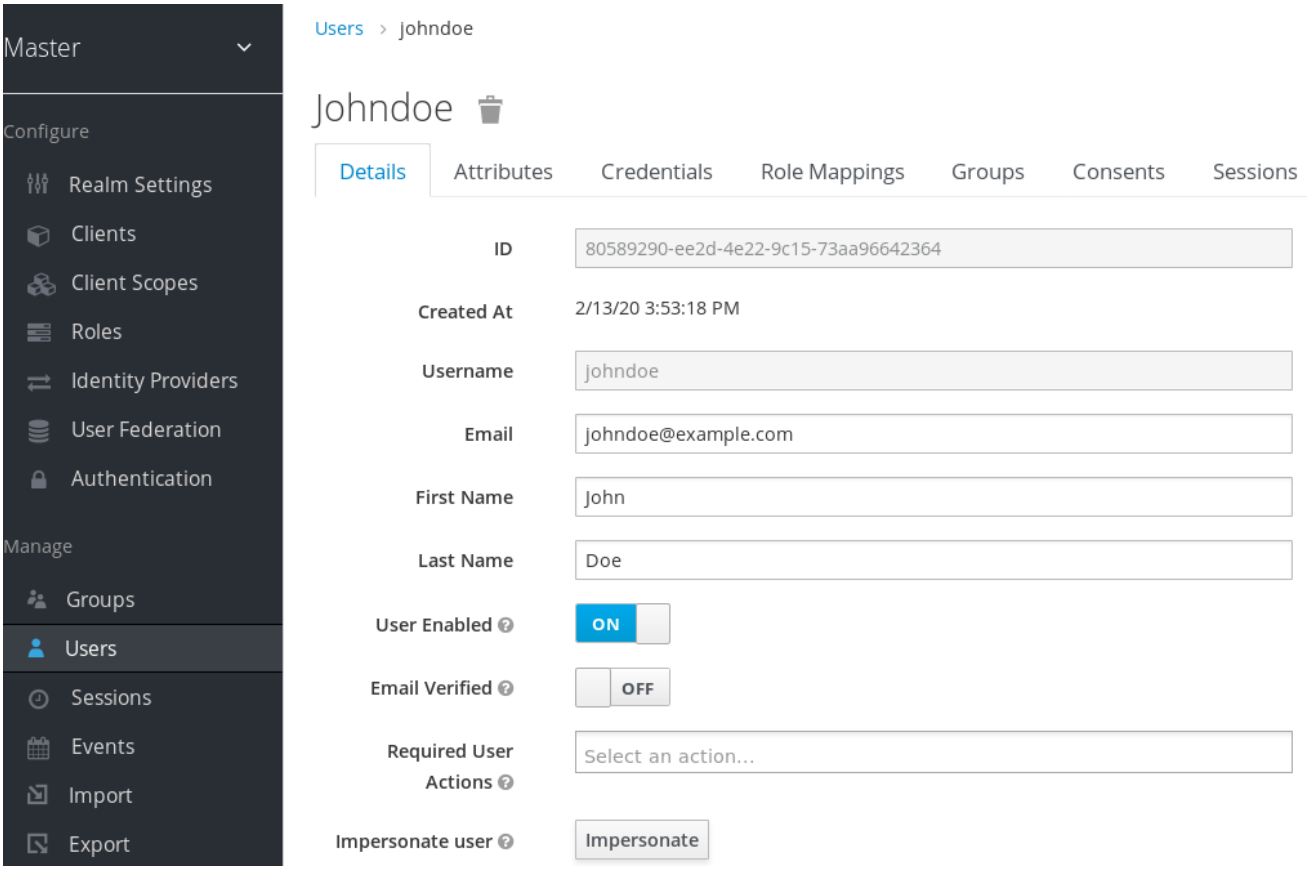
注意

此操作不可逆。将删除您在 Red Hat Single Sign-On 中的所有数据。

5.9. 模拟用户

具有适当权限的管理员可模拟用户。例如，如果用户在应用程序中遇到错误，管理员可以模拟用户来调查或复制问题。

任何在 realm 中具有 **模拟** 角色的用户都可以模拟用户。



- 如果管理员和用户处于同一域中，则管理员将注销并自动以模拟用户的身份登录。
- 如果管理员和用户在不同的域中，管理员将仍登录，另外还将以该用户的身份登录。

在两个实例中，会显示模拟用户的 **User Account Management** 页面。

您可以从 **Users** 页面中的 **Details** 选项卡访问 **Impersonate** 按钮。

其他资源

- 有关分配管理权限的更多信息，请参阅 管理控制台访问控制 章节。

5.10. 启用 RECAPTCHA

为保护注册免受 bots 的注册，Red Hat Single Sign-On 与 Google reCAPTCHA 集成。

启用 reCAPTCHA 后，您可以编辑登录中的 **register.ftl**，以配置注册页面上的 reCAPTCHA 按钮的放置和样式。

流程

1. 在浏览器中输入以下 URL:

```
https://developers.google.com/recaptcha/
```

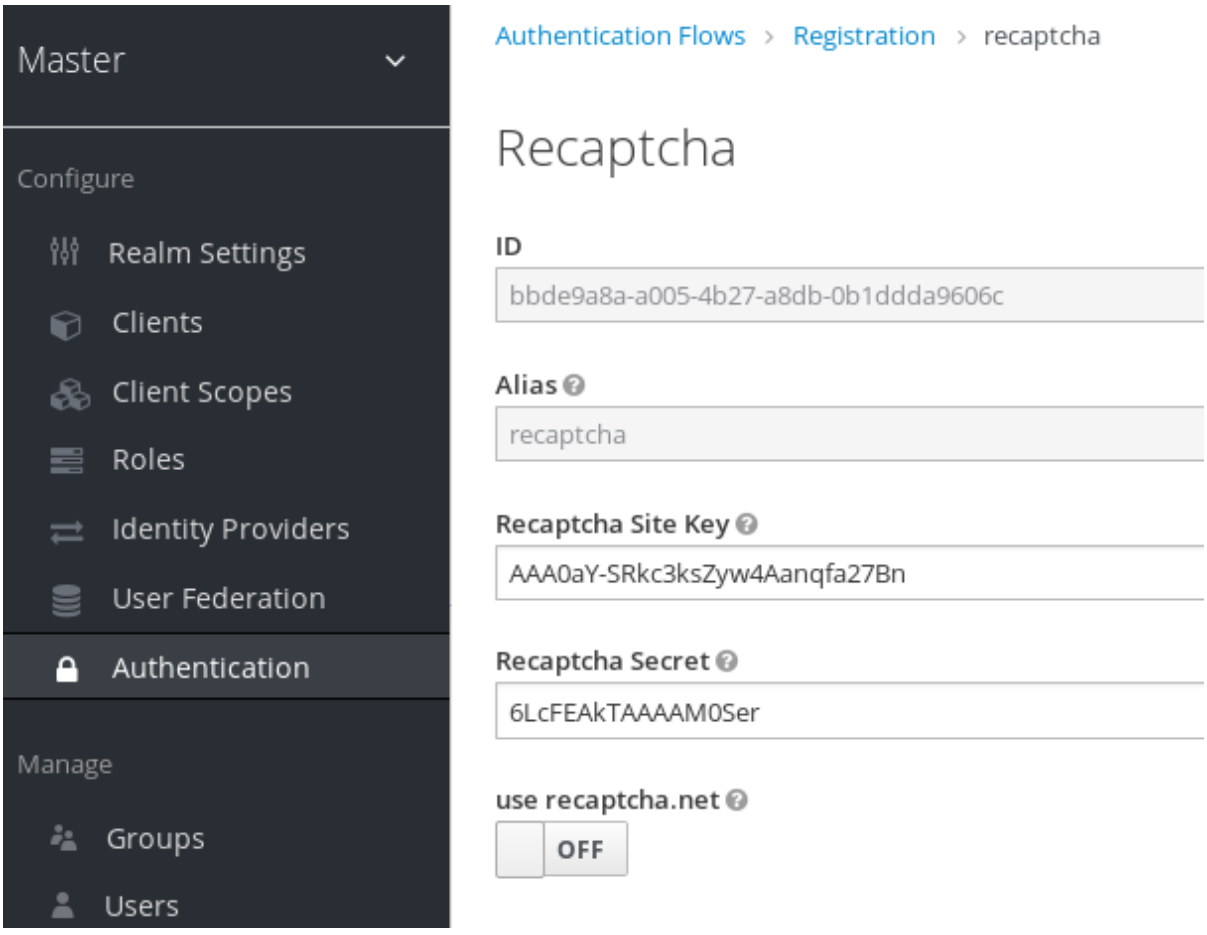
2. 创建 API 密钥以获取您的 reCAPTCHA site key 和 secret。请注意，reCAPTCHA site key 和 secret 以供以后在此流程中使用。

注意

默认情况下，localhost 可以正常工作。您不必指定域。

3. 导航到 Red Hat Single Sign-On 管理控制台。
4. 在菜单中，单击 **Authentication**。
5. 点 **Flows** 选项卡。
6. 从下拉菜单中选择 **Registration**。
7. 将 **reCAPTCHA** 要求设置为 **Required**。这将启用 reCAPTCHA。
8. 点 reCAPTCHA 流条目右侧的 **Actions**。
9. 点 **Config** 链接。

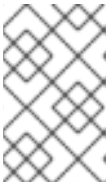
Recaptcha config 页



a. 输入由 Google reCAPTCHA 网站生成的 **Recaptcha Site Key**。

b. 输入由 Google reCAPTCHA 网站生成的 **Recaptcha Secret**。

10. 授权 Google 使用注册页面作为 iframe。



注意

在 Red Hat Single Sign-On 中，网站不能在 iframe 中包含一个登录页面对话框。这个限制是防止点击攻击攻击。您需要更改 Red Hat Single Sign-On 中设置的默认 HTTP 响应标头。

a. 点菜单中的 **Realm Settings**。

b. 点 **Security Defenses** 选项卡。

c. 在 **X-Frame-Options** 标头的字段中输入 **https://www.google.com**。

d. 在 **Content-Security-Policy** 标头的字段中输入 **https://www.google.com**。

其他资源

- 有关扩展和创建它们的更多信息，请参阅 服务器开发人员指南。

5.11. 定义用户配置集

在 Red Hat Single Sign-On 中，用户与一组属性关联。这些属性用于更好地描述和识别 Red Hat Single Sign-On 中的用户，以及传递给应用程序的附加信息。

用户配置集定义了一条明确的 schema，用于代表用户属性以及如何在域中管理。通过提供对用户信息的一致视图，管理员可以控制管理属性的不同方面，并更容易扩展红帽单点登录以支持附加属性。

除其他功能外，用户配置集使管理员能够：

- 为用户属性定义 schema
- 根据上下文信息（例如，仅要求用户或管理员）或同时定义属性是否需要属性，或根据所请求的范围。）
- 定义特定权限以查看和编辑用户属性，从而遵循强大的隐私要求，其中第三方（包括管理员）无法看到或更改某些属性。
- 动态强制实施用户配置集合规性，以便始终更新用户信息并与属性相关的元数据和规则
- 利用内置的验证器或编写自定义的验证规则，以每个attribute 为基础定义验证规则
- 根据属性定义，用户可以动态呈现与帐户控制台中的注册、更新配置文件、代理和个人信息等交互，而无需手动更改它们。

User Profile 功能由 User Profile SPI 支持。默认情况下，这些功能被禁用，域被配置为使用默认配置，以便向后兼容旧行为。

注意

旧行为是在管理用户 root 属性（如用户名、电子邮件、第一和姓氏）时使用的默认限制，而不考虑如何管理自定义属性的任何限制。有关用户流（如注册、配置集更新、代理）并通过帐户控制台管理帐户，用户将限制为使用前所述属性来更改主题模板以支持附加属性。

如果您已经使用 Red Hat Single Sign-On，旧的行为是您目前正在使用的行为。

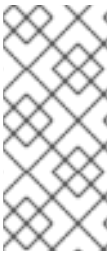
与传统行为不同，声明性提供程序为您提供了很多灵活性，以通过管理控制台和明确定义的 JSON 模式将用户配置文件配置定义为域。

在下一部分中，我们将了解如何使用声明性提供程序定义您自己的用户配置文件配置。

注意

在未来，红帽单点登录将不再支持传统行为。理想情况下，您应该开始关注 User Profile 提供的新功能，并相应地迁移您的域。

5.11.1. 启用用户配置集

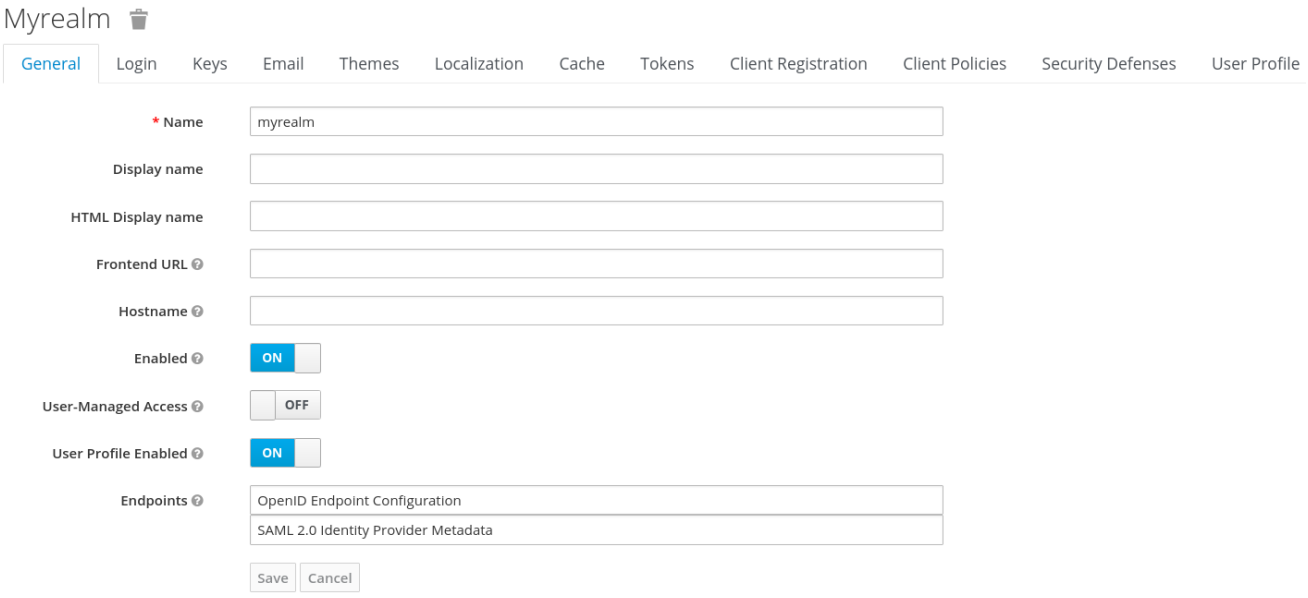


注意

声明性用户配置集 **是技术预览**，不被完全支持。此功能默认为禁用。

要使用 **-Dkeycloak.profile=preview** 或 **-Dkeycloak.profile.feature.feature.declarative_user_profile=enabled 来启用** 服务器。如需了解更多详细信息，请参阅 配置文件。

除了启用 **declarative_user_profile** 功能外，您应该为域启用用户配置文件。为此，可单击左侧菜单中的 **Realm Settings** 链接，再打开 **User Profile Enabled** switch。



启用并单击 **Save** 按钮后，您可以从中访问 **User Profile** 选项卡，从中可以管理用户属性的配置。

通过为域启用用户配置文件，Red Hat Single Sign-On 将会根据用户配置集配置来施加额外的限制如何管理属性。总之，以下列出了在启用该功能时应该期望的内容：

- 从管理的角度看，用户详情页面上的 **Attributes** 选项卡将仅显示用户配置文件配置中定义的属性。根据每个属性定义的条件也会在管理属性时考虑。
- 用户面临的表单，如帐户控制台中的注册、更新配置文件、代理和个人信息，将根据用户配置文件配置动态呈现。为此，Red Hat Single Sign-On 将依赖于不同的模板来动态呈现这些表单。

在下一个主题中，我们将探索如何管理用户配置文件配置以及它对域的影响。

5.11.2. 管理用户配置集

用户配置集配置以每个域为基础进行管理。为此，请单击左侧菜单上的 **Realm Settings** 链接，然后单击 **User Profile** 选项卡。

User Profile 选项卡

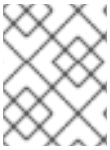
Attributes　Attribute Groups　JSON Editor

Create

Name	Display name	Attribute Group	Actions
username	${username}		Edit Delete
email	${email}		Edit Delete
firstName	${firstName}		Edit Delete
lastName	${lastName}		Edit Delete

在 **Attributes** 子选项卡中，您有一个当前与用户配置文件关联的属性列表。默认情况下，配置基于用户 root 属性创建，每个属性在验证和权限方面都会配置一些默认值。

在 **"属性** 组"子选项卡中，您可以管理属性组。属性组允许您关联属性，以便在呈现用户面向形式时一起显示它们。



注意

现在，属性组仅用于渲染目的，而在将来，它们还应启用与它们链接的属性定义的顶级配置。

在 **JSON Editor** 子选项卡中，您可以使用定义的 JSON 模式查看和编辑配置。在其他任何标签页中反映到此选项卡中的 JSON 配置时，您所做的任何更改都会反映出来。

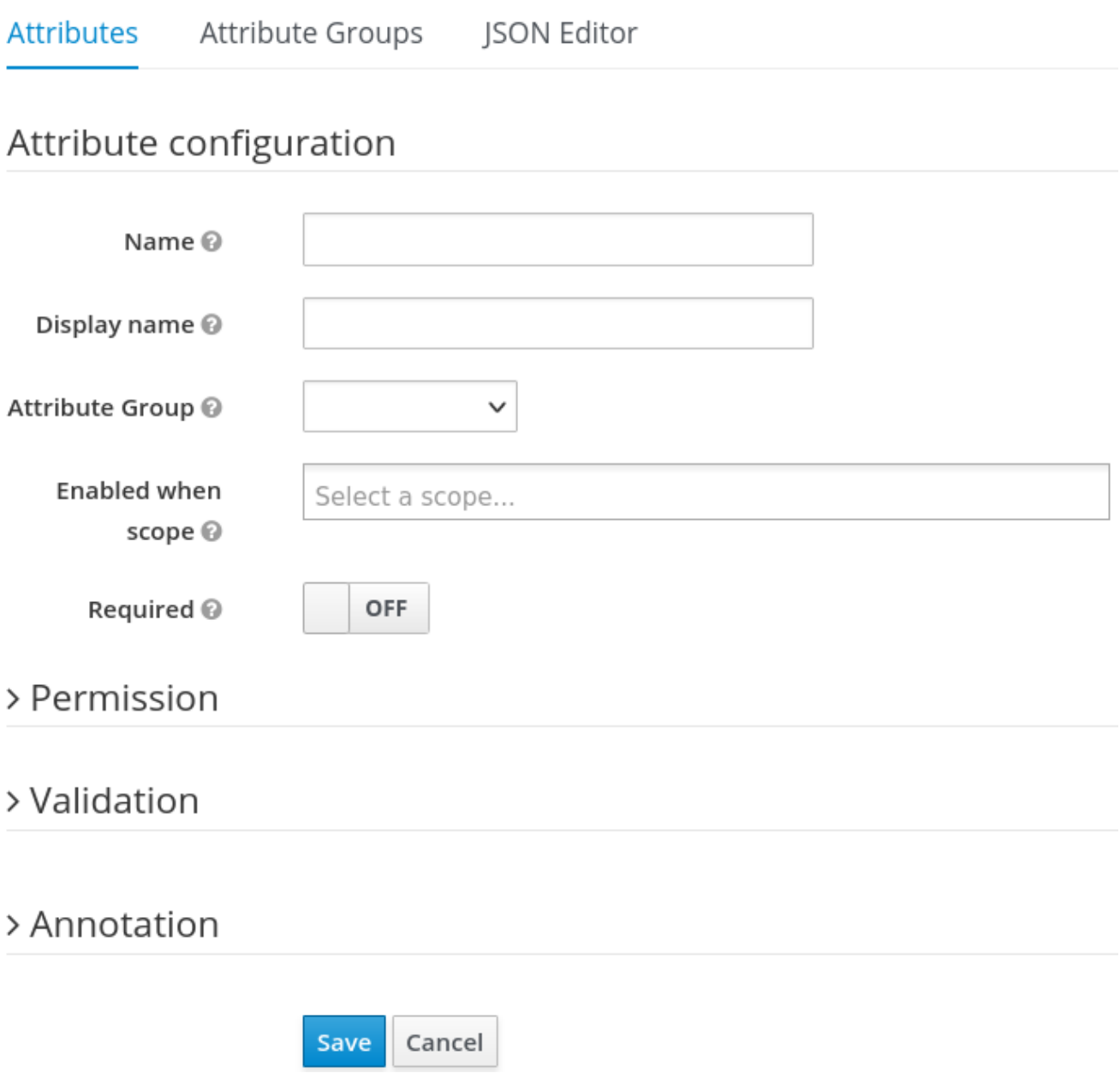
在下一部分中，您将了解如何从 **属性** 子选项卡中管理配置。

5.11.3. 管理属性

在 **Attributes** 子选项卡中，您可以创建、编辑和删除与用户配置文件关联的属性。

要定义新属性并将其与用户配置文件关联，请点属性列表右上角的 **Create** 按钮。

属性配置



在配置属性时，您可以定义以下设置：

Name

属性的名称。

显示名称

属性的用户友好的名称，主要在呈现面向用户表单时使用。它支持国际化，以便可以从消息捆绑包加载值。

属性组

属性组（若有）。

在范围时启用

允许您定义范围列表来动态启用属性。如果没有设置，则 属性总是被启用，在管理用户配置集以及渲染面向用户的表单时始终强制执行其限制。否则，只有在客户端请求列表中任何范围时，同样的限制才会应用。

必需

根据需要设置 属性。如果没有启用，则属性是可选的。否则，用户必须和管理员提供属性才能使用户或管理员仅根据客户端所请求的范围而必要的属性。

权限

在本节中，您可以为用户和管理员定义读取和写入权限。

验证

在本节中，您可以定义管理属性值时要执行的验证。Red Hat Single Sign-On 提供了一组内置验证器，您可以选择它自行添加。

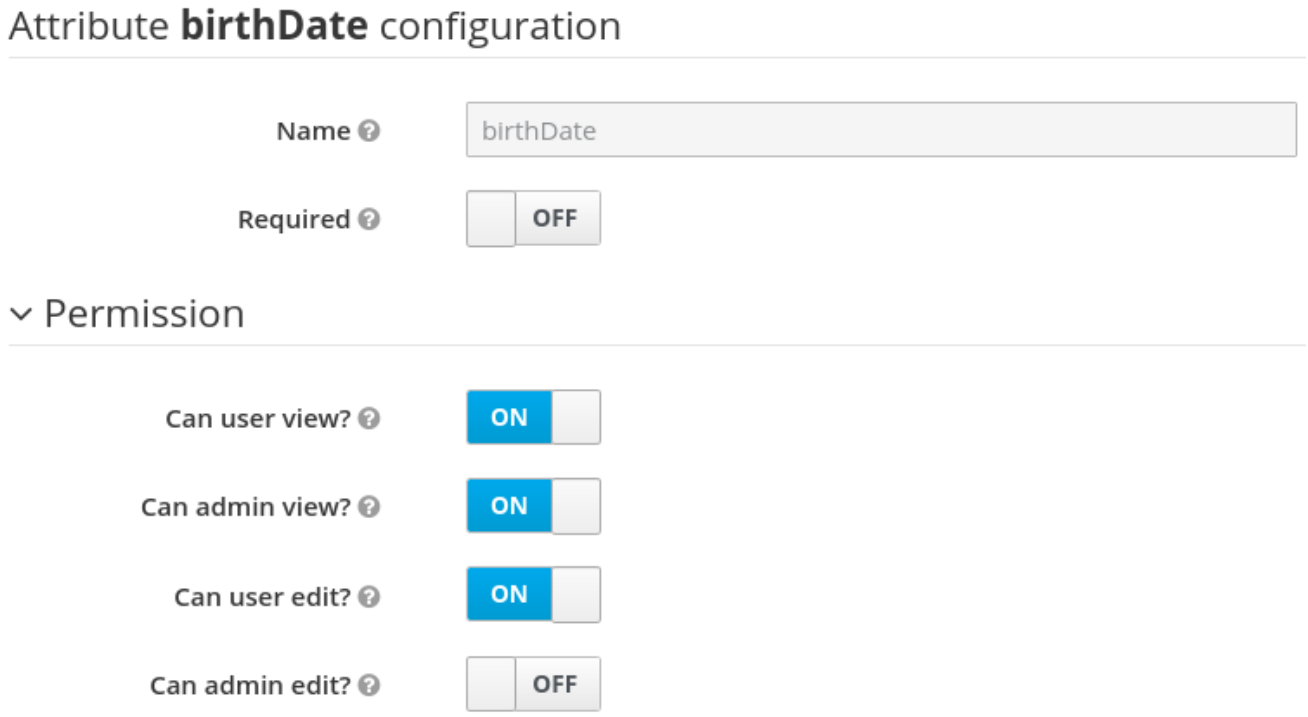
注解

在此部分中，您可以将注解与属性关联。注解主要用于将额外元数据传递给 frontend 以便进行渲染。

5.11.3.1. 管理权限

在 **Permission** 部分中，您可以定义访问权限用户级别，管理员必须读取和写入属性。

属性权限



因此，您可以使用以下设置：

用户是否可以查看？

如果启用，用户可以查看 属性。否则，用户无法访问 属性。

用户是否可以编辑？

如果启用，用户可以查看和编辑 属性。否则，用户无权写入属性。

管理可以查看吗？

如果启用，管理员可以查看 属性。否则，管理员无法访问 属性。

管理员是否可以编辑？

如果启用，管理员可以查看和编辑 属性。否则，管理员无权写入属性。

注意

当您创建属性时，没有权限设置为 属性。有效地，用户或管理员无法访问该属性。创建属性后，请确保相应地设置该属性仅对目标受众可见的权限。

权限会对如何管理属性以及属性呈现方式直接影响。

例如，通过将属性标记为只能查看属性，则管理员在通过管理控制台管理用户时无法访问 属性（从用户 API 中）。另外，用户在更新其配置集时无法更改属性。如果从现有的身份存储（解码）获取用户属性，并且您希望使属性对用户可见，而无需通过源身份存储更新其他属性，则比较有趣的配置。

同样，您还可以将属性标记为只针对用户具有只读访问权限的管理员可写属性。在这种情况下，只有管理员才可以管理属性。

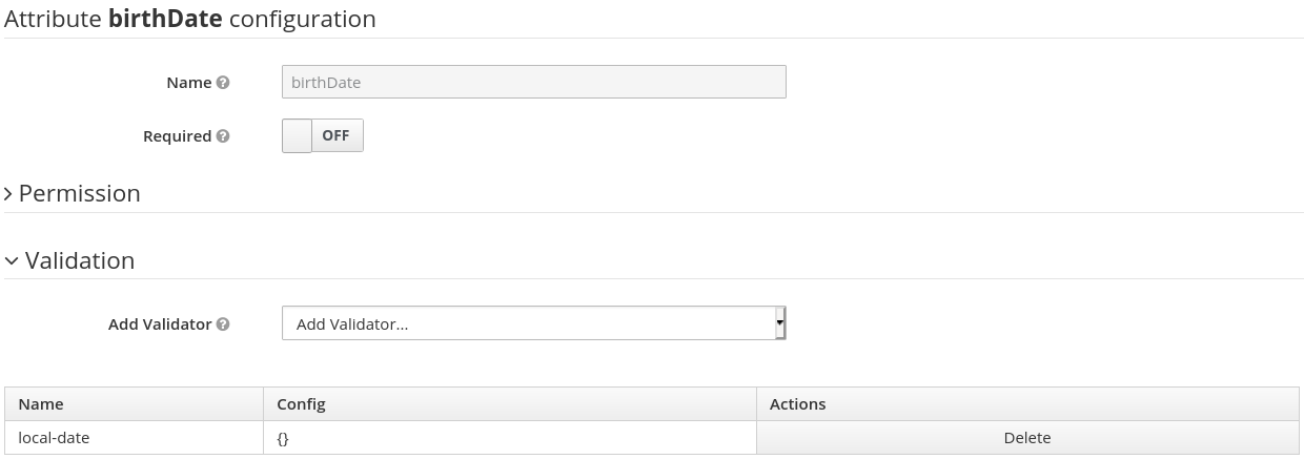
根据您的隐私要求，您可能还希望管理员无法访问属性，但用户具有读写权限。

每次向用户配置文件配置添加新属性时，务必设置正确的权限。

5.11.3.2. 管理验证

在 **Validation** 部分中，您可以从不同类型的验证中选择，以确保属性值符合特定的规则。

属性验证



Red Hat Single Sign-On 提供了不同的验证程序：

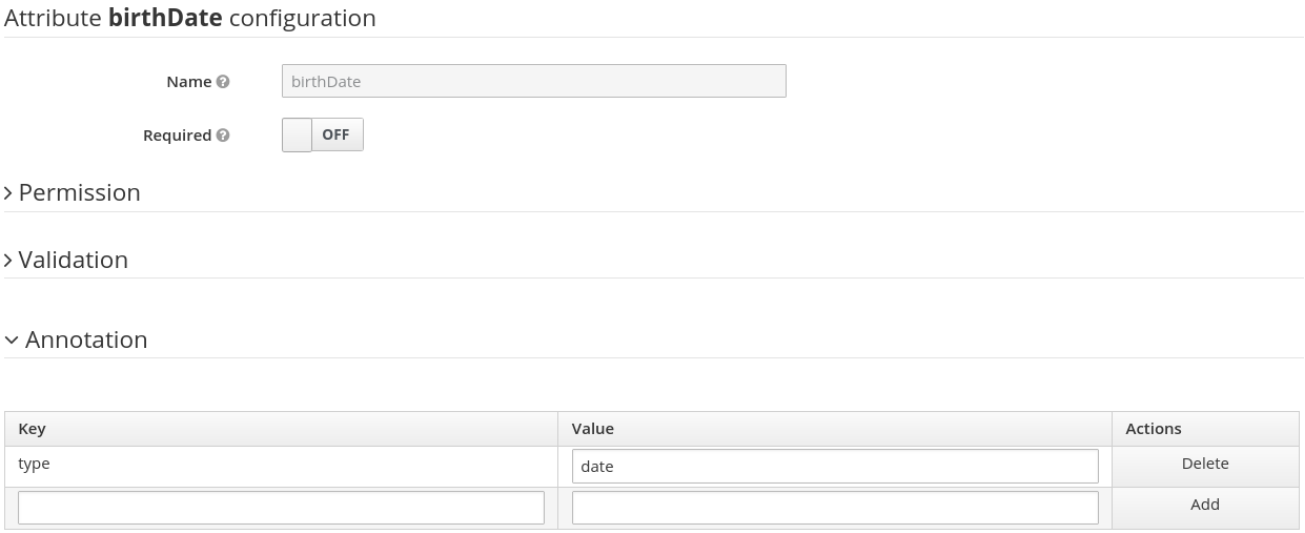
Name	描述	Configuration
长度	根据最小和最大长度检查字符串值的长度。	**min** ：一个整数来定义允许的最短长度。 **max** ：一个整数来定义允许的最大长度。 **trim-disabled** ：一个布尔值，用于定义在验证前是否修剪该值。
整数	检查该值是否为整数，并在较低和/或高范围内。如果没有定义范围，则验证器将仅检查该值是否有效。	**min** ：一个整数来定义较低范围。 **max** ：一个整数来定义上限。
double	检查该值是否是双，在较低和/或高范围内。如果没有定义范围，则验证器将仅检查该值是否有效。	**min** ：一个整数来定义较低范围。 **max** ：一个整数来定义上限。
uri	检查该值是否为有效的 URI。	None

Name	描述	Configuration
pattern	检查值是否与特定 RegEx 模式匹配。	**Pattern** ：验证值时要使用的 RegEx 模式。 **error-message** ：i18n bundle 中错误消息的密钥。如果未设置通用消息。
email	检查该值是否具有有效的电子邮件格式。	None
local-date	根据 realm 和/或用户区域设置，检查该值是否具有有效的格式。	None
person-name-prohibited-characters	检查该值是否为用于攻击的额外障碍，如脚本注入。验证基于一个默认的 RegEx 模式，它阻断了不常在人名称中的字符。	**error-message** ：i18n bundle 中错误消息的密钥。如果未设置通用消息。
username-prohibited-characters	检查该值是否为用于攻击的额外栏方，如脚本注入。验证基于一个默认 RegEx 模式，它阻断了在用户名中不常见的字符。	**error-message** ：i18n bundle 中错误消息的密钥。如果未设置通用消息。
选项	检查值是否来自定义的允许值集合。用于验证通过 select 和 multiselect 字段输入的值很有用。	选项 ：包含允许值的字符串数组。

5.11.3.2.1. 管理注解

要将额外信息传递给 frontends，可以使用注解来解码属性来指示属性是如何呈现的。此功能在扩展 Red Hat Single Sign-On themes 时，根据与属性关联的注解动态呈现页面时，此功能很有用。此机制 用于为属性 配置 Form 输入文件。

属性注解



5.11.4. 管理属性组

在 **Attribute Groups** 子选项卡中，您可以创建、编辑和删除属性组。属性组允许您为关联属性定义容器，以便在面向用户的表单时将它们呈现在一起。

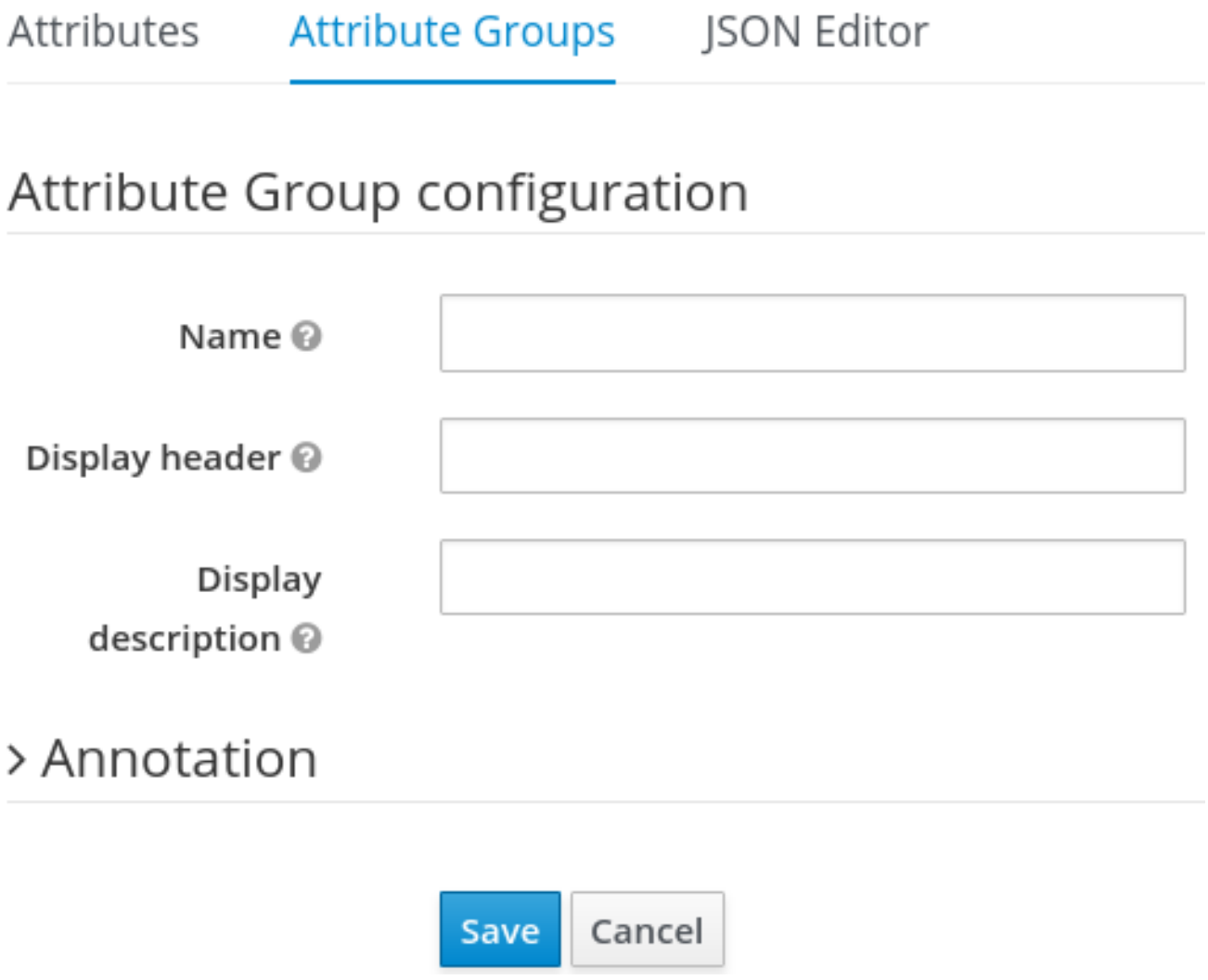
闲置组列表

注意

您不能删除绑定到属性的属性组。为此，您应该首先更新属性以删除绑定。

要创建新组，请单击属性组列表右上角的 **Create** 按钮。

属性组配置



在配置组时，您可以定义以下设置：

Name

组名称。

显示名称

对组的用户友好名称，在呈现面向用户的表单时主要使用。它支持国际化，以便可以从消息捆绑包加载值。

显示描述

用户友好的文本，在呈现面向用户的表单时将显示为工具提示。

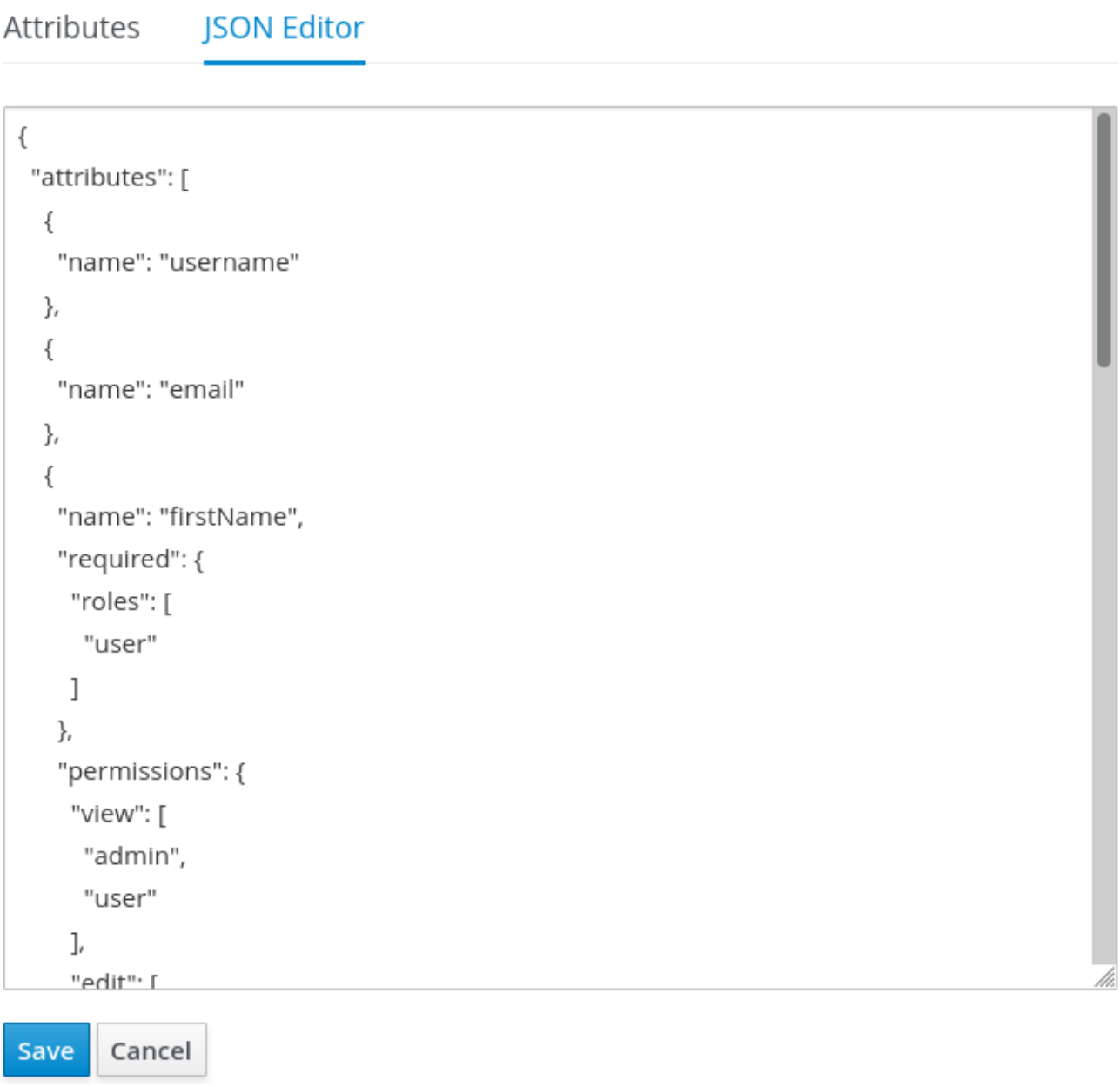
注解

在此部分中，您可以将注解与属性关联。注解主要用于将额外元数据传递给 frontend 以便进行渲染。

5.11.5. 使用 JSON 配置

用户配置集配置使用定义良好的 JSON 模式存储。您可以通过点击 **JSON Editor** sub-tab 直接从编辑用户配置集配置中选择。

JSON 配置



JSON 模式定义如下：

```
{
  "attributes": [
    {
      "name": "myattribute",
      "required": {
        "roles": [ "user", "admin" ],
        "scopes": [ "foo", "bar" ]
      },
      "permissions": {
        "view": [ "admin", "user" ],
        "edit": [ "admin", "user" ]
      },
      "validations": {
        "email": {},
        "length": {
          "max": 255
        }
      },
      "annotations": {
        "myannotation": "myannotation-value"
      }
    }
  ],
  "groups": [
    {
      "name": "personalInfo",
      "displayHeader": "Personal Information"
    }
  ]
}
```

架构支持尽可能多的属性。

对于每个属性，您应定义一个 **名称**，以及可选的 **所需、权限 和注解** 设置。

5.11.5.1. 所需的属性

required 设置定义属性是否是必需的。Red Hat Single Sign-On 允许您根据不同条件设置属性。

当 **required** 设置定义为空对象时，则始终需要。

```
{
  "attributes": [
    {
      "name": "myattribute",
      "required": {}
    }
  ]
}
```

另一方面，您可以选择为用户或管理员做出所需的属性。另外，只有在 Red Hat Single Sign-On 中用户进行身份验证时，才要求将 属性标记为必需属性。

要标记用户和/或管理员所需的属性，请按如下所示设置 **roles** 属性：

```
{
```

```
  "attributes": [
   {
    "name": "myattribute",
    "required": {
     "roles": ["user"]
    }
  ]
}
```

roles 属性需要一个数组，其值可以是 **user** 或 **admin**，具体取决于用户还是管理员所需要的属性。

同样，您可以选择在验证用户时，选择为客户端请求一个或多个范围时所需的属性。为此，您可以使用 **scopes** 属性，如下所示：

```
{
  "attributes": [
   {
    "name": "myattribute",
    "required": {
     "scopes": ["foo"]
    }
  ]
}
```

scopes 属性是一个数组，其值可以是代表客户端范围的任何字符串。

5.11.5.2. 权限属性

属性级 **权限** 属性可用于定义属性的读取和写入权限。权限会根据用户或管理员是否能够对属性来执行这些操作。

```
{
  "attributes": [
   {
    "name": "myattribute",
    "permissions": {
     "view": ["admin"],
     "edit": ["user"]
    }
  ]
}
```

view 和 **edit** 属性都要求一个数组，其值可以是 **用户或 admin**，具体取决于属性是否可以被分别查看或管理员编辑。

当授予**edit**权限时，而会获得 **view** 权限。

5.11.5.3. annotations 属性

属性级 **注解** 属性可用于将额外元数据与属性关联。注解主要用于根据用户配置集配置传递属性到 frontends rendering 用户属性的额外信息。每个注解都是键/值对。

```
{
  "attributes": [
```

```
    {
      "name": "myattribute",
      "annotations": {
        "foo": ["foo-value"],
        "bar": ["bar-value"]
      }
    ]
  }
```

5.11.6. 使用动态表单

User Profile 的主要功能之一是可以根据属性元数据动态呈现面向用户表单。当您为域启用了该功能时，会使用特定的主题模板根据用户配置集配置来动态显示页面，如注册和更新配置文件。

也就是说，如果默认渲染机制根据您的需要，则应该不需要自定义模板。如果您仍然需要将自定义自定义到它们，请参考以下模板：

模板	描述
base/login/update-user-profile.ftl	呈现更新配置集页面的模板。
base/login/register-user-profile.ftl	呈现注册页面的模板。
base/login/idp-review-user-profile.ftl	在生成用户通过代理时，渲染查看/更新用户配置集的页面的模板。
base/login/user-profile-commons.ftl	基于属性配置以表单形式呈现输入字段的模板。在上述所有三个页面模板中使用。此处可实施新的输入类型。

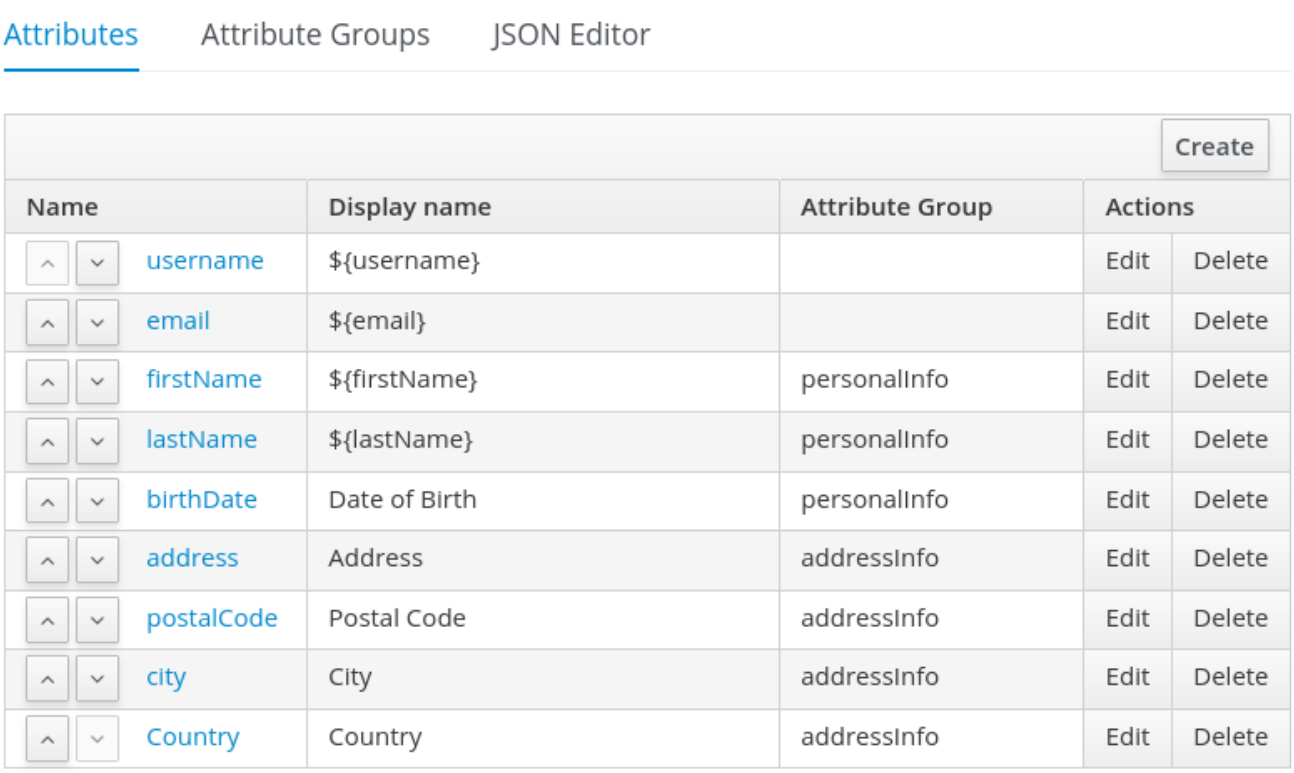
默认渲染机制提供以下功能：

- 根据设置为属性的权限动态显示字段。
- 根据设置为属性的约束，为必填字段动态呈现标记。
- 动态呈现字段输入类型（文本、日期、数字、选择、多选）设置为属性。
- 根据设置为属性的权限，动态呈现只读字段。
- 根据分配给属性的顺序设置动态订购字段。
- 属于同一属性组的动态组字段。

5.11.6.1. 排序属性

在属性列表页面中时，单击上下箭头和下箭头即可设置属性顺序。

排序属性

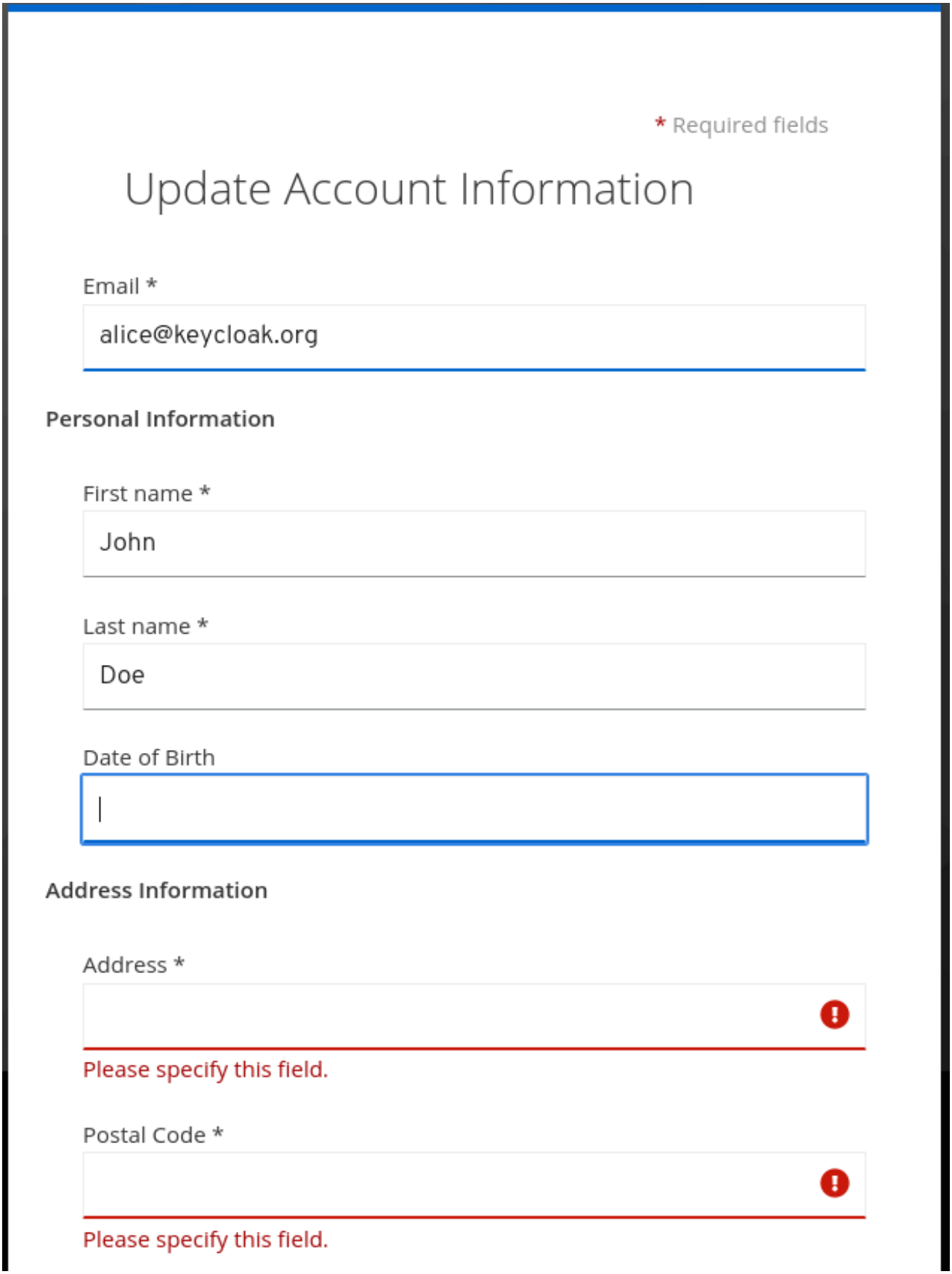


当字段以动态形式呈现时，会遵循您在本页中设置的顺序。

5.11.6.2. 分组属性

呈现动态表单时，它们将尝试将属于同一属性组的属性分组在一起。

动态更新配置文件表单



注意

当属性链接到属性组时，属性顺序也很重要，以确保同一组中的属性在相同的组标题中关闭。否则，如果组中的属性没有连续顺序，则可能会以动态的形式呈现同一组标头多次。

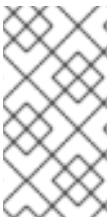
5.11.6.3. 为属性配置表单输入文件

Red Hat Single Sign-On 提供了内置注解，以配置哪些输入类型用于动态形式的属性及其视觉化的其他方面。

可用注解：

Name	描述
inputType	表单输入字段的类型。下表描述了可用类型。
inputHelperTextBefore	在输入字段之前呈现的帮助文本（再次）输入字段。此处可以使用直接文本或国际化模式（如 **${i18n.key}**）。在将文本呈现到页面时，文本不是 html 转义，因此您可以使用 HTML 标签来格式化文本，但您还必须正确转义 html 控制字符。
inputHelperTextAfter	在输入字段后（在下面）呈现帮助文本。此处可以使用直接文本或国际化模式（如 **${i18n.key}**）。在将文本呈现到页面时，文本不是 html 转义，因此您可以使用 HTML 标签来格式化文本，但您还必须正确转义 html 控制字符。
inputOptionsFromValidation	选择和多选类型的注解。用于获取输入选项的自定义属性验证的可选名称。请参阅以下 详细说明。
inputOptionLabelsI18nPrefix	选择和多选类型的注解。在 UI 中呈现选项的国际化密钥前缀。请参阅以下 详细说明。
inputOptionLabels	选择和多选类型的注解。可选映射来为选项定义 UI 标签（直接或使用国际化）。请参阅以下 详细说明。
inputTypePlaceholder	应用于字段的 HTML 输入 **占位符** 属性 - 指定一个简短提示，用于描述输入字段的预期值（如示例值或预期格式的简短描述）。简短提示显示在用户输入值前的输入字段中。
inputTypeSize	应用于字段的 HTML 输入 **大小** 属性 - 指定单行输入字段的字符，以字符为单位。对于基于 HTML **选择** 的字段，它通过显示的选项指定行数。可能无法工作，具体取决于使用的 css！
inputTypeCols	应用于字段的 HTML 输入 **cols** 属性 - 指定宽度（对于 **textarea** 类型）。可能无法工作，具体取决于使用的 css！
inputTypeRows	应用于字段的 HTML 输入 **行** 属性 - 为 **textarea** 类型指定十六进制值（以字符表示）。对于选择字段，它指定了带有显示选项的行数。可能无法工作，具体取决于使用的 css！

Name	描述
inputTypePattern	HTML 输入 **模式** 属性应用于提供客户端侧验证的字段 - 指定一个正则表达式，用于检查输入字段的值。对于单行输入很有用。
inputTypeMaxLength	应用于提供客户端验证的字段 HTML 输入 **maxlength** 属性 - 可以输入到输入的文本的最大长度。对文本字段很有用。
inputTypeMinLength	HTML 输入 **minlength** 属性应用于字段提供客户端侧验证 - 文本的最小长度，可以输入到输入字段。对文本字段很有用。
inputTypeMax	HTML 输入 **max** 属性应用于提供客户端侧验证的字段 - maximal 值，可在输入字段中输入。对于数字字段很有用。
inputTypeMin	HTML 输入 **min** 属性应用于提供客户端侧验证 - 最小值，可在输入字段中输入。对于数字字段很有用。
inputTypeStep	应用到字段的 HTML 输入 **步骤** 属性 - 在输入字段中指定法律编号之间的间隔。对于数字字段很有用。



注意

字段类型使用 HTML 表单字段标签和应用到它们的属性 - 它们基于 HTML 规格和浏览器支持。

视觉化呈现还取决于使用主题中应用的 cs 风格。

可用的 **inputType** 注解值：

Name	描述	使用的 HTML 标签
text	单行文本输入。	输入
文本区域	多行文本输入.	文本区域
select	常见单选输入。请参考 如何在下面配置选项的说明。	select
select-radiobuttons	单个选择通过一组单选按钮的输入。请参考 如何在下面配置选项的说明。	输入组
multiselect	常见多选输入.请参考 如何在下面配置选项的说明。	select

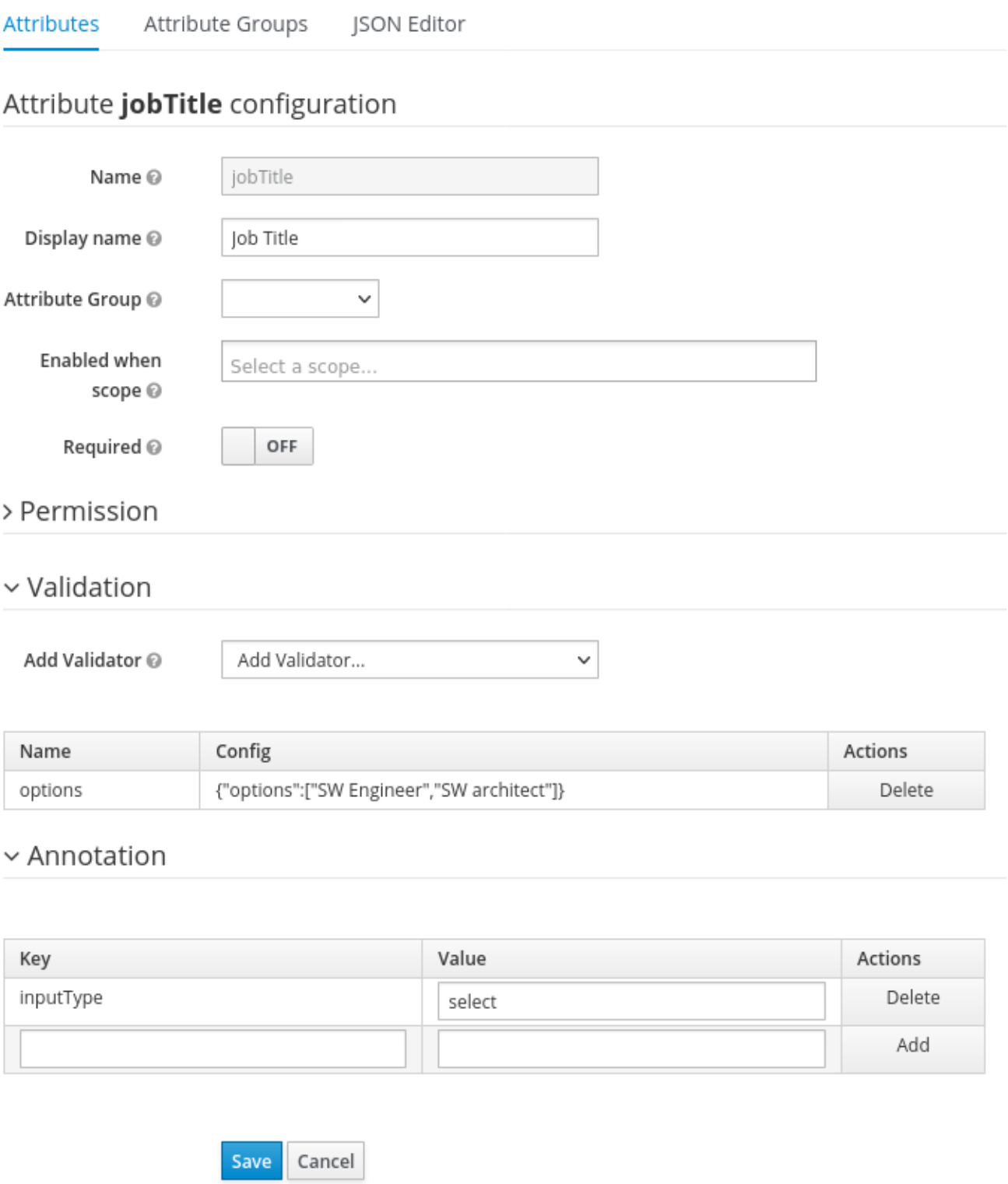
Name	描述	使用的 HTML 标签
multiselect-checkboxes	通过组复选框进行多选择输入。请参考 如何在下面配置选项的说明。	输入组
html5-email	基于 HTML 5 规格的电子邮件地址的单行文本输入。	输入
html5-tel	基于 HTML 5 规格的电话号码的单行文本输入。	输入
html5-url	基于 HTML 5 规格的 URL 的单行文本输入。	输入
html5-number	基于 HTML 5 规格的单行输入（整数或浮点，取决于 **step**）	输入
html5-range	基于 HTML 5 规格输入的数字幻灯片。	输入
html5-datetime-local	基于 HTML 5 规格的日期时间输入。	输入
html5-date	基于 HTML 5 规格的日期输入。	输入
html5-month	基于 HTML 5 规格的每月输入。	输入
html5-week	基于 HTML 5 规格的星期一输入。	输入
html5-time	基于 HTML 5 规格的时间输入。	输入

5.11.6.3.1. 定义选择和多选字段的选项

选择和多选择字段的选项从验证应用到属性，以确保 UI 中提供的验证和字段选项始终一致。默认情况下，选项取自内置 选项 验证。

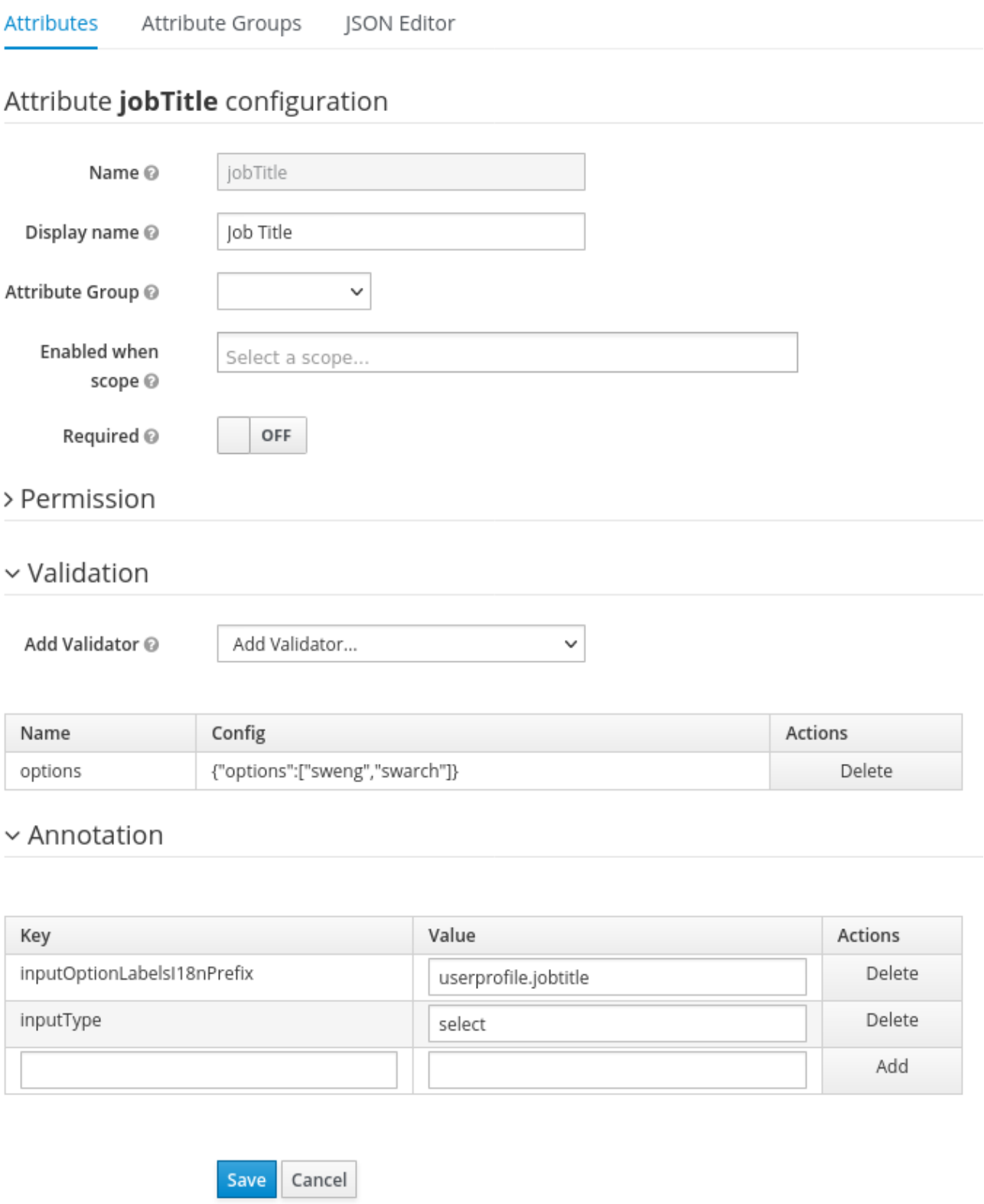
您可以使用各种方法为选择和多选选项提供 nice 人类可读标签。最简单的情况是，属性值与 UI 标签相同。本例中不需要任何额外的配置。

选项值与 UI 标签相同



如果属性值是不适合 UI 的 ID，您可以使用 **inputOptionLabelsI18nPrefix** 注解提供的简单国际化支持。它定义了国际化键的前缀，选项值是附加到这个前缀的点。

使用 i18n 键前缀进行 UI 标签的简单国际化



选项值的本地化 UI 标签文本必须由 **userprofile.jobtitle.sweng** 和 **userprofile.jobtitle.swarch** 键提供，然后使用常见的本地化机制。

您还可以使用 **inputOptionLabels** 注解为单个选项提供标签。它包含映射中选项 - 键的标签映射（在验证中定义），映射中的值是 UI 标签文本自身或其国际化模式（如 **${i18n.key}**）用于该选项。

注意

您必须使用 User Profile **JSON Editor** 输入 map 作为 **inputOptionLabels** 注解值。

没有国际化的单个选项直接输入标签示例：

```
"attributes": [
...
{
  "name": "jobTitle",
  "validations": {
    "options": {
      "options":[
        "sweng",
        "swarch"
      ]
    }
  },
  "annotations": {
    "inputType": "select",
    "inputOptionLabels": {
      "sweng": "Software Engineer",
      "swarch": "Software Architect"
    }
  }
}
...
]
```

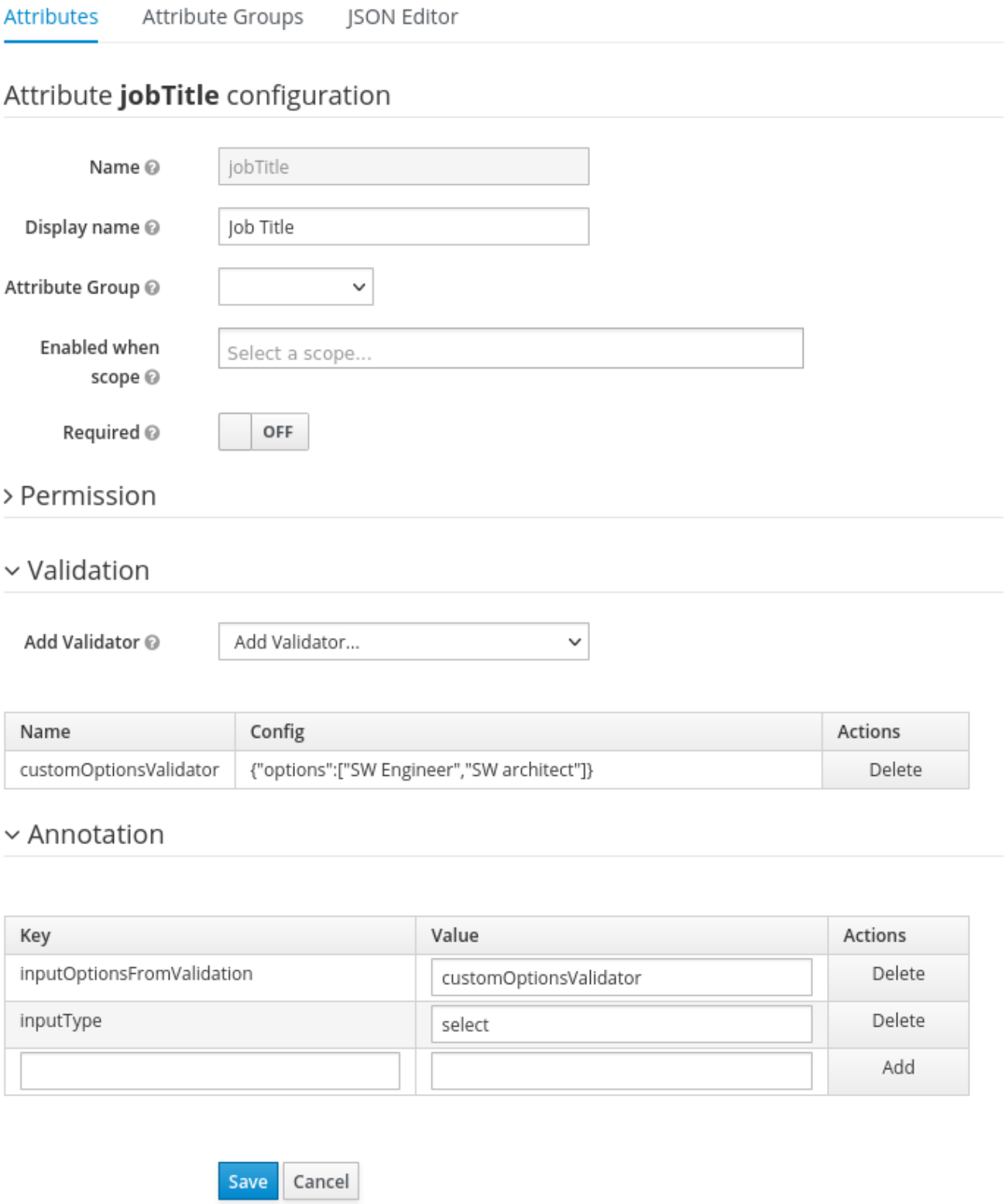
独立选项的国际化标签示例：

```
"attributes": [
...
{
  "name": "jobTitle",
  "validations": {
    "options": {
      "options":[
        "sweng",
        "swarch"
      ]
    }
  },
  "annotations": {
    "inputType": "select-radiobuttons",
    "inputOptionLabels": {
      "sweng": "${jobtitle.swengineer}",
      "swarch": "${jobtitle.swarchitect}"
    }
  }
}
...
]
```

本地化文本必须由 **jobtitle.swengineer** 和 **jobtitle.swarchitect** 键提供，使用常见的本地化机制。

自定义验证器可用于提供选项，因为 **inputOptionsFromValidation** attribute 注解。此验证必须具有提供选项**的** config 数组的选项。国际化的方式与内置 选项验证**的选项相同**。

自定义验证器提供的选项



5.11.7. 强制用户配置文件合规性

为确保用户配置集符合配置，管理员可以使用 **VerifyProfile** required 操作来最终强制用户在向 Red Hat Single Sign-On 进行身份验证时更新其配置集。

注意

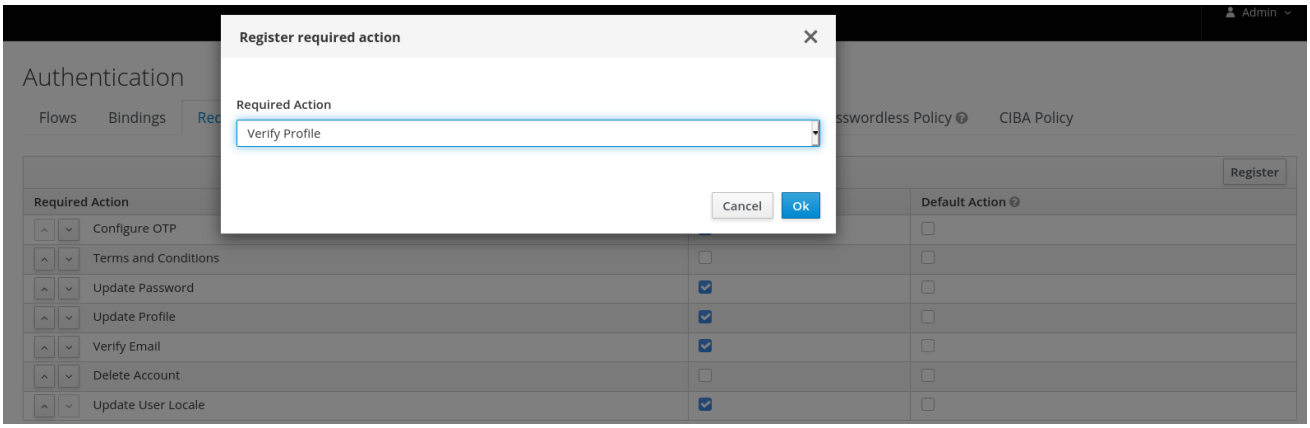
VerifyProfile 操作与 **UpdateProfile** 操作类似。但是，它会利用用户配置集提供的所有功能来自动强制实施与用户配置文件配置的合规性。

启用后，**VerifyProfile** 操作会在用户进行身份验证时执行以下步骤：

- 检查用户配置集是否与用户配置集配置完全符合该域。
- 如果没有，请在身份验证过程中执行额外的步骤，以便用户可以更新任何缺失或无效属性。
- 如果用户配置集与配置兼容，则不会执行额外的步骤，用户继续进行身份验证过程。

默认情况下，**VerifyProfile** 操作被禁用。要启用它，请单击左侧菜单中的 **Authentication** 链接，然后单击 **Required Actions** 选项卡。在这个标签页中，点 **Register** 按钮并选择 **VerifyProfile** action。

Registring VerifyProfile Required Action



5.11.8. 迁移到用户配置集

在为域启用用户配置文件功能前，您需要了解一些重要的注意事项。通过提供单一位置来管理属性元数据，该功能对可设置为用户的属性以及管理方式而言非常严格。

在用户管理方面，管理员只能管理用户配置文件配置中定义的属性。将任何其他属性设置为用户，在用户配置文件配置中尚未定义的任何属性都无法访问。建议您使用您要向用户或管理员公开的所有用户属性更新用户配置集配置。

同样的建议适用于访问用户 REST API 来查询用户信息。

关于 Red Hat Single Sign-On 内部用户属性，例如 **LDAP_ID**、**LDAP_ENTRY_DN** 或 **KERBEROS_PRINCIPAL**，如果您要访问这些属性，应在用户配置集配置中具有属性。建议将这些属性标记为只能供管理员查看，因此当通过管理控制台管理用户属性或通过用户 API 查询用户属性时，您可以查看它们。

就它们而言，如果您已经对旧模板进行了自定义（使用 root 属性进行硬编码），则您的自定义模板在呈现用户表单时不会被使用，但呈现这些表单的新模板会动态呈现这些表单。理想情况下，您应该避免对模板进行任何自定义，并尝试使用这些新模板提供的行为来动态呈现形式。如果它们仍不足以满足您的要求，您可以对其进行自定义，或提供任何反馈意见，以便我们讨论是否对新模板进行增强。

5.12. RED HAT SINGLE SIGN-ON 收集的个人数据

默认情况下，Red Hat Single Sign-On 会收集以下数据：

- 基本用户数据，如用户电子邮件、名字和姓氏。
- 使用社交登录时，用于社交帐户和对社交帐户引用的基本用户配置文件数据。
- 为审计和安全目的收集的设备信息，如 IP 地址、操作系统名称和浏览器名称。

Red Hat Single Sign-On 中收集的信息可高度自定义。在自定义自定义时适用以下指南：

- 注册和帐户表单可以包含自定义字段，如每天、天和国家/地区。管理员可以配置 Red Hat Single Sign-On，以从社交供应商或用户存储供应商（如 LDAP）检索数据。
- Red Hat Single Sign-On 会收集用户凭证，如密码、OTP 代码和 WebAuthn 公钥。此信息将被加密并保存在数据库中，因此 Red Hat Single Sign-On 管理员不可见。每种凭证类型都可以包含对管理员可见的非机密元数据，例如用于哈希密码的算法和用于哈希密码的哈希迭代数量。
- 启用授权服务和 UMA 支持后，Red Hat Single Sign-On 可以保存有关特定用户是所有者的一些对象的信息。

第 6 章 管理用户会话

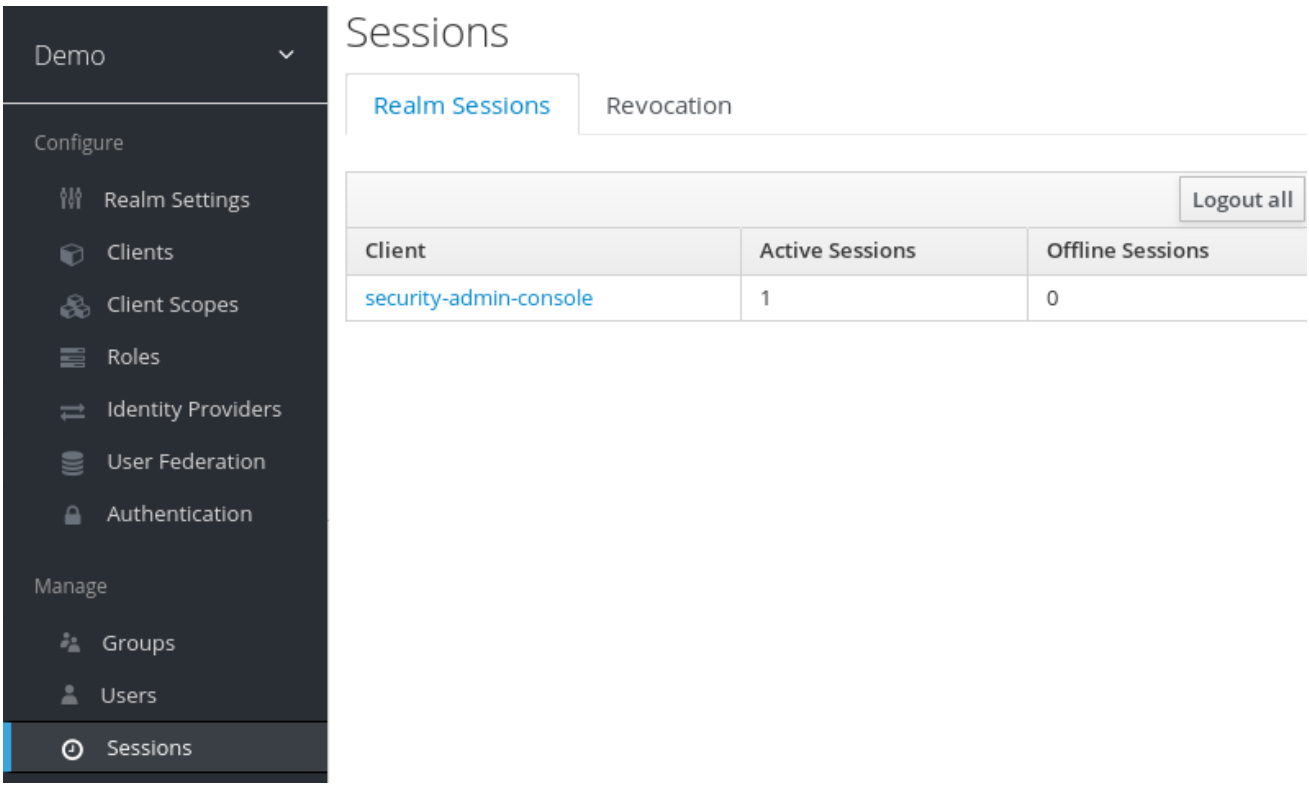
当用户登录到域时，Red Hat Single Sign-On 会为每个用户维护一个用户会话，并记住用户在会话中访问的每个客户端。realm 管理员可以在每个用户会话上执行多个操作：

- 查看域的登录统计信息。
- 查看活动用户及其登录位置。
- 从其会话注销用户。
- 撤销令牌。
- 设置令牌超时。
- 设置会话超时。

6.1. 管理会话

要查看 Red Hat Single Sign-On 中活动客户端和会话的顶级视图，请单击菜单中的 **Sessions**。

会话



6.1.1. Logout all Operation

您可以通过点击 **Logout all** 按钮注销域中的所有用户。

当您单击 **Logout all** 按钮时，所有 SSO Cookie 将变为无效，并在活跃浏览器会话中请求身份验证的客户端必须再次登录。Red Hat Single Sign-On 使用 Red Hat Single Sign-On OIDC 客户端适配器通知客户端。SAML 等客户端类型不会接收后端注销请求。

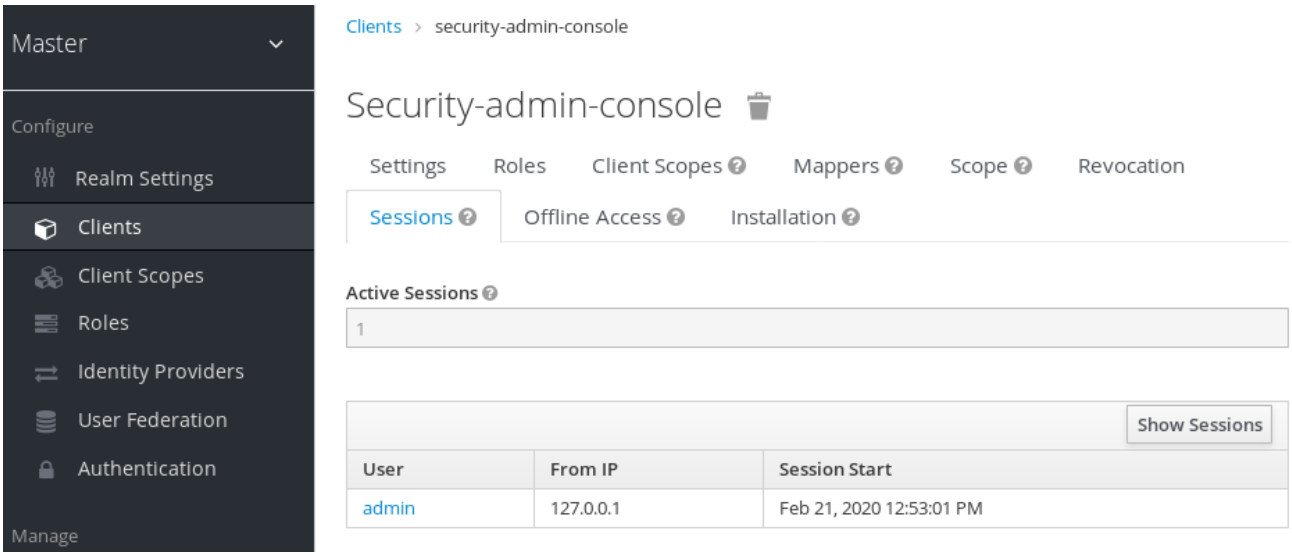
注意

点击 **Logout all** 不会撤销未处理的访问令牌。未完成的令牌必须自然而然。对于使用 Red Hat Single Sign-On OIDC 客户端适配器的客户端，您可以推送 撤销策略来 撤销令牌，但这不适用于其他适配器。

6.1.2. 应用程序导航

在 **Sessions** 页面上，您可以点击每个客户端来进入该客户端 **的会话** 选项卡。点 **Show Sessions** 按钮查看应用程序中的用户。

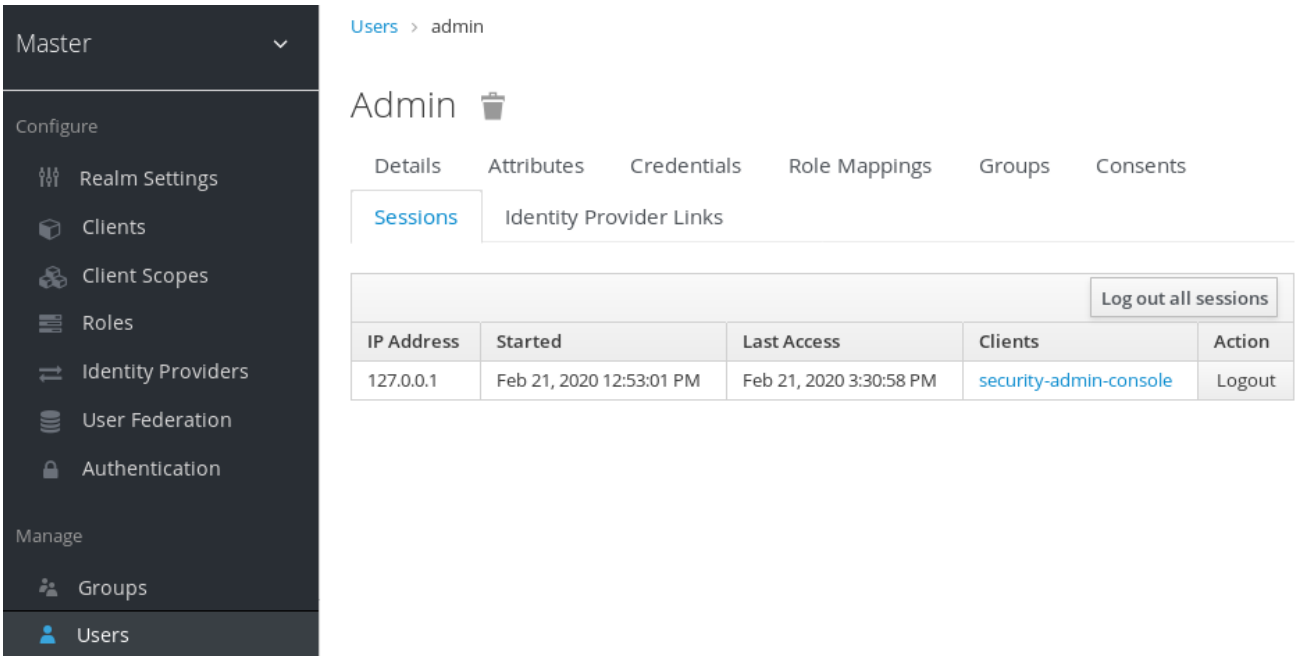
应用程序会话



6.1.3. 用户导航

如果进入单个用户的 **Sessions** 选项卡，您也可以查看用户的会话信息。

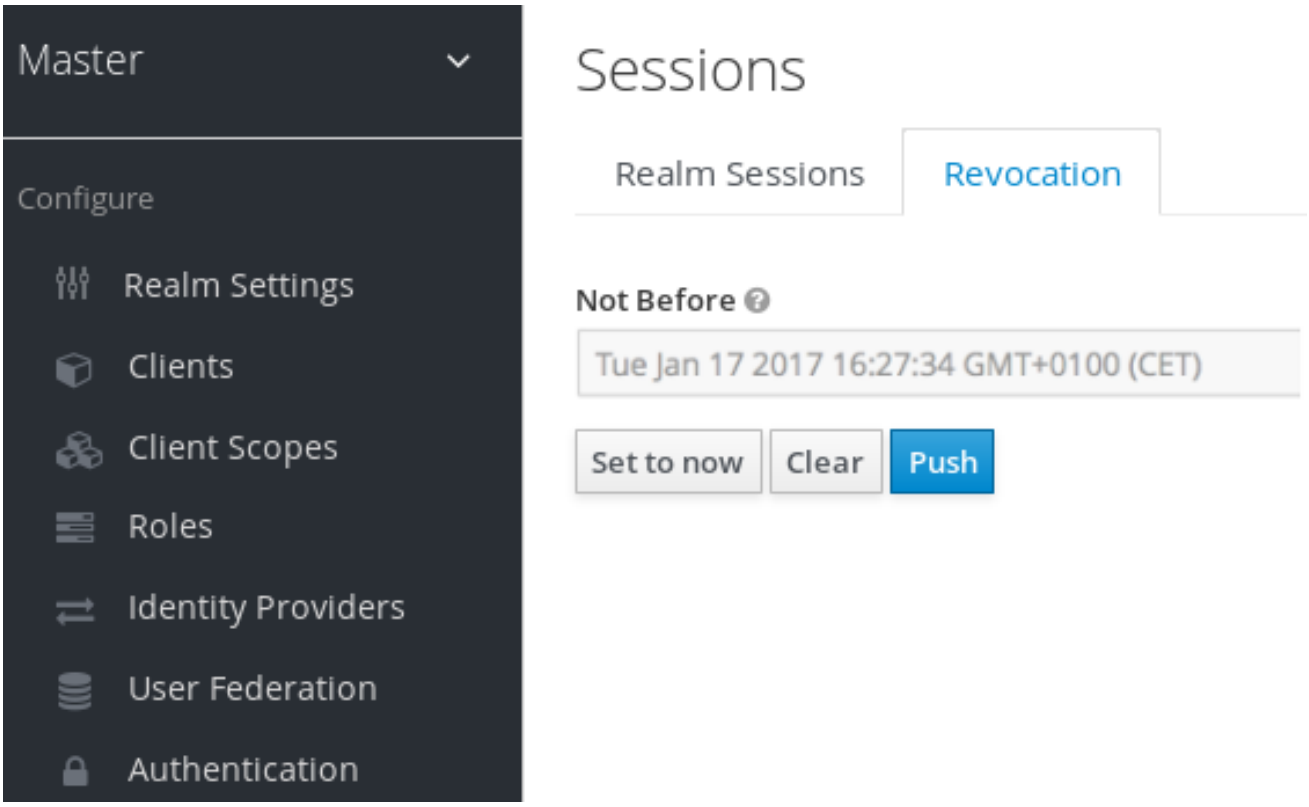
用户会话



6.2. 撤销策略

如果您的系统被破坏，您可以点击 **Sessions** 屏幕 **Revocation** 选项卡来撤销所有活跃会话和访问令牌。

撤销

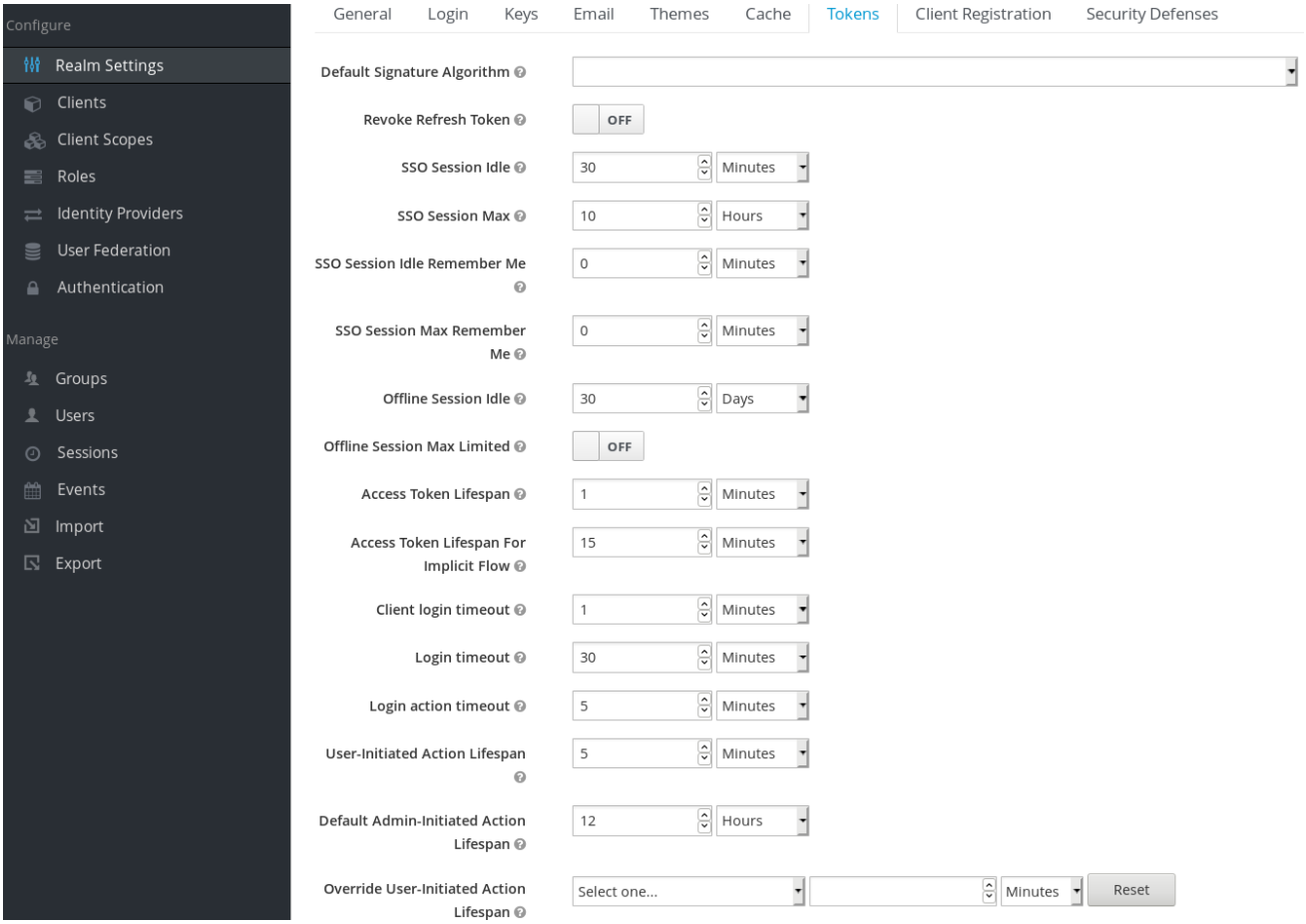


您可以指定在时间和日期之前发布的会话或令牌的时间和日期，使用此控制台无效。点 **Set to now** 将策略设置为当前时间和日期。点击 **Push** 使用 Red Hat Single Sign-On OIDC 客户端适配器将此撤销策略推送到任何注册的 OIDC 客户端中。

6.3. 会话和令牌超时

Red Hat Single Sign-On 通过 **Realm Settings** 菜单中的 **Tokens** 选项卡包含会话、cookie 和令牌超时的控制。

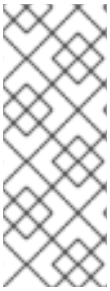
令牌选项卡



Configuration	描述
默认签名算法	用于为域分配令牌的默认算法。
撤销刷新令牌	**ON** 时，Red Hat Single Sign-On 会撤销刷新令牌，并发出客户端必须使用的另一个令牌。此操作适用于执行刷新令牌流的 OIDC 客户端。
SSO 会话 Idle	这个设置只适用于 OIDC 客户端。如果用户不活跃的时间超过这个超时时间，则用户会话会被无效。此超时值会在客户端请求身份验证或发送刷新令牌请求时重置。Red Hat Single Sign-On 在会话无效生效前为空闲超时添加时间窗口。请参见本节稍后 的备注。
SSO Session Max	用户会话过期前的最长时间。
SSO 会话 Idle Remember Me	此设置与标准 SSO 会话 Idle 配置类似，但特定于在启用了 **Remember Me** 的情况下进行登录。当用户在登录时点 **Remember Me** 时，可以指定较长的会话闲置超时。此设置是一种可选配置，如果其值不大于零，它会使用与 SSO 会话 Idle 配置相同的空闲超时。

Configuration	描述
SSO Session Max Remember Me	此设置与标准的 SSO Session Max 类似，但特定于 **Remember Me** 登录。当用户在登录时点击 **Remember Me** 时可以指定更长的会话。此设置是一种可选配置，如果其值不大于零，它会使用相同的会话寿命与 SSO Session Max 配置。
离线会话操作	此设置用于 离线访问。在 Red Hat Single Sign-On 撤销其离线令牌前，会话仍然闲置的时间。Red Hat Single Sign-On 在会话无效生效前为空闲超时添加时间窗口。请参见本节稍后 的备注。
离线会话 Max Limited	此设置用于 离线访问。如果这个标记是 **ON**，则 Offline Session Max 都可以控制离线令牌保持活跃时间，无论用户活动是什么。如果标志为 **OFF**，则离线会话永远不会由 lifespan 过期；这些会话仅通过闲置而过期。激活此选项后，您可以配置 Offline Session Max （域级别上的global 选项）和 **Client Offline Session Max** (**高级设置** 选项卡中的特定客户端级别选项)。
离线会话最大	此设置用于 离线访问，它是 Red Hat Single Sign-On 吊销对应的离线令牌前的最长时间。这个选项控制离线令牌保持活跃时间的最长时间，而不考虑用户活动。
客户端离线会话 Idle	此设置用于 离线访问。如果一个用户不活跃的时间超过这个超时时间，则离线令牌请求禁止闲置超时。此设置指定离线令牌的闲置超时与离线会话闲置超时。用户可以为每个客户端覆盖此设置。此设置是一个可选配置，当设置为零时，在离线会话 Idle 配置中使用相同的闲置超时。
客户端离线会话最大	此设置用于 离线访问。离线令牌过期和失效前的最长时间。此设置指定了一个比离线会话超时较短的令牌超时，但用户可针对单个客户端覆盖它。此设置是一个可选配置，当设置为零时，在 Offline Session Max 配置中使用相同的闲置超时。

Configuration	描述
客户端会话识别	客户端会话的闲置超时。如果用户不活跃的时间超过这个超时时间，客户端会话将无效，刷新令牌请求会禁止闲置超时。此设置永远不会影响一般的 SSO 用户会话，这是唯一的。注意 SSO 用户会话是零个或多个客户端会话的父项。为用户登录的每个客户端应用程序创建一个客户端会话。这个值应该指定比 **SSO Session Idle** 更短的空闲超时。用户可以在 **Advanced Settings** client 选项卡中覆盖单个客户端。此设置是一个可选配置，当设置为零时，在 SSO 会话 Idle 配置中使用相同的闲置超时。
客户端会话最大	客户端会话以及刷新令牌过期和无效前的最长时间。与上一个选项中所示，此设置永远不会影响 SSO 用户会话，并且应指定比 **SSO Session Max** 更短的值。用户可以在 **Advanced Settings** client 选项卡中覆盖单个客户端。此设置是一个可选配置，当设为零时，在 SSO Session Max 配置中使用相同的最大超时。
访问令牌 Lifespan	当 Red Hat Single Sign-On 创建 OIDC 访问令牌时，这个值会控制令牌的生命周期。
访问令牌 Lifespan 用于 Implicit Flow	使用 Implicit Flow，Red Hat Single Sign-On 没有提供刷新令牌。对于由 Implicit Flow 创建的访问令牌，存在单独的超时。
客户端登录超时	客户端必须在 OIDC 中完成授权代码流的最长时间。
登录超时	登录总时间必须花费。如果身份验证所需的时间超过此时间，用户必须再次启动身份验证过程。
登录操作超时	在验证过程中，最大时间用户可以安置在任意一个页面上。
用户初始化的操作 Lifespan	用户操作权限过期前的最长时间。保持这个值的短，因为用户通常会快速响应自创建的操作。
默认 Admin-Initiated Action Lifespan	管理员到期操作权限前的最大时间。保留这个值较长，以便管理员将电子邮件发送到离线用户。在发出令牌前，管理员可以覆盖默认超时。
覆盖用户初始化的操作 Lifespan	为每个操作指定独立的超时（例如，电子邮件验证、忘记密码、用户操作和身份提供者验证）。这个值默认为在 *User-Initiated Action Lifespan* 上配置的值。



注意

对于闲置超时，会话处于活跃状态的两分钟时间窗存在。例如，当您将超时设置为 30 分钟时，会话过期前将为 32 分钟。

对于一些情况下，集群和跨数据中心环境中需要此操作，其中令牌会在一个集群节点短时间内刷新，而其他集群节点会错误地将会话视为过期，因为它们还没有从刷新节点收到有关成功刷新节点的信息。

6.4. 离线访问

在 离线访问 登录过程中，客户端应用程序会请求离线令牌，而不是刷新令牌。客户端应用保存了这个离线令牌，并可在用户注销时使用它来登录。如果您的应用程序需要代表用户执行离线操作，则此操作很有用。例如，常规数据备份。

客户端应用负责保留存储中的离线令牌，然后使用它从 Red Hat Single Sign-On 服务器检索新的访问令牌。

刷新令牌和离线令牌之间的区别是离线令牌永远不会过期，且不受 **SSO Session Idle** 超时和 **SSO Session Max** lifespan 的影响。在用户注销或服务器重新启动后，离线令牌有效。您必须每 30 天或 Offline Session Idle 指定的天数使用离线令牌进行刷新令牌操作。

如果启用了 Offline Session Max Limited，离线令牌会在 60 天后过期，即使您使用离线令牌操作。您可以在管理门户中更改此值 Offline Session Max。

使用离线访问时，客户端闲置和最大超时可在 客户端级别 覆盖。在 client **Advanced Settings** 选项卡中，选项 **Client Offline Session Idle** 和 **Client Offline Session Max** 可让您为特定应用程序有较短的离线超时。请注意，客户端会话值也控制刷新令牌过期，但它们永远不会影响全局离线用户 SSO 会话。只有在 realm 级别中 Offline Session Max Limited 被设置为 **Enabled** 时，选项 **Client Offline Session Max** 才会被评估。

如果启用 Revoke Refresh Token 选项，则只能使用每个离线令牌。刷新后，您必须从刷新响应存储新的离线令牌，而不是上一个令牌。

用户可以查看和撤销红帽单点登录在 用户帐户控制台中授予的离线令牌。管理员可以在 **Consents** 选项卡中为单独的用户撤销离线令牌。管理员可以查看各个客户端"离线 **访问"选项卡中发布的所有离线** 令牌。管理员可以通过设置撤销 策略来撤销离线令牌。

若要发出离线令牌，用户必须拥有域级 **offline_access** 角色的角色映射。客户端也必须在其范围内拥有该角色。客户端必须添加 **offline_access** 客户端范围作为 **可选客户端范围** 到角色，该范围是默认完成的。

在向红帽单点登录发送授权请求时，客户端可以通过添加参数 **scope=offline_access** 来请求离线令牌。当您用来访问应用程序的安全 URL （如 http://localhost:8080/customer-portal/secured?scope=offline_access）时，Red Hat Single Sign-On OIDC 客户端适配器会自动添加这个参数。如果您在身份验证请求正文中包含 **scope=offline_access**，则 Direct Access Grant 和 Service Account 支持离线令牌。

6.5. 离线会话预加载

除了 Infinispan 缓存外，离线会话存储在数据库中，这意味着即使在服务器重启后也会可用。默认情况下，在服务器启动过程中，离线会话不会从数据库加载到 Infinispan 缓存中，因为如果很多离线会话被预加载，则这种方法存在缺陷。它可能会在服务器启动时间显著降低。因此，离线会话默认从数据库获取。

但是，Red Hat Single Sign-On 可以配置为在服务器启动期间将离线会话从数据库加载到 Infinispan 缓存。它可通过将 **userSessions** SPI 中的 **preloadOfflineSessionsFromDatabase** 属性设置为 **true** 来实现。

以下示例演示了如何配置离线会话预加载。

```
<subsystem xmlns="urn:jboss:domain:keycloak-server:1.2">
    ...
    <spi name="userSessions">
        <default-provider>infinispan</default-provider>
        <provider name="infinispan" enabled="true">
            <properties>
                <property name="preloadOfflineSessionsFromDatabase" value="true"/>
            </properties>
        </provider>
    </spi>
    ...
</subsystem>
```

使用 CLI 命令进行等同的配置：

```
/subsystem=keycloak-server/spi=userSessions:add(default-provider=infinispan)
/subsystem=keycloak-server/spi=userSessions/provider=infinispan:add(properties=
{preloadOfflineSessionsFromDatabase => "true"},enabled=true)
```

6.6. 临时会话

您可以在 Red Hat Single Sign-On 中进行临时会话。在使用临时会话时，Red Hat Single Sign-On 在成功身份验证后不会创建用户会话。Red Hat Single Sign-On 为成功验证用户的当前请求的范围创建一个临时的临时会话。Red Hat Single Sign-On 可以在身份验证后使用临时会话运行 协议 映射程序。

在临时会话中，客户端应用无法刷新令牌、内省令牌或验证特定的会话。有时这些操作是不必要的，因此您可以避免额外的资源来保留用户会话。此会话可节省性能、内存和网络通信（在集群和跨数据中心环境中）资源。

第 7 章 使用角色和组群分配权限

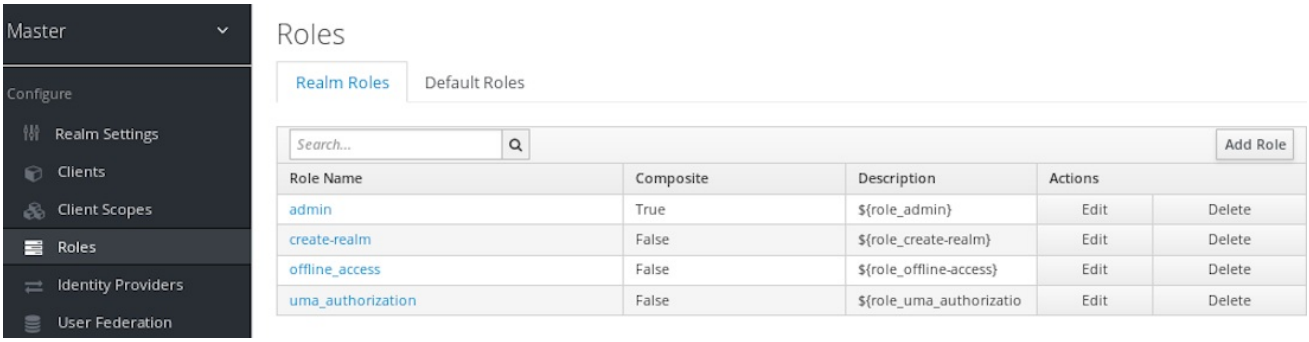
角色和组具有类似目的，即授予用户使用应用程序的访问权限和权限。组(group)是应用角色和属性的用户的集合。角色定义特定的应用权限和访问控制。

角色通常应用到一种用户。例如，一个机构可能会包括 **admin**, **user**, **manager**, 和 **employee** 角色。应用可以为角色分配访问权限和权限，然后将多个用户分配到该角色，以便用户拥有相同的访问权限和权限。例如，Admin Console 具有授予用户访问管理控制台不同部分的权限。

角色有一个全局命名空间，每个客户端也都有自己的专用命名空间，其中可以定义角色。

7.1. 创建域角色

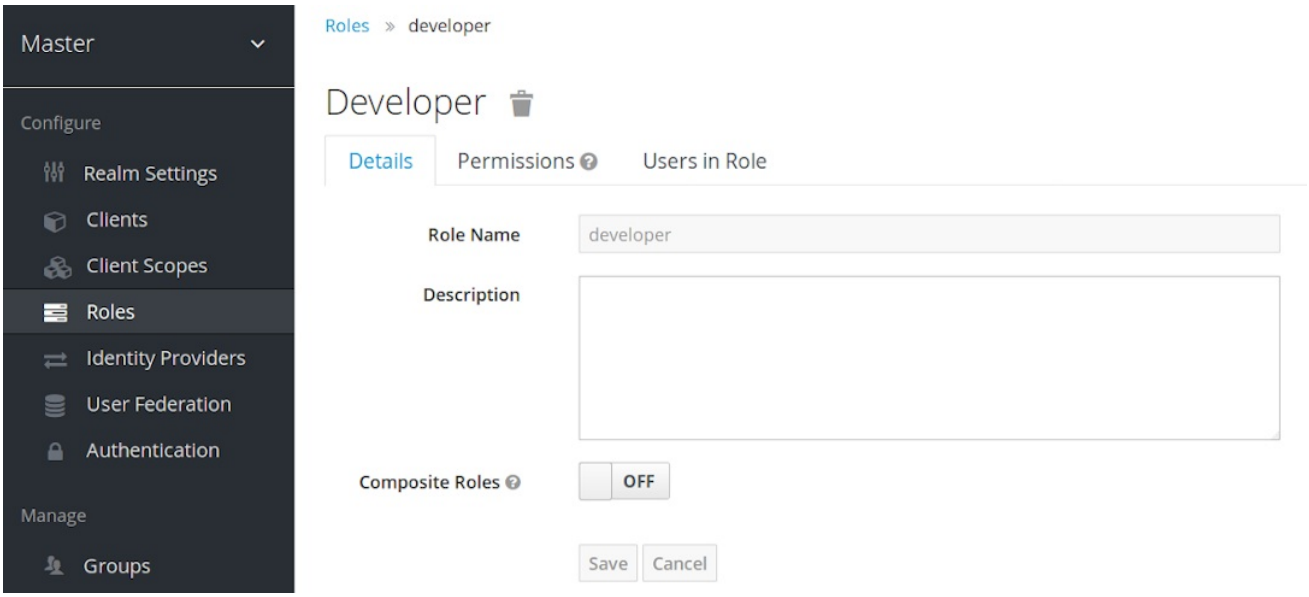
realm-level 角色是用于定义您的角色的命名空间。若要查看角色列表，可单击菜单中的 **Roles**。



流程

1. 单击 **Add Role**。
2. 输入 **角色名称**。
3. 输入 **描述**。
4. 点 **Save**。

添加角色



description 字段可以通过使用 **${var-name}** 字符串指定替换变量来本地识别。localized 值被配置为您的主题。详情请查看服务器开发人员指南。

7.2. 客户端角色

客户端角色是专用于客户端的命名空间。每个客户端都有自己的命名空间。客户端角色在各个客户端的 **Roles** 选项卡下进行管理。与这个 UI 交互的方式与域级别角色相同。

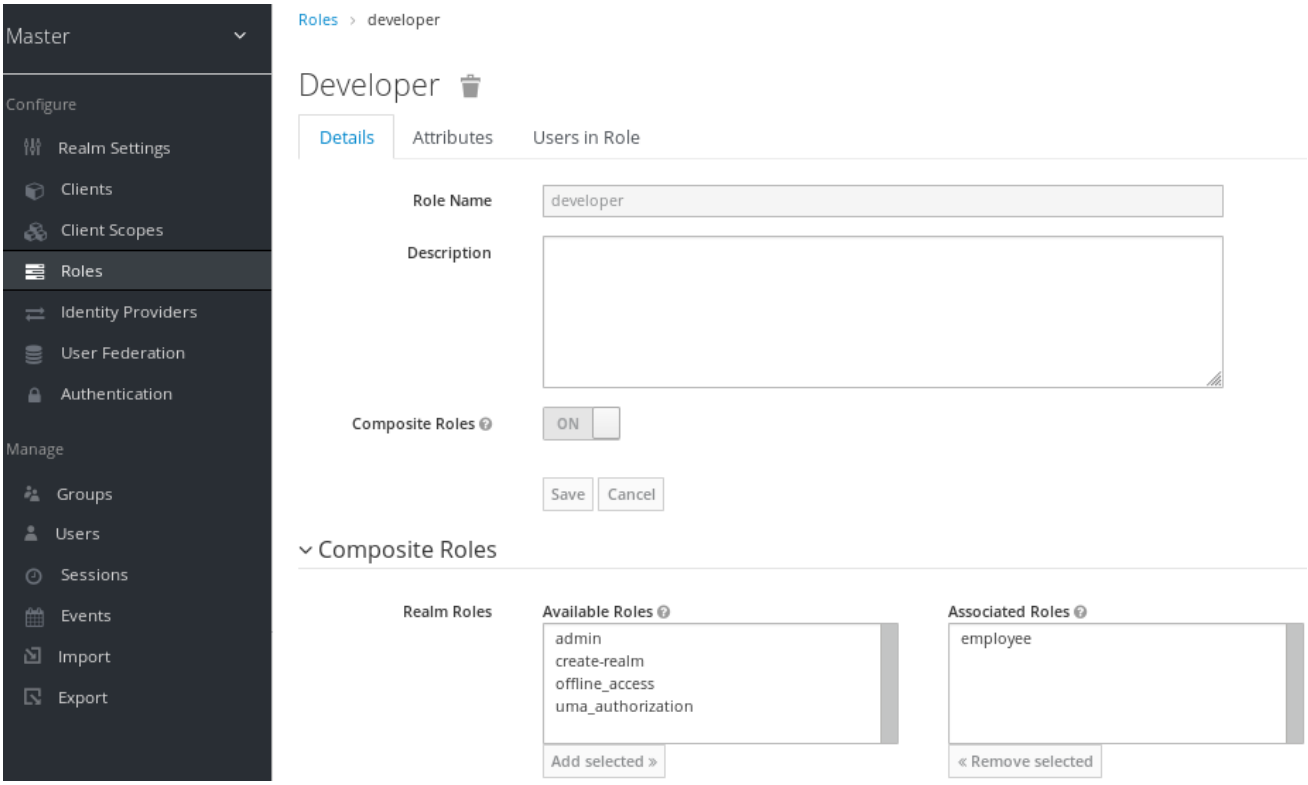
7.3. 将角色转换为复合角色

任何域或客户端级别角色都可以成为 *复合角色*。*复合角色* 是一个角色，它关联有一个或多个额外的角色。将复合角色映射到用户后，用户将获得与复合角色关联的角色。这个继承是递归的，因此用户也会继承任何复合的复合。但是，我们建议您使用复合角色。

流程

1. 在菜单中，单击 **Roles**。
2. 点击您要转换的角色。
3. 将 **Composite 角色** 切换到 **ON**。

复合角色



角色选择 UI 在页面中显示，您可以将 realm 级别和客户端级别角色与您要创建的复合角色相关联。

在本例中，**员工** 域级角色 **与开发人员** 复合角色关联。具有 **developer** 角色的任何用户也会继承 **员工** 角色。

注意

在创建令牌和 SAML 断言时，任何复合也将其关联角色添加到声明中，并断言发送到客户端的身份验证响应。

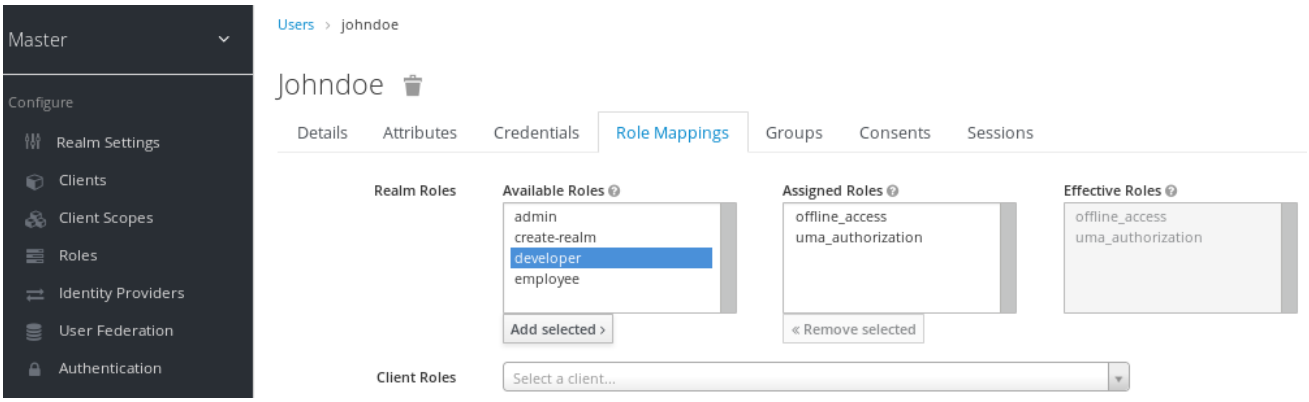
7.4. 分配角色映射

您可以通过该用户的角色映射选项卡 为**用户分配角色** 映射。

流程

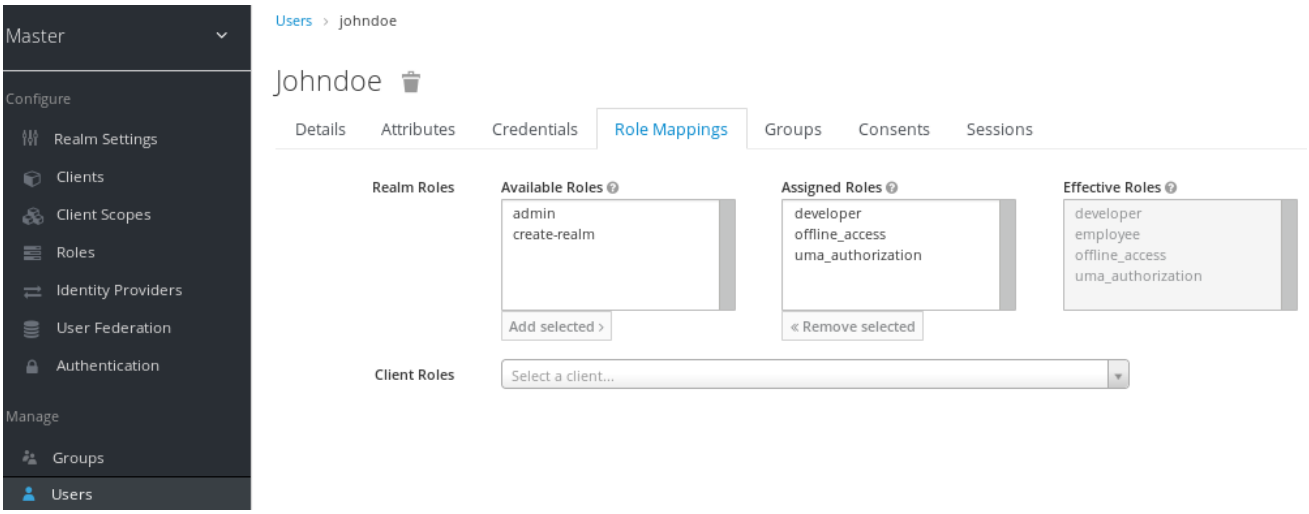
1. 点菜单中的 **Users**。
2. 点您要执行角色映射的用户。如果没有显示用户，请单击 **View all users**。
3. 点 **Role Mappings** 选项卡。
4. 在 **Available Roles** 框中，点您要分配给该用户的角色。
5. 点 **Add selected**。

角色映射



在前面的示例中，我们将复合角色 **开发人员** 分配给用户。该角色在 Composite Roles 主题中创建。

有效角色映射



分配了 **developer** 角色时，与 **developer** 复合关联的 **employee** 角色将在 **Effective Roles** 框中显示。**有效角色** 是明确分配给从复合继承的用户和角色的角色。

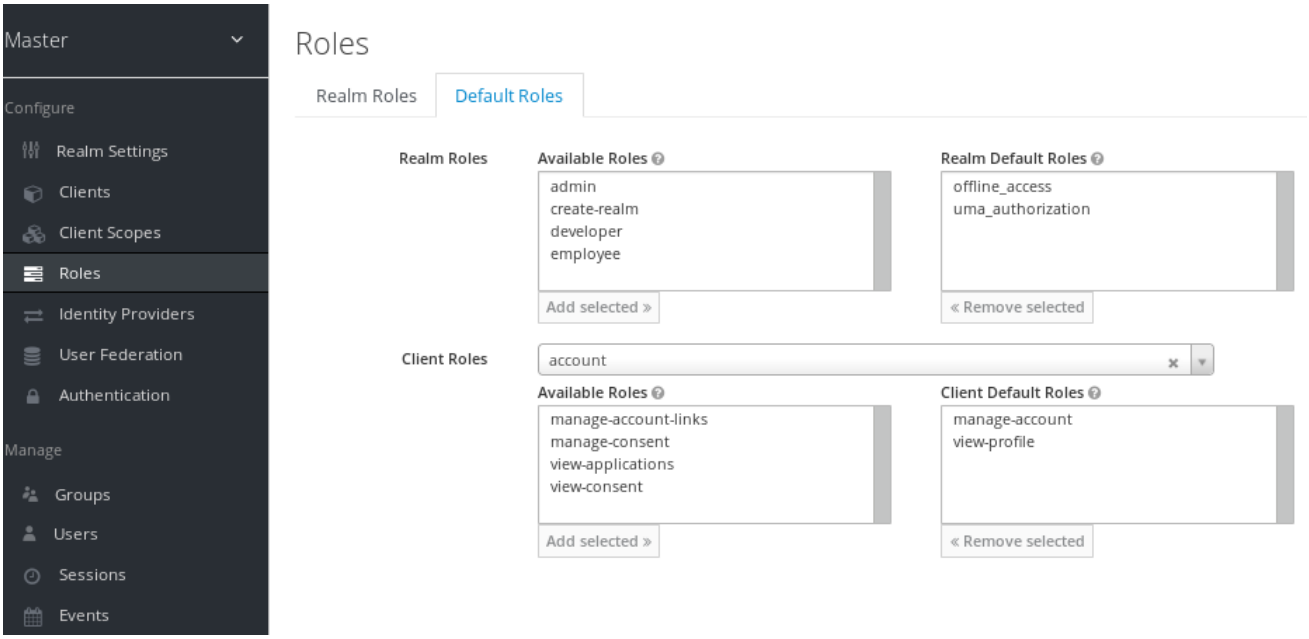
7.5. 使用默认角色

当用户通过 Identity Brokering 创建或导入用户时，使用默认角色映射。

流程

1. 在菜单中点击 **Roles**
2. 单击 **Default Roles** 选项卡。

默认角色



此屏幕截图显示一些 *默认角色* 已经存在。

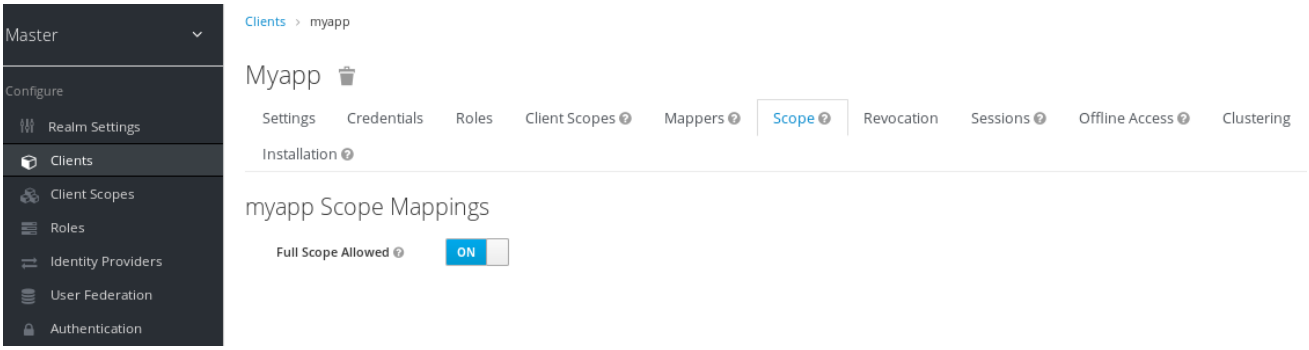
7.6. 角色范围映射

在创建 OIDC 访问令牌或 SAML 断言时，用户角色映射会成为令牌或断言中的声明。应用程序使用这些声明来做出对应用程序控制的资源的访问决策。红帽单点登录以数字化为访问令牌和应用签名，以重新使用它们调用远程保护的 REST 服务。但是，这些令牌存在相关的风险。攻击者可以获得这些令牌，并使用其权限破坏您的网络。要防止这种情况，请使用 *角色范围映射*。

角色 Scope Mappings 限制访问令牌中声明的角色。当客户端请求用户身份验证时，它们收到的访问令牌仅包含为客户端的范围明确指定的角色映射。其结果是，您可以限制每个单独的访问令牌的权限，而不是授予客户端访问所有用户的权限。

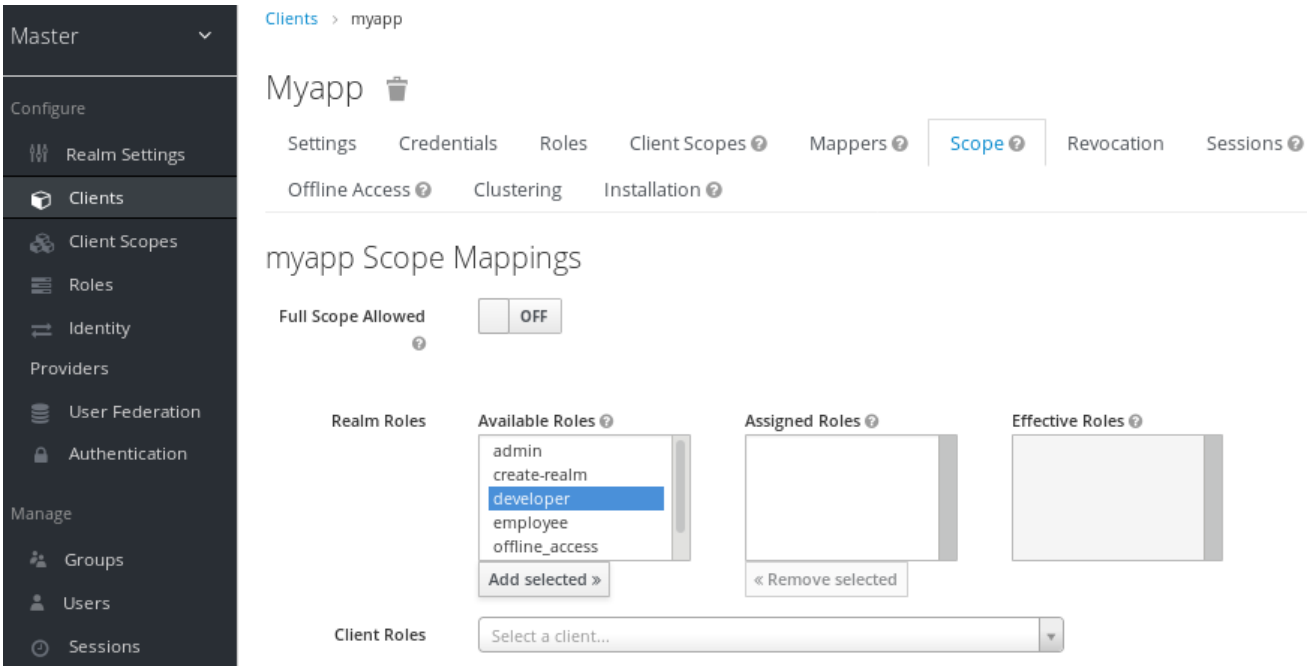
默认情况下，每个客户端获取用户的所有角色映射。您可以在每个客户端的 **Scope** 选项卡中查看角色映射。

完整范围



默认情况下，范围的有效角色是 realm 中每一个声明的角色。要更改此默认行为，将 **Full Scope Allowed** to **ON** 并声明每个客户端所需的特定角色。您还可以使用 客户端范围为 一组客户端定义相同的角色范围映射。

部分范围

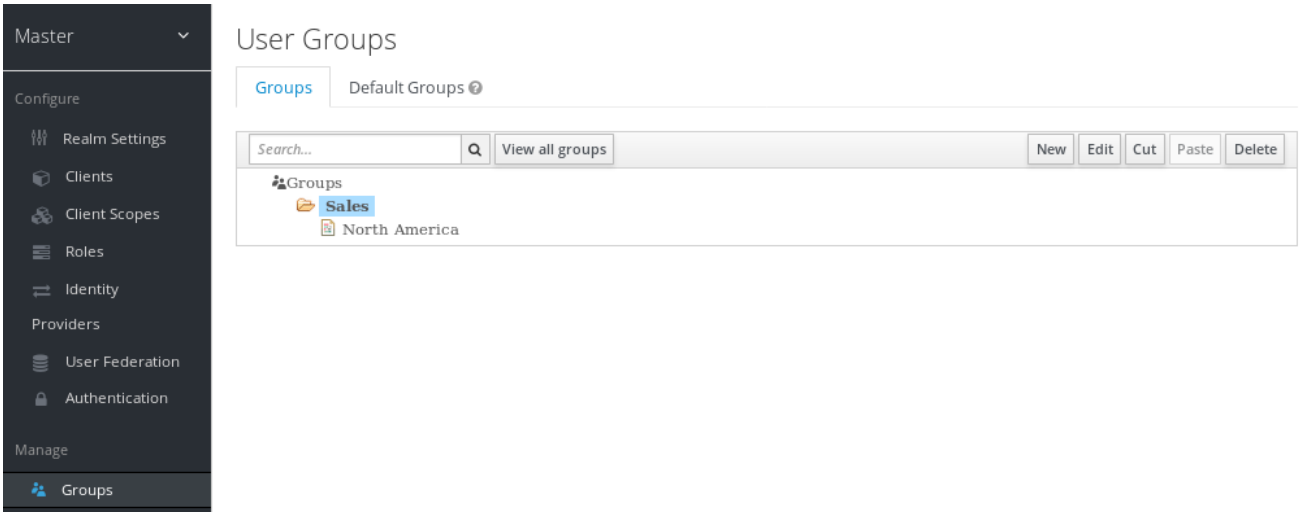


7.7. 组

Red Hat Single Sign-On 中的组为每个用户管理一组通用的属性和角色映射。用户可以是任意数量的组的成员，并继承分配给各个组的属性和角色映射。

要管理组，点菜单中的 **Groups**。

组



组是层级化。组可以有多个子组，但组只能有一个父组。子组从其父级继承属性和角色映射。用户也从其父级继承属性和角色映射。

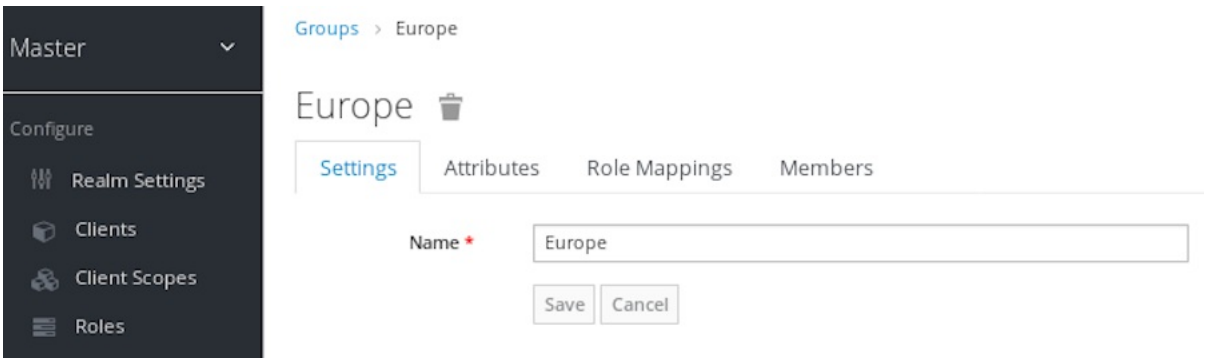
如果您有父组和子组，以及仅属于子组的用户，则子组中的用户会继承父组和子组的属性和角色映射。

以下示例包括顶级 **销售** 组和子 **北美** 子组。

添加组：

1. 点 组。
2. 单击 **New**。
3. 选择树中的 **组** 图标来生成顶级组。
4. 在 **Create Group** 屏幕中输入组名称。
5. 点 **Save**。
   此时会显示组管理页面。

   组

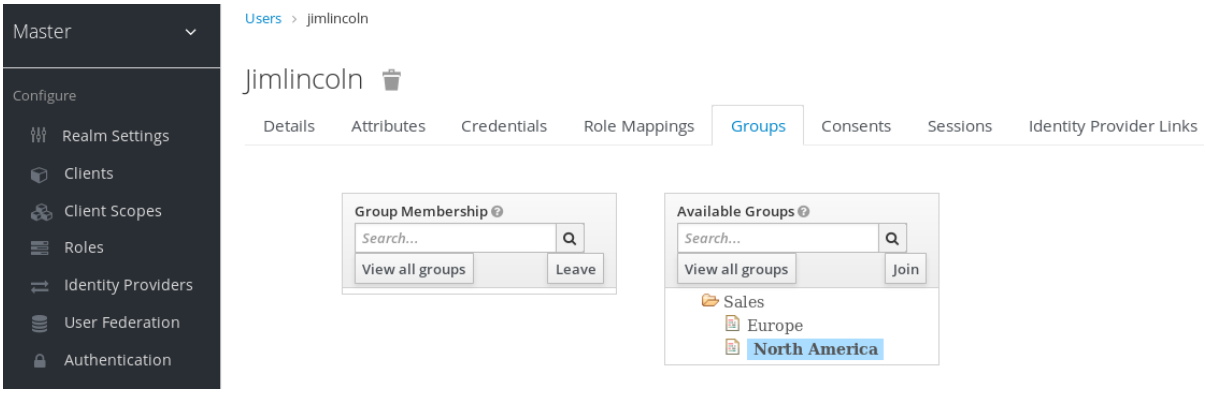


您定义的属性和角色映射由属于组成员的组和用户继承。

将用户添加到组中：

1. 点菜单中的 **Users**。
2. 点您要执行角色映射的用户。如果没有显示用户，请单击 **View all users**。
3. 点 **Groups**。

用户组



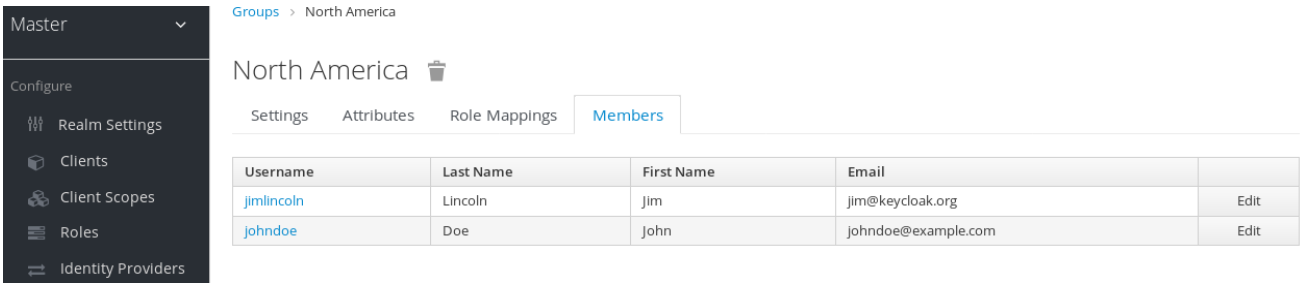
4. 从 **Available Groups** 树中选择一个组。
5. 点 **Join**。

从用户中删除组：

1. 从 **Group Membership** 树中选择组。
2. 点 **Leave**。

在这个示例中，用户 *jimlincoln* 位于 *北美* 组中。您可以看到 *jimlincoln* 在组的 **Members** 标签页中显示。

组成员资格



7.7.1. 与角色相比的组

组和角色具有一些相似性和差异。在 Red Hat Single Sign-On 中，组是应用角色和属性的用户的集合。角色定义用户和应用类型，为角色分配权限和访问控制。

复合角色 与组类似，因为它们提供相同的功能。它们之间的差别概念是概念。复合角色将权限模型应用到一组服务和应用。使用复合角色来管理应用程序和服务。

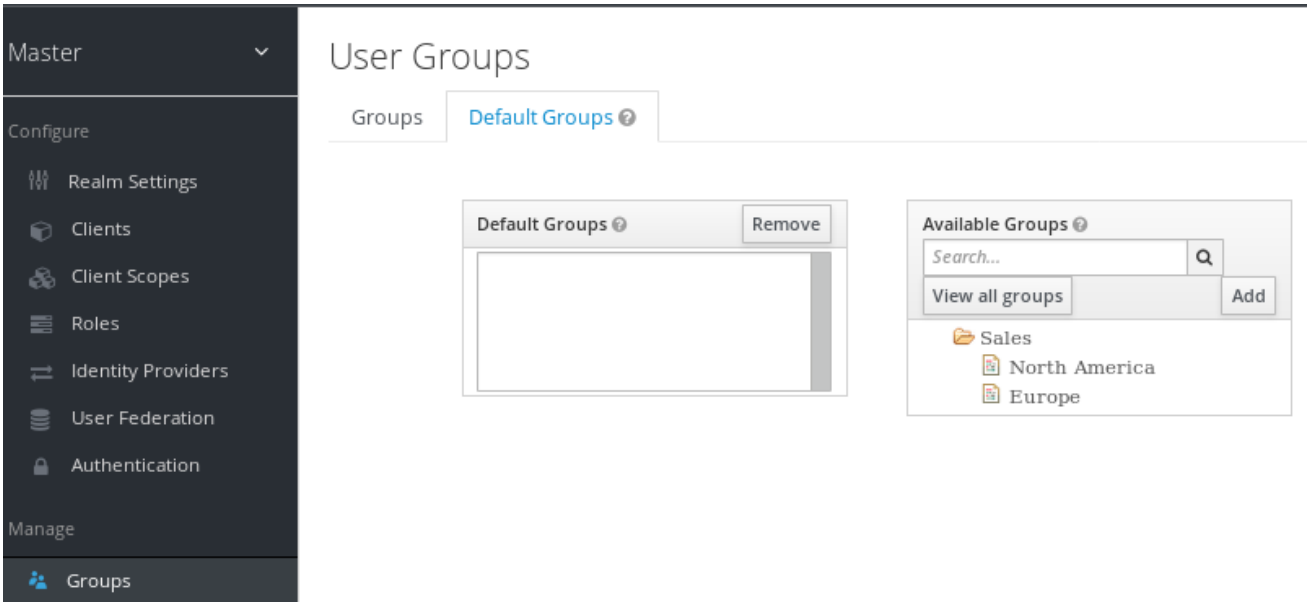
组专注于用户的集合及其在组织中的角色。使用组管理用户。

7.7.2. 使用默认组

要自动将组成员资格分配给创建或通过 Identity Brokering 导入的任何用户，您可以使用默认组。

1. 点菜单中的 **Groups**。
2. 单击 **Default Groups** 选项卡。

默认组



此截屏显示一些 *默认的组* 已存在。

第 8 章 配置身份验证

本章涵盖了几个身份验证主题。这些主题包括：

- 强制严格的密码和一次性密码(OTP)策略。
- 管理不同的凭据类型。
- 使用 Kerberos 登录.
- 禁用并启用内置凭证类型。

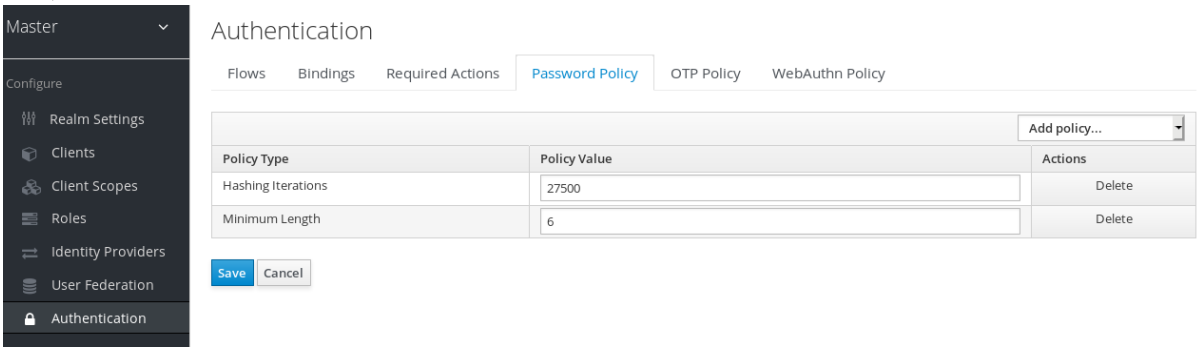
8.1. 密码策略

当 Red Hat Single Sign-On 创建域时，它不会将密码策略与域关联。您可以设置一个简单的密码，对长度、安全性或复杂性没有限制。在生产环境中无法接受简单的密码。Red Hat Single Sign-On 通过管理控制台提供一组密码策略。

流程

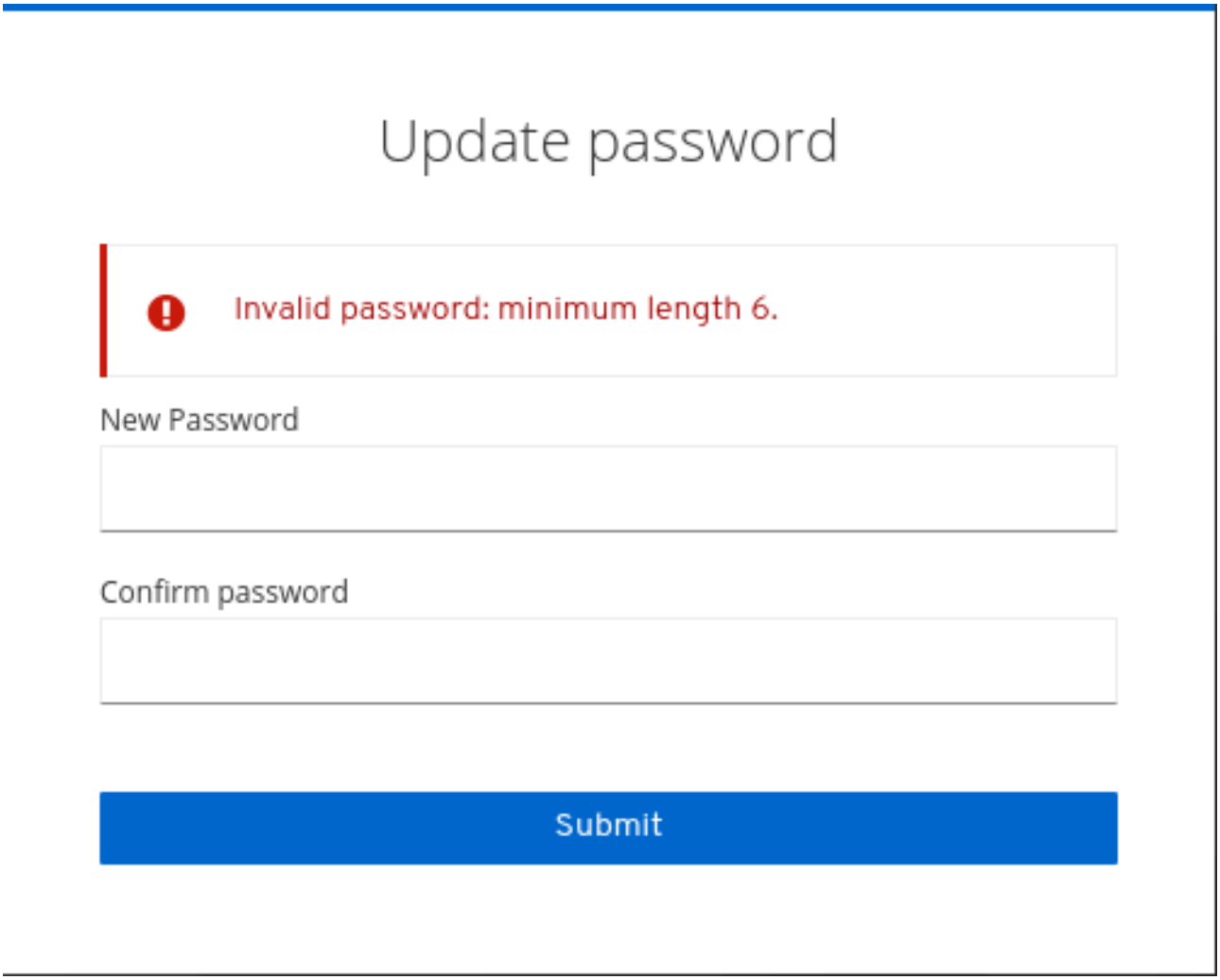
1. 在菜单中，单击 **Authentication**。
2. 点 **Password Policy** 选项卡。
3. 选择要在 **Add policy** 下拉列表中添加的策略。
4. 输入与所选策略对应的 **Policy Value** 的值。
5. 点 **Save**。

密码策略



保存策略后，Red Hat Single Sign-On 会强制执行新用户的策略，并为现有用户设置 Update Password 操作，以确保他们在下次登录时更改密码。例如：

失败的密码策略



8.1.1. 密码策略类型

8.1.1.1. 哈希算法

密码不会以明文形式存储。在存储或验证前，使用支持 PBKDF2、PBKDF2-SHA256 和 PBKDF-SHA512 哈希算法的标准哈希算法的红帽单点登录哈希密码。

8.1.1.2. 哈希迭代

指定存储或验证前 Red Hat Single Sign-On 哈希密码的次数。默认值为 27,500。

Red Hat Single Sign-On 哈希密码，以确保有访问密码数据库的恶意执行者无法通过反向工程读取密码。

注意

高哈希迭代值可能会影响性能，因为它需要较高的 CPU 电源。

8.1.1.3. 数字

密码字符串中所需的数字数字数。

8.1.1.4. 小写字符

密码字符串中所需的小写字母数量。

8.1.1.5. 大写字符

密码字符串中所需的大写字母数。

8.1.1.6. 特殊字符

密码字符串中所需的特殊字符数量。

8.1.1.7. Not username

密码不能与用户名相同。

8.1.1.8. 未通过电子邮件

密码不能与用户的电子邮件地址相同。

8.1.1.9. 正则表达式

密码必须与一个或多个定义的正则表达式模式匹配。

8.1.1.10. 过期密码

密码有效天数。当天数已过期时，用户必须更改其密码。

8.1.1.11. 最近使用

用户无法使用密码。红帽单点登录存储了使用的密码的历史记录。存储的旧密码数量可在 Red Hat Single Sign-On 中进行配置。

8.1.1.12. 密码黑名单

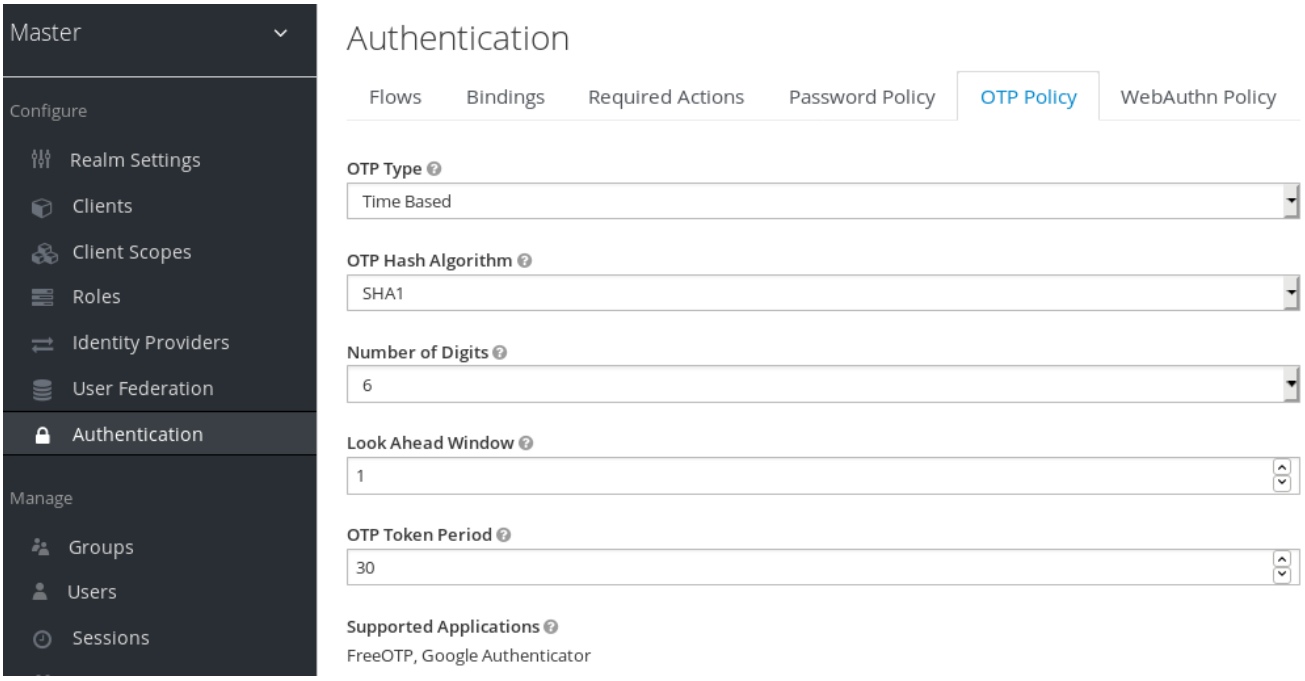
密码不能位于黑名单文件中。

- 将文件列入黑名单为 UTF-8 纯文本文件，以 Unix 行结尾。每行都代表列入黑名单的密码。
- Red Hat Single Sign-On 以不区分大小写的方式比较密码。黑名单中的所有密码都必须小写。
- blacklist 文件的内容必须是 blacklist 文件的名称。
- 将文件列入黑名单为 **${jboss.server.data.dir}/password-blacklists/**。使用以下方法自定义此路径：
  - **keycloak.password.blacklists.path** 属性。
  - **passwordBlacklist** 策略 SPI 配置的 **blacklistsPath** 属性。

8.2. 一个时间密码(OTP)策略

Red Hat Single Sign-On 具有多个策略来设置 FreeOTP 或 Google Authenticator One-Time 密码生成器。点 **Authentication** 菜单，点 **OTP Policy** 选项卡。

OTP 策略



Red Hat Single Sign-On 根据 OTP **策略** 选项卡中配置的信息在 OTP set-up 页面中生成一个 QR 代码。FreeOTP 和 Google Authenticator 在配置 OTP 时扫描 QR 代码。

8.2.1. 基于时间或基于计数器的一次性密码

适用于您的 OTP 生成器的 Red Hat Single Sign-On 中的算法基于时间和计数器。

使用基于时间的一次性密码(TOTP)，令牌生成器将显示当前的时间和共享 secret。服务器通过将时间内的哈希值与已提交的值进行比较来验证 OTP。TOTP 在较短的时间内有效。

使用基于计数器的一次性密码(HOTP)，Red Hat Single Sign-On 使用共享计数器而不是当前时间。Red Hat Single Sign-On 服务器在每次成功 OTP 登录时递增计数器。成功登录后有效的 OTP 更改。

TOTP 比 HOTP 更安全，因为匹配 OTP 在较短的时间内有效，而 HOTP 的 OTP 则有效一段时间。HOTP 比 TOTP 更友好，因为没有时间限制进入 OTP。

每次服务器递增计数器时，HOTP 需要数据库更新。在这个版本中，在负载过重时，在身份验证服务器上会排空性能。为提高效率，toTP 不会记住使用的密码，因此无需执行数据库更新。缺陷在于，可以在有效时间间隔中重新使用 TOTP。

8.2.2. TOTP 配置选项

8.2.2.1. OTP 哈希算法

默认算法是 SHA1。另一个更安全的选项是 SHA256 和 SHA512。

8.2.2.2. 数字数

OTP 的长度。短 OTP 是用户友好的，更容易键入，更容易记住。更长的 OTP 比 OTP 更安全。

8.2.2.3. 查看窗口

服务器尝试与哈希匹配的间隔数。如果 TOTP 生成器或身份验证服务器的时钟不同步，则此选项将出现在 Red Hat Single Sign-On 中。1 的默认值足够。例如，如果令牌的时间间隔是 30 秒，则默认值 1 表示它将在 90 秒窗口中接受有效令牌（时间间隔 30 秒 + 查看前 30 秒 + 查看后 30 秒）：这个值的每个递增都会

将有效窗口增加 60 秒（在 30 秒前加上 + 后面查找 30 秒）。

8.2.2.4. OTP 令牌周期

服务器与哈希匹配的时间间隔（以秒为单位）。每次通过间隔时，令牌生成器会生成一个 TOTP。

8.2.3. HOTP 配置选项

8.2.3.1. OTP 哈希算法

默认算法是 SHA1。另一个更安全的选项是 SHA256 和 SHA512。

8.2.3.2. 数字数

OTP 的长度。短 OTP 是用户友好的，更容易键入，更容易记住。比 OTP 更短的 OTP 时间越长。

8.2.3.3. 查看窗口

服务器尝试与哈希匹配的间隔数。如果 TOTP 生成器或身份验证服务器的时钟已过期，则 Red Hat Single Sign-On 会出现在 Red Hat Single Sign-On 中。1 的默认值足够。这个选项存在于 Red Hat Single Sign-On 中，以覆盖用户计数器何时进入服务器。

8.2.3.4. 初始计数器

初始计数器的值。

8.3. 身份验证流程

身份验证 流程是身份验证、屏幕和操作的容器，登录、注册和其他红帽单点登录工作流期间。要查看所有流、操作和检查，每个流程都需要：

流程

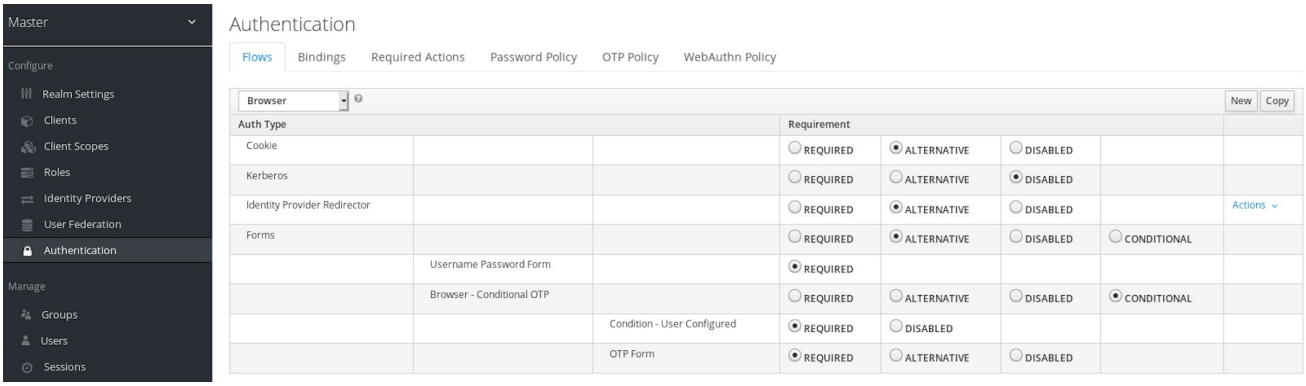
1. 在菜单中，单击 **Authentication**。
2. 点 **Flows** 选项卡。

8.3.1. 内置流

Red Hat Single Sign-On 有几个内置流。您无法修改这些流，但您可以更改流程的需求以满足您的需要。

在下拉列表中，选择 **浏览器** 以显示 Browser Flow 屏幕。

浏览器流



将鼠标悬停在下拉列表的 question-mark 工具上，以查看流的描述。有两个部分。

8.3.1.1. auth 类型

要执行的验证或操作的名称。如果验证缩进，则会在子流中。它可能或无法执行，具体取决于其父进程的行为。

1. cookie
   当用户第一次成功登录时，Red Hat Single Sign-On 会设置一个会话 Cookie。如果已经设置了 cookie，这个验证类型可以成功。由于 Cookie 提供程序返回成功，因此这一级别上的每个执行都是 *备选* 的，因此 Red Hat Single Sign-On 不会执行任何其他执行。这会导致登录成功。

2. Kerberos
   这个验证器默认被禁用，并在浏览器流中跳过。

3. 身份提供程序 Redirector
   此操作通过 **Actions** > **Config** 链接进行配置。它重定向到另一个 IdP 的 身份代理。

4. 表单
   由于此子流标记为 *替代*，因此如果传递 **Cookie** 验证类型，则不会执行。此子流包含需要执行的额外验证类型。Red Hat Single Sign-On 加载此子流的执行并处理它们。

第一个执行是 **Username Password Form**，它是一个呈现用户名和密码页面的身份验证类型。它已被标记为 *必需的*，因此用户必须输入有效的用户名和密码。

第二个执行是 **浏览器 - Conditional OTP**子流。这个子流是 *条件*，并根据 **Condition - User Configured** execution 的结果来执行。如果结果正确，Red Hat Single Sign-On 会加载此子流的执行并处理它们。

下一个执行是 **Condition - User Configured** authentication。此身份验证检查 Red Hat Single Sign-On 是否在用户流中配置了其他执行。**Browser - Conditional OTP**子流仅在用户配置了 OTP 凭证时才会执行。

最后的执行是 **OTP Form**。Red Hat Single Sign-On 会将这个执行标记为 *required*，但只有用户因为 *条件* 子流中的设置而设置 OTP 凭证时才会运行。如果没有，用户不会看到 OTP 表单。

8.3.1.2. 要求

一组控制操作执行的单选按钮。

8.3.1.2.1. 必需

流中的 所有必需元素都必须成功执行。如果需要的元素失败，流会终止。

8.3.1.2.2. 替代方案

只有单一元素必须成功执行流程才能成功评估成功。因为 *Required* 流元素足以将流标记为成功，所以包含 *Required* 流元素的 *Alternative* 流元素将无法执行。

8.3.1.2.3. Disabled

该元素不计将流标记为成功。

8.3.1.2.4. 条件

这个要求类型只在子流中设置。

- *Conditional* 子流包含执行。这些执行必须评估为逻辑语句。
- 如果所有执行都评估为 *true*，*则条件* 子流充当 *必需项*。
- 如果所有执行都评估为 *false*，Condition *al* sub-flow 充当 *Disabled*。
- 如果没有设置执行，*Conditional* 子流充当 *Disabled*。
- 如果流包含执行，且流没有设置为 *Conditional*，Red Hat Single Sign-On 不会评估执行，且执行被视为功能 *禁用*。

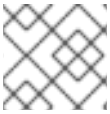
8.3.2. 创建流

在设计流程时，应用重要的功能和安全考虑因素。

要创建流，请执行以下操作：

流程

1. 在菜单中，单击 **Authentication**。
2. 单击 **New**。



注意

您可以复制并修改现有流。选择流，单击 **Copy**，然后为新流输入一个名称。

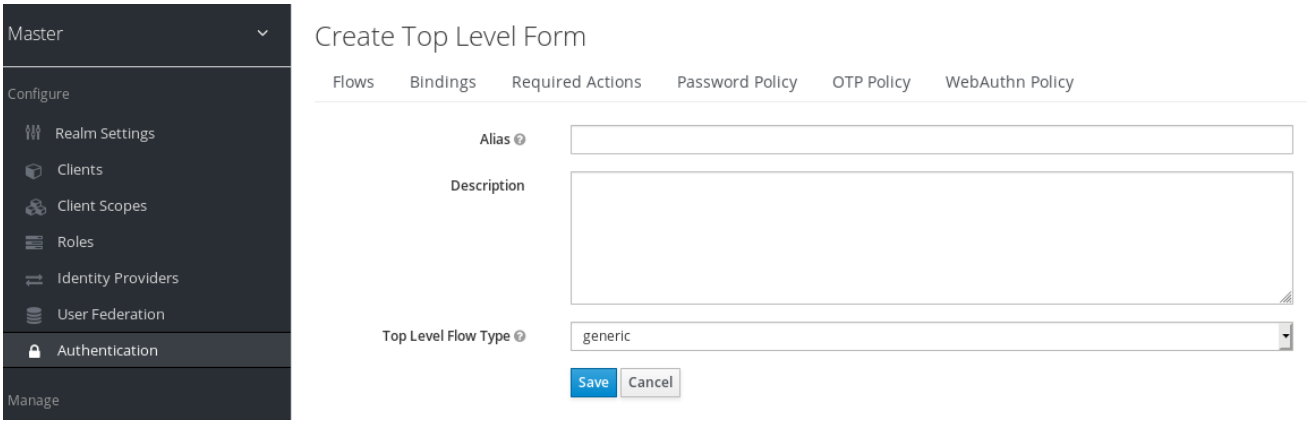
创建新流时，您必须首先使用以下选项创建一个顶级流：

Alias
流的名称。

描述
您可以设置为流的描述。

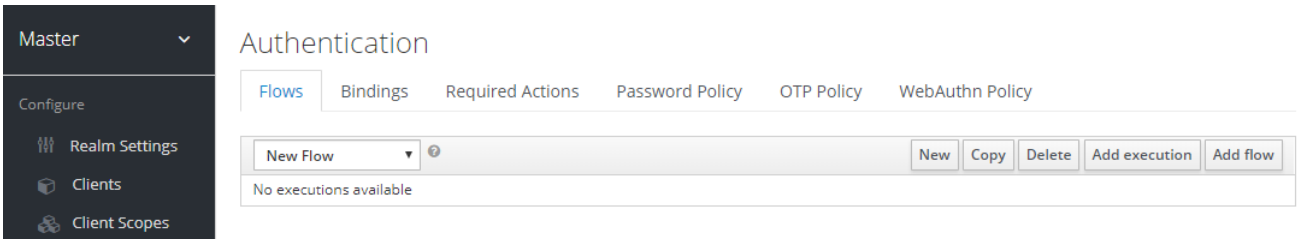
顶级流类型
流的类型。类型 **客户端** 仅用于客户端的身份验证（应用程序）。对于所有其他情况，请选择 **通用**。

创建顶级流



当 Red Hat Single Sign-On 创建流时，Red Hat Single Sign-On 会显示 **Delete**、**Add execution** 和 **Add flow** 按钮。

空新流

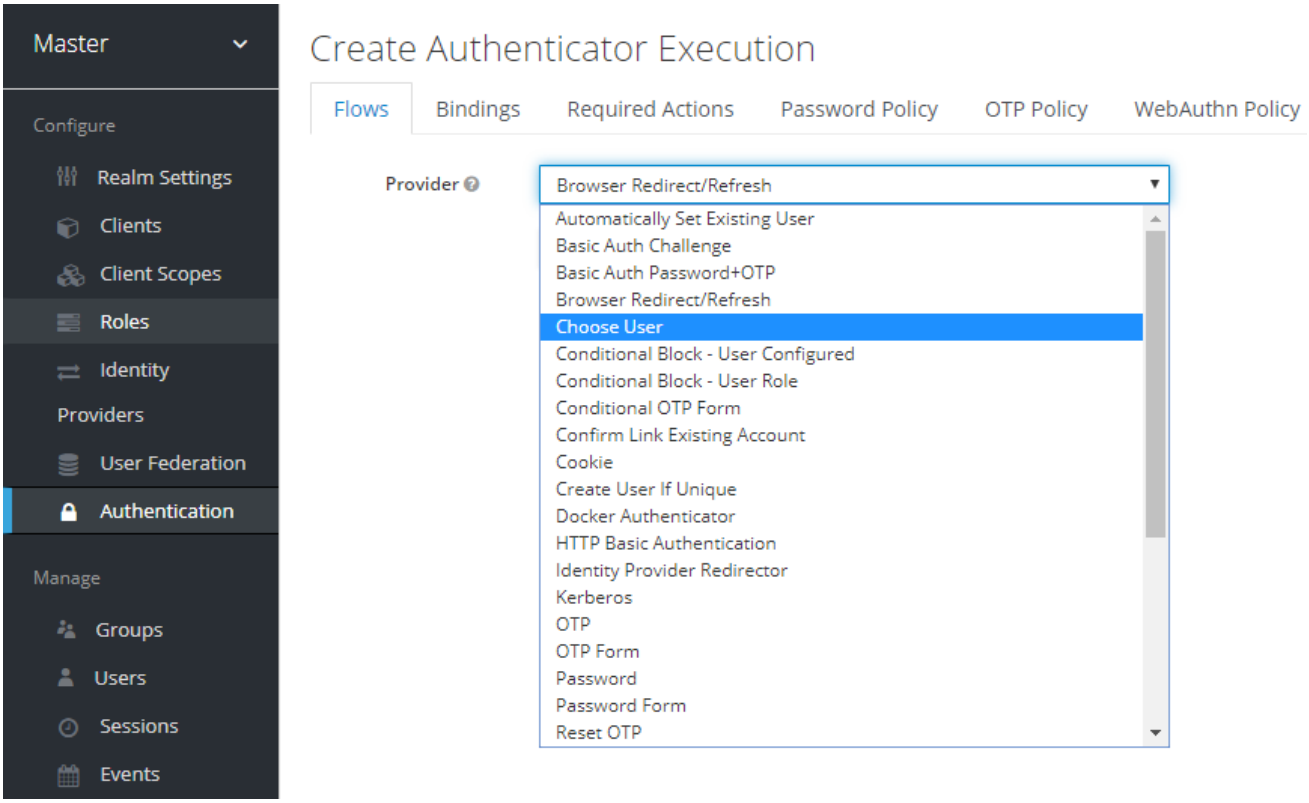


三个因素决定了流和子流的行为。

- 流和子流的结构。
- 流中的执行
- 在子流和执行过程中设置的要求。

执行操作有多种，从发送重置电子邮件以验证 OTP。使用 **Add execution** 按钮添加执行。将鼠标悬停在 **提供程序** 旁边的问号上，以查看执行的描述。

添加身份验证执行



有两种执行类型，*即自动执行 和交互式执行*。*Automatic executions* 与 **Cookie** 执行类似，并将在流中自动执行操作。*交互式执行*将暂停流以获取输入。执行操作成功将其状态设置为 *success*。对于完成流程，至少需要一个执行状态 *成功*。

您可以使用 **Add flow** 按钮将子流添加到顶级流中。**Add flow** 按钮显示 **Create Execution Flow** 页面。此页面与 **Create Top Level Form**页类似。区别在于，**Flow Type** 可以是 **通用** （默认）或 **表单**。**表单** 类型构造了为用户生成表单的子流，类似于内置的 **注册流程**。子流成功取决于它们的执行程度，包括它们的子流。如需了解子流的工作方式，请参阅执行 要求部分。

注意

添加执行后，检查要求具有正确的值。

流中的所有元素在 **Actions** 菜单中都有一个 **Delete** 选项。此操作会从流中删除这个元素。执行具有 **Config** 菜单选项来配置执行。还可以使用 **Add execution** and **Add flow** 菜单选项把执行和子流加入到子流。

由于执行顺序非常重要，因此您可以使用名称旁边的上和下个按钮在流内移动和执行和子流。

警告

当您配置身份验证流以确认您的设置中存在安全漏洞时，请确保正确测试您的配置。我们建议您测试各种类型情况。例如，考虑在身份验证前从用户帐户中删除不同凭证时测试用户的验证行为。

例如，当有双因素验证器（如 OTP Form 或 WebAuthn Authenticator）时，作为 REQUIRED 且用户没有特定类型的凭证，用户可以在验证自身期间设置特定的凭证。这种情况意味着，在身份验证过程中，用户不会通过这个凭证进行身份验证。要进行浏览器身份验证，请确保使用一些1因素凭证（如 Password 或 WebAuthnless Authenticator）配置您的身份验证流程。

8.3.3. 创建无密码浏览器登录流程

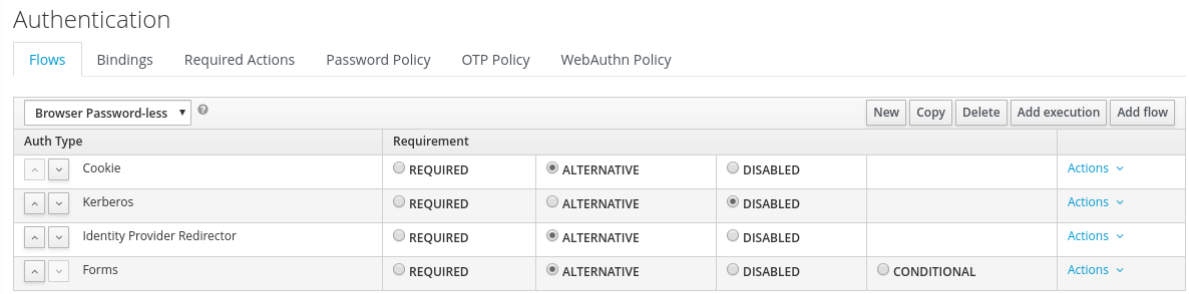
为了说明流程的创建，本节描述了创建高级浏览器登录流。此流程的目的是允许用户选择使用无密码的 WebAuthn 登录，或使用密码和 OTP 进行双因素身份验证。

流程

1. 在菜单中，单击 **Authentication**。
2. 点 **Flows** 选项卡。
3. 单击 **New**。
4. **输入浏览器密码** 作为别名。
5. 点 **Save**。
6. 单击 **Add execution**。
7. 从下拉列表中选择 **Cookie**。
8. 点 **Save**。
9. 点 **Alternative**，**Cookie** 验证类型将它的 requirement 设置为 alternative。
10. 单击 **Add execution**。
11. 从下拉列表中选择 **Kerberos**。
12. 单击 **Add execution**。
13. 从下拉列表中选择 **Identity Provider Redirector**。
14. 点 **Save**。
15. 点 **Alternative**，**Identity Provider Redirector**验证类型将它的 requirement 设置为 alternative。
16. 点 **Add flow**。

17. 输入 **Forms** 作为别名。
18. 点 **Save**。
19. 点 **Alternative**，**Forms** 验证类型将它的 requirement 设置为 alternative。

浏览器流的常用部分



20. 点 **Forms** execution 的 **Actions**。
21. 选择 **Add execute**。
22. 从下拉列表中选择 **Username Form**。
23. 点 **Save**。
24. 单击 **Username Form** authentication type 所需的 **Required**，以将其要求设置为 required。

在这个阶段，表单需要用户名，但没有密码。我们必须启用密码验证来避免安全风险。

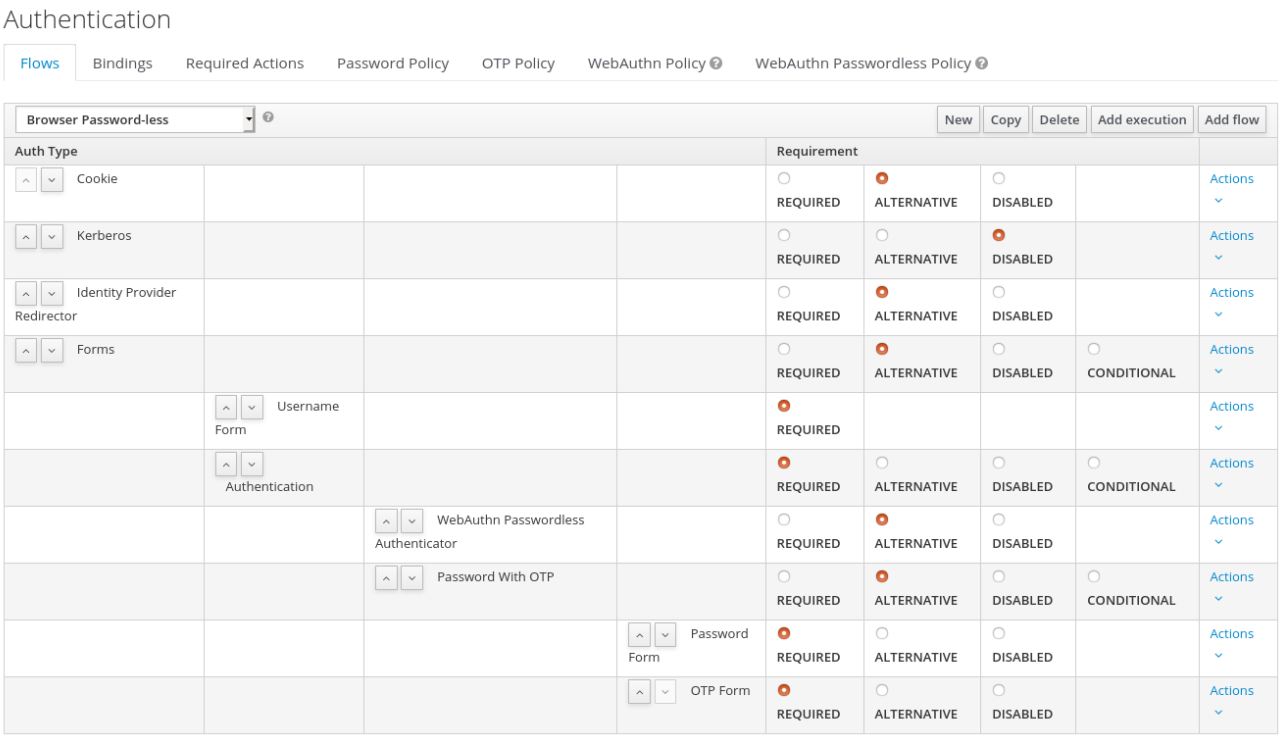
1. 点 **Forms** 子流的 **Actions**。
2. 点 **Add flow**。
3. 输入 **Authentication** 作为别名。
4. 点 **Save**。
5. 单击 **Authentication** 身份验证类型的必需，以设置其要求。
6. 点 **Authentication** 子流的 **Actions**。
7. 单击 **Add execution**。
8. 从下拉列表中选择 **Webauthn Passwordless Authenticator**。
9. 点 **Save**。
10. 点击 **Webauthn Passwordent** icator 身份验证类型的替代选择。
11. 点 **Authentication** 子流的 **Actions**。
12. 点 **Add flow**。
13. 输入 **Password with OTP** 作为一个别名。
14. 点 **Save**。

15. 点击 **OTP 身份验证类型 的替代方案**，将其要求设置为替代方案。
16. 点 **带有 OTP 子流的密码** 的 **Actions**。
17. 单击 **Add execution**。
18. 从下拉列表中选择 **Password Form**。
19. 点 **Save**。
20. 单击 **Password Form** authentication type **所需**，以将其要求设置为必填。
21. 点 **带有 OTP 子流的密码** 的 **Actions**。
22. 单击 **Add execution**。
23. 从下拉列表中选择 **OTP Form**。
24. 点 **Save**。
25. **点击 OTP Form** 身份验证类型所需的设置要求。

最后，更改绑定。

1. 单击 **Bindings** 选项卡。
2. 点击 **Browser Flow** 下拉列表。
3. 从下拉列表中选择 **Browser Password-less**。
4. 点 **Save**。

免密码浏览器登录



输入用户名后，流程如下：

如果用户记录了 WebAuthn无密码的凭证，他们可以使用这些凭证直接登录。这是无密码登录。用户还可以使用 **OTP** 选择 **Password**，因为 **WebAuthn Passwordless** execution 和 **使用 OTP 流的 Password** 设置为 **alternatives**。如果将其设定为 **Required**，则用户必须输入 WebAuthn、password 和 OTP。

如果用户选择带有 **WebAuthn passwordless** 验证的 **Try another way** 链接，用户可以在 **Password** 和 **Security Key** (WebAuthn passwordless) 间进行选择。选择密码时，用户需要继续使用分配的 OTP 登录。如果用户没有 WebAuthn 凭证，用户必须输入密码，然后 OTP。如果用户没有 OTP 凭证，则会要求您记录它。

注意

由于 WebAuthn 免密码执行设定为 **替代方案**，而不是 **必需的**，因此这个流不会要求用户注册 WebAuthn 凭据。用户若要具有 Webauthn 凭据，管理员必须向用户添加所需的操作。为此：

1. 在 realm 中启用 **Webauthn Register Passwordless** 必需操作（请参阅 WebAuthn 文档）。
2. 使用用户凭据管理菜单的 **Credential Reset** 部分设置必要的操作。

创建等高级流可能会产生副作用。例如，如果启用为用户重置密码的功能，可以通过密码表单访问此密码。在默认的 **Reset Credentials** 流中，用户必须输入其用户名。由于用户已在 **浏览器式密码** 流中之前输入了用户名，因此这个操作对于 Red Hat Single Sign-On 和 sub-optimal 对用户体验来说是不必要的。要解决这个问题，您可以：

- 复制 **重置凭据** 流。将其名称设置为 **"重置无密码"的凭据**，例如：
- 在 **Choose user** 执行的 **Actions** 菜单中选择 **Delete**。
- 在 **Bindings** 菜单中，将 reset credential 流从 **Reset Credentials** 更改为 **Reset Credentials forless**

8.3.4. 使用步骤机制创建浏览器登录流程

这部分论述了如何使用步骤机制创建高级浏览器登录流。步骤身份验证的目的是允许根据用户的特定身份验证级别访问客户端或资源。

流程

1. 在菜单中，单击 **Authentication**。
2. 点 **Flows** 选项卡。
3. 单击 **New**。
4. 输入 **Browser Incl Step up Mechanism** 作为别名。
5. 点 **Save**。
6. 单击 **Add execution**。
7. 从 item 列表中选择 **Cookie**。
8. 点 **Save**。
9. 点 **Alternative**，**Cookie** 验证类型将它的 requirement 设置为 alternative。

10. 点 **Add flow**。
11. 输入 **Auth Flow** 作为别名。
12. 点 **Save**。
13. 点 **Alternative**，**Auth Flow** 验证类型将它的 requirement 设置为 alternative。

现在，您可以为第一个身份验证级别配置网络流。

1. 为 **Auth Flow** 点 **Actions**。
2. 点 **Add flow**。
3. 输入 **1st Condition Flow** 作为别名。
4. 点 **Save**。
5. 为 **1st Condition Flow** 验证类型点击 **Conditional**，将其要求设置为条件。
6. 点 **1st Condition Flow** 的 **Actions**。
7. 单击 **Add execution**。
8. 从项列表中选择 **Conditional - Authentication Level**。
9. 点 **Save**。
10. 点 **Conditional - Level Of Authentication**authentication type 来设置其要求。
11. 点 **Authentication Conditional - Level 的Actions**。
12. 单击 **Config**。
13. 输入 **级别 1** 作为别名。
14. 为验证级别(LoA)输入 **1**。
15. 将 Max Age 设置为 **36000**。这个值以秒为单位，相当于 10 小时，这是在 realm 中设置的默认 **SSO Session Max** 超时。因此，当用户使用此级别进行身份验证时，后续的 SSO 登录可以重新使用此级别，用户不需要通过这个级别进行身份验证，直到用户会话结束前，默认为 10 小时。
16. 点 **Save**

为第一个身份验证级别配置条件

17. 点 **1st Condition Flow** 的 **Actions**。
18. 单击 **Add execution**。

19. 从 items 列表中选择 **Username Password Form**。
20. 点 **Save**。
21. 在 **Username Password Form** authentication type 中点 **Required**，以将其要求设置为 required。

现在，您可以为第二个身份验证级别配置网络流。

1. 为 **Auth Flow** 点 **Actions**。
2. 点 **Add flow**。
3. 输入 **2nd Condition Flow** 作为一个别名。
4. 点 **Save**。
5. 为 **2nd Condition Flow** 验证类型点击 **Conditional**，将其要求设置为条件。
6. 点击 **2nd Condition Flow** 的 **Actions**。
7. 单击 **Add execution**。
8. 从项列表中选择 **Conditional - Authentication Level**。
9. 点 **Save**。
10. 点 **Conditional - Level Of Authentication** authentication type 来设置其要求。
11. 点 **Authentication Conditional - Level** 的 **Actions**。
12. 单击 **Config**。
13. 输入 **Level 2** 作为别名。
14. 在验证级别(LoA)输入 **2**。
15. 将 Max Age 设置为 **0**。因此，当用户验证时，此级别仅对当前身份验证有效，但不是后续 SSO 身份验证。因此，在请求此级别时，用户总是需要再次通过这个级别进行身份验证。
16. 点 **Save**

配置第二个身份验证级别的条件

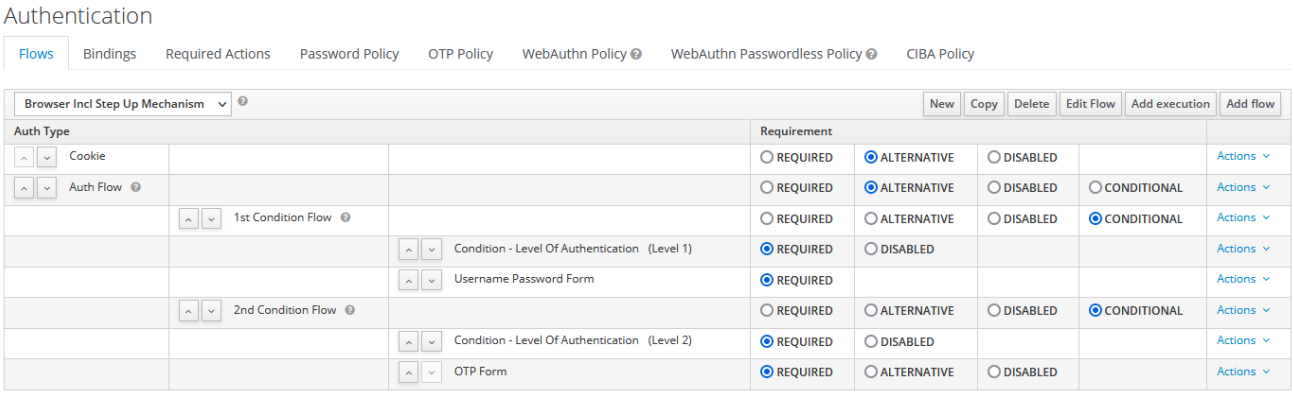
17. 点击 **2nd Condition Flow** 的 **Actions**。
18. 单击 **Add execution**。
19. 从 item 列表中选择 **OTP Form**。

20. 点 **Save**。
21. **点击 OTP Form 身份验证类型所需的设置要求。**

最后，更改绑定。

1. 单击 **Bindings** 选项卡。
2. 点 **Browser Flow** 项列表。
3. 从项目列表中选择 **Browser Incl Step up Mechanism**。
4. 点 **Save**。

使用步骤机制进行浏览器登录



请求特定的身份验证级别

要使用步骤机制，请在身份验证请求中指定请求的身份验证级别(LoA)。**claim** 参数用于此目的：

```
https://{DOMAIN}/auth/realms/{REALMNAME}/protocol/openid-connect/auth?client_id={CLIENT-ID}&redirect_uri={REDIRECT-URI}&scope=openid&response_type=code&response_mode=query&nonce=exg16fxdjcu&claims=%7B%22id_token%22%3A%7B%22acr%22%3A%7B%22essential%22%3Atrue%2C%22values%22%3A%5B%22gold%22%5D%7D%7D%7D
```

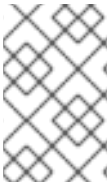
claims 参数以 JSON 指定：

```
claims= {
        "id_token": {
            "acr": {
                "essential": true,
                "values": ["gold"]
            }
        }
    }
```

Red Hat Single Sign-On javascript 适配器支持轻松构建此 JSON 并在登录请求中发送它。如需了解更多详细信息，请参阅 Javascript 适配器文档。

您还可以使用更简单的参数 **acr_values** 而不是 **claims** 参数以请求特定级别作为非意义。这在 OIDC 规格中提到。

您还可以为特定客户端配置默认级别，在参数 **acr_values** 或带有 **acr 声明** 的参数声明时使用。如需了解更多详细信息，请参阅 客户端 ACR 配置。



注意

要请求 acr_values 作为文本（如 **gold**）而不是数值，您可以配置 ACR 和 LoA 之间的映射。可以在 realm 级别（推荐）或客户端级别上配置它。有关配置，请参阅 ACR 到 LoA Mapping。

如需了解更多详细信息，请参阅 官方 OIDC 规格。

流逻辑

上述配置的身份验证流程的逻辑如下：
如果客户端需要高的身份验证级别，即身份验证 2 级别(LoA 2)，用户必须执行完整的双因素身份验证：Username/Password + OTP。但是，如果用户在 Keycloak 中已有一个会话，使用用户名和密码(LoA 1)登录，则用户只会请求第二个身份验证因素(OTP)。

条件中的 **Max Age** 选项决定了后续验证级别有效的时长（以秒为单位）。此设置有助于决定用户在后续身份验证过程中是否再次显示身份验证因素。如果 **声明** 或 **acr_values** 参数请求特定级别 X，并且用户已验证了级别 X，但其已过期（例如，最大期限配置为 310 秒前验证为 300，用户会被要求再次以特定级别重新验证。但是，如果该级别还没有过期，则用户将自动视为具有该级别的身份验证。

使用 **Max Age** 及值 0 意味着该特定级别仅对这个单一身份验证有效。因此，请求该级别的每个验证都需要再次通过该级别进行身份验证。这可用于应用程序中需要更高安全性的操作（例如发送付款），并且始终需要使用特定级别进行身份验证。

警告

请注意，当从客户端发送到 Red Hat Single Sign-On 时，在从客户端发送到 Red Hat Single Sign-On 时，可能会更改 **声明** 或 **acr_values** 等参数。如果客户端使用 PAR（推送授权请求）、请求对象或其他机制（阻止用户在 URL 中重写参数），则可以缓解这种情况。因此，在身份验证后，建议客户端检查 ID 令牌来再次检查令牌中的 **acr** 对应于预期的级别。

如果没有参数明确要求，Red Hat Single Sign-On 将要求身份验证在身份验证流中找到的第一个 LoA 条件，如上例中的 Username/Password。当用户已经通过该级别验证并且该级别已过期时，用户不需要重新验证，但令牌中的 **acr** 将具有值 0。因此，根据 OIDC Core 1.0 规格第 2 部分所述，这只 **基于长期浏览器 Cookie** 被视为身份验证。

注意

当管理员指定了多个流时，可能会发生冲突情况，将不同的 LoA 级别设置为不同的客户端，并为不同的客户端分配流。但是，该规则始终相同：如果用户具有特定级别，则只需要将该级别连接到客户端。它是管理员，确保 LoA 一致。

示例情境

1. 最大期限的 1 条件配置为 300 秒。

2. 发送登录请求，但不请求任何cr。将使用 1 级别，用户需要使用用户名和密码进行身份验证。令牌将具有 **acr=1**。

3. 在 100 秒后发送另一个登录请求。由于 SSO 到期，用户会自动进行身份验证，令牌将返回 **acr=1**。

4. 另一登录请求在 201 秒后发出（自 2 中的身份验证后计算为 301 秒）由于 SSO 的关系，用户会自动进行身份验证，但令牌将在级别 1 被视为过期时返回 **acr=0**。

5. 发送另一个登录请求，但现在，它将在 **声明** 参数中明确请求为级别 1 的 ACR。用户需要使用用户名/密码重新进行身份验证，然后在令牌中返回 **acr=1**。

令牌中的 ACR 声明

ACR 声明通过在 **acr** 客户端范围中定义的 **acr loa level** 协议映射程序添加到令牌中。此客户端范围是 realm 默认客户端范围，因此将添加到 realm 中所有新创建的客户端。

如果您不希望令牌内部的 **acr** 声明或需要一些自定义逻辑来添加它，您可以从客户端中删除客户端范围。

请注意，当登录请求使用请求作为 **基本 声明的** claim 参数发起请求 **时**，Red Hat Sign-On 将始终返回其中一个指定级别。如果无法返回指定级别之一（例如，如果请求的级别未知或大于身份验证流程中的配置条件），则 Red Hat Single Sign-On 将抛出错误。

8.3.5. 配置用户会话限制

对用户可以配置的会话数量限制。每个域或每个客户端都可以限制会话。

要向流添加会话限制，请执行以下步骤：

1. 单击 **流的 Add** execute。
2. 从项目列表中选择 **User Session Count Limiter**。
3. 点 **Save**。
4. 为 **User Session Count Limiter** 验证类型点 **Required**，将它的 requirement 设置为 required。
5. 单击 **User Session Count Limiter** 的 **Actions**。
6. 单击 **Config**。
7. 输入此配置的别名。
8. 输入用户可在此域中具有的最大会话数。如果使用 0，则禁用此检查。
9. 输入用户可为客户端拥有的最大会话数。如果使用 0，则禁用此检查。
10. 选择在达到限制后尝试创建会话时所需的行为。可用的 bahaviors 是：

    > Deny new session - when a new session is requested and the session limit is reached, no new sessions can be created.
    > Terminate oldest session - when a new session is requested and the session limit has been reached, the oldest session will be removed and the new session created.

11. 另外，还可在达到限制时添加自定义错误消息。

请注意，用户会话限制应添加到绑定 **浏览器流**、**Direct Grant Flow**、**Reset Credentials** 以及任何配置的身份提供程序上的任何 **post Login Flow** 中。目前，管理员负责在不同配置之间保持一致性。

另外，CIBA 不提供用户会话限制功能。

8.4. KERBEROS

Red Hat Single Sign-On 支持通过 Simple 和 Protected GSSAPI Negotiation Mechanism (SPNEGO)协议进行 Kerberos 票据登录。SPNEGO 在用户进行身份验证后通过 Web 浏览器进行透明验证。对于非 Web 情况，或在登录过程中无法使用一个 ticket，Red Hat Single Sign-On 支持使用 Kerberos 用户名和密码登录。

Web 身份验证的典型用例如下：

1. 用户登录桌面。
2. 用户使用浏览器访问由 Red Hat Single Sign-On 保护的 Web 应用。
3. 应用重定向到 Red Hat Single Sign-On 登录。
4. Red Hat Single Sign-On 会呈现 HTML 登录屏幕，状态为 401 和 HTTP 标头 **WWW-Authenticate: Negotiate**
5. 如果浏览器有来自桌面登录的 Kerberos 票据，浏览器会将桌面登录信息传送到标题 **授权中的 Red Hat Sign-On: Negotiate 'spnego-token'**。否则，它会显示标准登录屏幕，用户输入登录凭证。
6. Red Hat Single Sign-On 从浏览器中验证令牌并验证用户。
7. 如果将 LDAPFederationProvider 与 Kerberos 验证支持搭配使用，红帽单点登录从 LDAP 中置备用户数据。如果使用 KerberosFederationProvider，Red Hat Single Sign-On 允许用户更新配置集并预先填充登录数据。
8. 红帽单点登录返回应用程序。Red Hat Single Sign-On，应用程序通过 OpenID Connect 或 SAML 消息进行通信。红帽单点登录充当 Kerberos/SPNEGO 登录的代理。因此，应用程序会隐藏通过 Kerberos 进行 Red Hat Single Sign-On 验证。

执行以下步骤设置 Kerberos 身份验证：

1. Kerberos 服务器的设置和配置。
2. Red Hat Single Sign-On 服务器的设置和配置。
3. 客户端机器的设置和配置。

8.4.1. Kerberos 服务器设置

设置 Kerberos 服务器的步骤取决于操作系统(OS)和 Kerberos 供应商。有关设置和配置 Kerberos 服务器的说明，请参阅 Windows Active Directory、MIT Kerberos 和您的 OS 文档。

在设置过程中，执行以下步骤：

1. 在 Kerberos 数据库中添加一些用户主体。您还可以将 Kerberos 与 LDAP 集成，以便从 LDAP 服务器置备用户帐户。

2. 为 "HTTP" 服务添加服务主体。例如，如果 Red Hat Single Sign-On 服务器在 **www.mydomain.org** 上运行，请添加服务主体 **HTTP/www.mydomain.org@<kerberos realm>**。
   在 MIT Kerberos 中，您将运行"kadmin"会话。在带有 MIT Kerberos 的机器中，您可以使用以下命令：

```
sudo kadmin.local
```

然后，使用命令添加 HTTP 主体并将其密钥导出到 keytab 文件中，例如：

```
addprinc -randkey HTTP/www.mydomain.org@MYDOMAIN.ORG
ktadd -k /tmp/http.keytab HTTP/www.mydomain.org@MYDOMAIN.ORG
```

在运行 Red Hat Single Sign-On 的主机上，确保 keytab 文件 **/tmp/http.keytab** 可访问。

8.4.2. 设置和配置 Red Hat Single Sign-On 服务器

在机器上安装 Kerberos 客户端。

流程

1. 安装 Kerberos 客户端。如果您的机器运行 Fedora、Ubuntu 或 RHEL，请安装 freeipa-client 软件包，其中包含 Kerberos 客户端和其他实用程序。

2. 配置 Kerberos 客户端（在 Linux 中，配置设置位于 /etc/krb5.conf 文件中）。
   将您的 Kerberos 域添加到配置中，并配置服务器运行的 HTTP 域。

   例如，对于 MYDOMAIN.ORG 域，您可以配置 **domain_realm** 部分，如下所示：

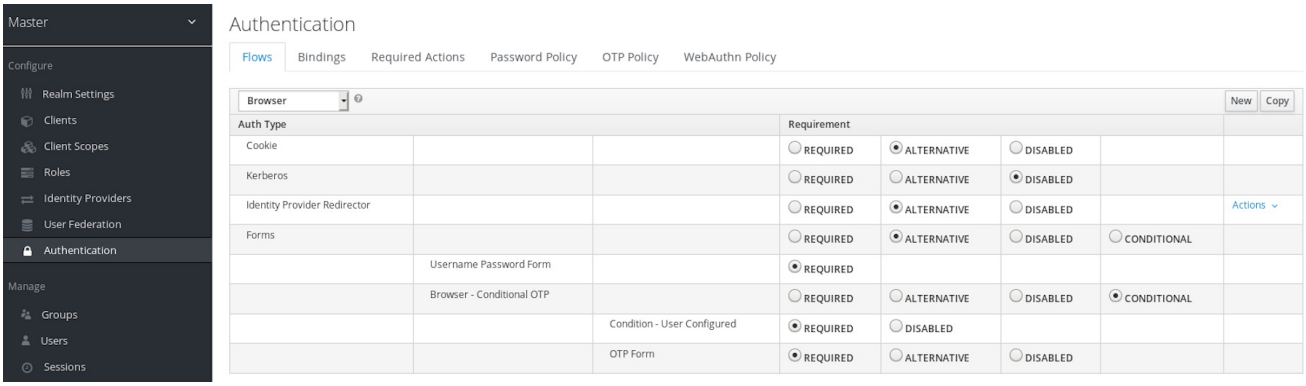
```
[domain_realm]
  .mydomain.org = MYDOMAIN.ORG
  mydomain.org = MYDOMAIN.ORG
```

3. 使用 HTTP 主体导出 keytab 文件，并确保运行 Red Hat Single Sign-On 服务器的进程可以访问该文件。对于生产环境，请确保此文件只可由这个进程读取。
   对于以上 MIT Kerberos 示例，我们将 keytab 导出到 **/tmp/http.keytab** 文件。如果您的 *密钥分发中心(KDC)* 和 Red Hat Single Sign-On 在同一主机上运行，则该文件已可用。

8.4.2.1. 启用 SPNEGO 处理

默认情况下，Red Hat Single Sign-On 禁用 SPNEGO 协议支持。要启用它，请转至 浏览器流 并启用 **Kerberos**。

浏览器流



将 *禁用的* **Kerberos** 要求设置为 *备选* (Kerberos 是可选的)或 *必需* （浏览器必须启用 Kerberos）。如果您尚未将浏览器配置为与 SPNEGO 或 Kerberos 一起使用，Red Hat Single Sign-On 将回退到常规登录屏幕。

8.4.2.2. 配置 Kerberos 用户存储联合 providerx

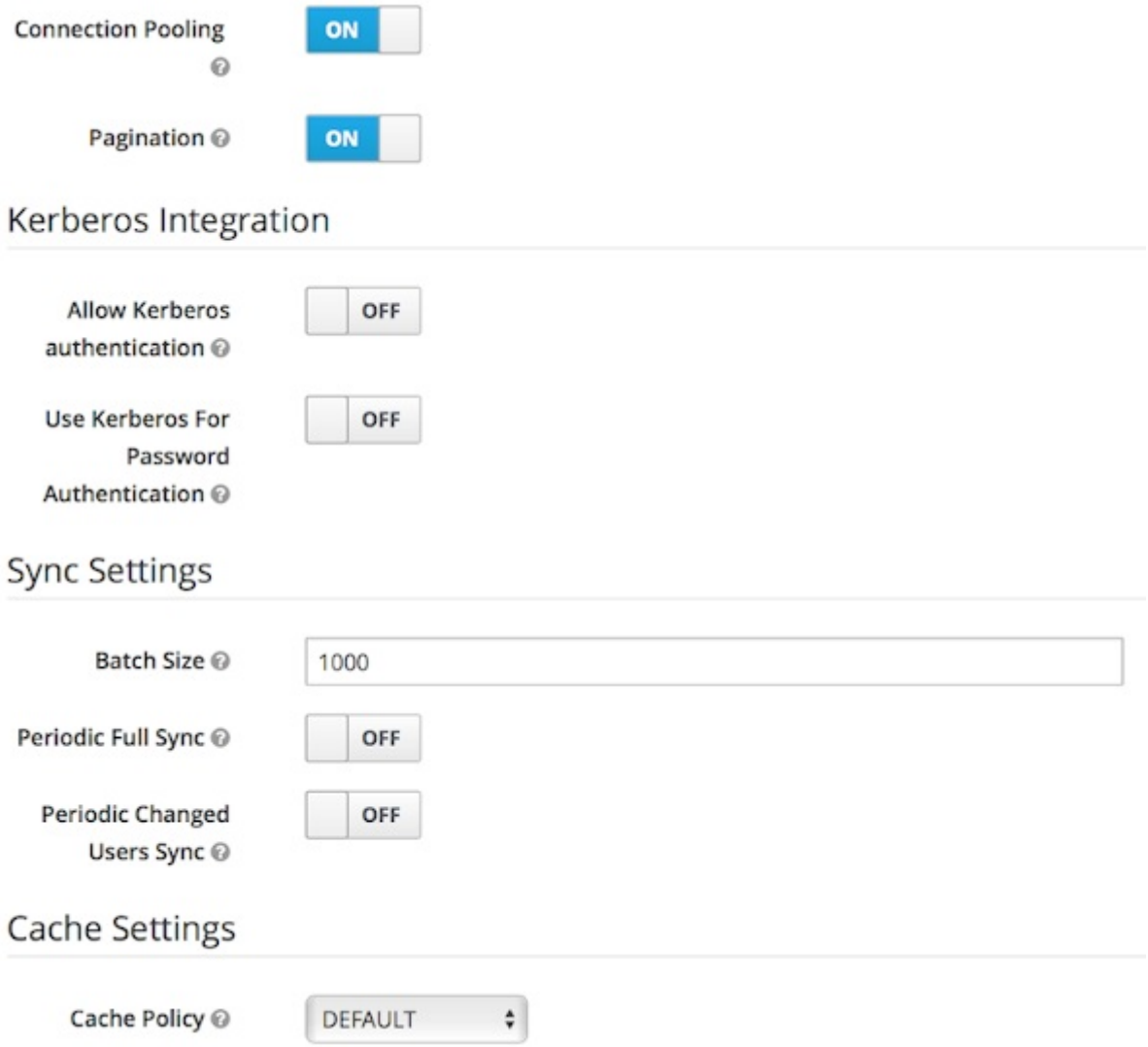
现在，您必须使用 User Storage Federation 来配置 Red Hat Single Sign-On 如何解释 Kerberos 票据。有两个不同的联合供应商提供 Kerberos 身份验证支持。

要通过 LDAP 服务器支持的 Kerberos 进行身份验证，请配置 LDAP Federation Provider。

流程

1. 进入 LDAP 供应商的配置页面。

LDAP kerberos 集成



2. 将允许 **Kerberos 身份验证** 切换到 **ON**

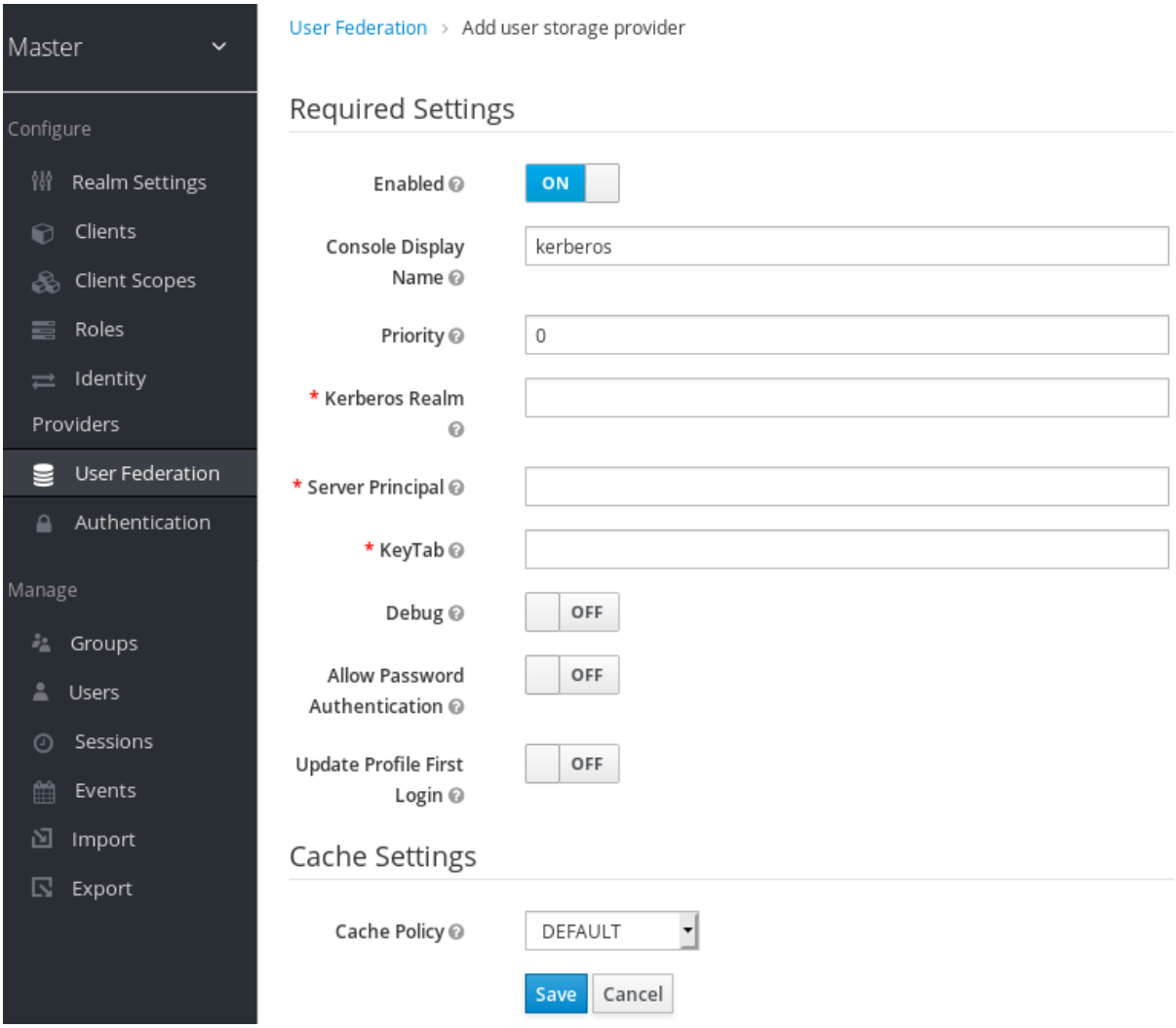
允许 Kerberos 身份验证 使 Red Hat Single Sign-On 使用 Kerberos 主体访问用户信息，以便信息可以导入到红帽单点登录环境中。

如果 LDAP 服务器没有备份您的 Kerberos 解决方案，请使用 **Kerberos** User Storage Federation Provider。

流程

1. 点菜单中的 **User Federation**。
2. 从 **Add provider** 中选择 **Kerberos**。

Kerberos 用户存储供应商



Kerberos 供应商解析 Kerberos ticket 用于简单主体信息，并将信息导入到本地 Red Hat Single Sign-On 数据库中。用户配置集信息（如名字、姓氏和电子邮件）不会被调配。

8.4.3. 设置和配置客户端机器

客户端机器必须具有 Kerberos 客户端并设置 **krb5.conf**，如上所述。客户端机器必须在其浏览器中启用 SPNEGO 登录支持。如果您使用的是 Firefox 浏览器，请参阅 为 Kerberos 配置 Firefox。

.mydomain.org URI 必须位于 **network。协商-auth.trusted-uris** 配置选项。

在 Windows 域中，客户端不需要调整其配置。Internet Explorer 和 Edge 已经参与 SPNEGO 身份验证。

8.4.4. 凭证委托

Kerberos 支持凭据委派。应用程序可能需要访问 Kerberos ticket，以便可以重新使用它与 Kerberos 保护的其他服务交互。因为 Red Hat Single Sign-On 服务器处理 SPNEGO 协议，所以您必须将 GSS 凭证传播到 OpenID Connect 令牌声明或 SAML 断言属性中的应用程序。Red Hat Single Sign-On 从 Red Hat Single Sign-On 服务器将其传输到您的应用程序。要将此声明插入到令牌或断言中，每个应用程序都必须启用内置协议映射程序 **gss delegation 凭据**。此 **映射程序** 位于应用程序客户端页面的"映射程序"选项卡中。如需了解更多详细信息，请参阅协议映射程序 章节。

应用程序必须反序列化它从 Red Hat Single Sign-On 接收的声明，然后才能向其他服务发出 GSS 调用。当您将凭证从访问令牌映射到 GSSCredential 对象时，使用传递给 **GSSManager.createContext** 方法创建 GSSContext。例如：

```
// Obtain accessToken in your application.
KeycloakPrincipal keycloakPrincipal = (KeycloakPrincipal) servletReq.getUserPrincipal();
AccessToken accessToken = keycloakPrincipal.getKeycloakSecurityContext().getToken();

// Retrieve Kerberos credential from accessToken and deserialize it
String serializedGssCredential = (String) accessToken.getOtherClaims().
    get(org.keycloak.common.constants.KerberosConstants.GSS_DELEGATION_CREDENTIAL);

GSSCredential deserializedGssCredential = org.keycloak.common.util.KerberosSerializationUtils.
    deserializeCredential(serializedGssCredential);

// Create GSSContext to call other Kerberos-secured services
GSSContext context = gssManager.createContext(serviceName, krb5Oid,
    deserializedGssCredential, GSSContext.DEFAULT_LIFETIME);
```

注意

在 **krb5.conf** 文件中配置 **forwardable** Kerberos 票据，并在浏览器中添加对授权凭证的支持。

警告

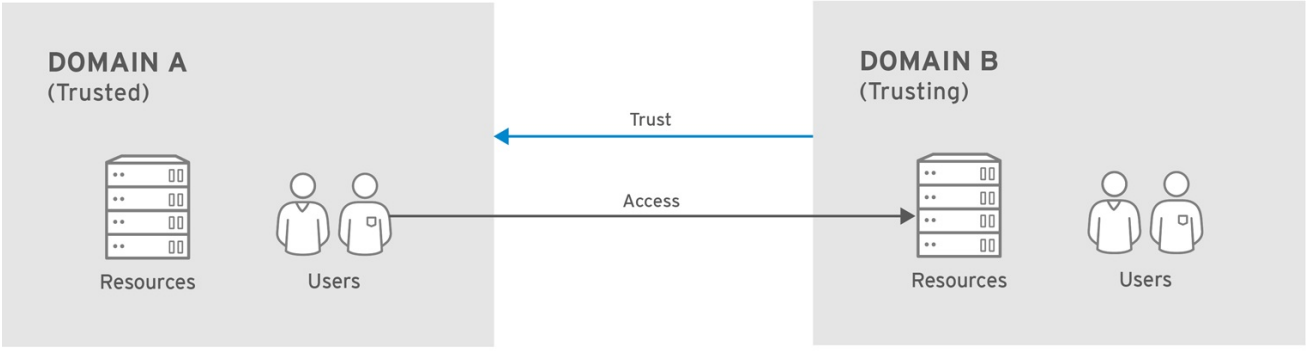
委派具有安全隐患，因此仅在需要且仅在 HTTPS 中使用它。有关更多详细信息和示例，请参阅本文。

8.4.5. 跨域信任

在 Kerberos 协议中，**realm** 是一组 Kerberos 主体。这些主体的定义存在于 Kerberos 数据库中，通常是 LDAP 服务器。

Kerberos 协议允许跨域信任。例如，如果存在 2 个 Kerberos 域、A 和 B，那么跨域信任将允许 realm A 的用户访问 realm B 的资源。realm B 信任域 A.

Kerberos 跨域信任



Red Hat Single Sign-On 服务器支持跨域信任。要做到这一点，请执行以下操作：

- 为跨域信任配置 Kerberos 服务器。实施此步骤取决于 Kerberos 服务器实现。此步骤是将 Kerberos 主体 **krbtgt/B@A** 添加到 realm A 和 B 的 Kerberos 数据库中。这个主体必须在 Kerberos 域上都有相同的密钥。主体必须在两个域中都有相同的密码、密钥版本号和密码。详情请查看 Kerberos 服务器文档。

注意

默认情况下，跨域信任是单向。您必须将主体 **krbtgt/A@B** 添加到 Kerberos 数据库，以便在域 A 和域 B 之间进行双向信任。但是，默认情况下信任是传输的。如果 realm B 信任域 A 和 realm C 信任域 B，那么 realm C 信任域 A 不使用主体 **krbtgt/C@A**，可用。Kerberos 客户端中可能需要其他配置（例如 **capaths**），以便客户端能够找到信任路径。详情请查看 Kerberos 文档。

- 配置红帽单点登录服务器
    - 当使用带有 Kerberos 支持的 LDAP 存储供应商时，为域 B 配置服务器主体，如下例所示：**HTTP/mydomain.com@B**。如果域 A 中的用户成功向 Red Hat Single Sign-On 进行身份验证，LDAP 服务器必须查找来自 realm A 的用户，因为红帽单点登录必须执行 SPNEGO 流，然后查找用户。

查找用户基于 LDAP 存储供应商选项 **Kerberos 主体属性**。当为实例配置了类似 **userPrincipalName** 的值的实例时，在用户 **john@A** 的 SPNEGO 身份验证后，Red Hat Single Sign-On 将尝试查找与 **john@A** 等效属性 **userPrincipalName** 的 LDAP 用户。如果 **Kerberos 主体属性** 留空，则 Red Hat Single Sign-On 将根据其 kerberos 主体的前缀并使用域省略来查找 LDAP 用户。例如，Kerberos 主体用户 **john@A** 必须在用户名 **john** 下的 LDAP 中可用，因此通常位于 LDAP DN 下，如 **uid=john,ou=People,dc=example,dc=com**。如果您希望域 A 和 B 中的用户进行身份验证，请确保 LDAP 可以从 realms A 和 B 中查找用户。

- 当使用 Kerberos 用户存储供应商（通常是没有 LDAP 集成）时，将服务器主体配置为 **HTTP/mydomain.com@B**，并且来自 Kerberos 域 A 和 B 的用户必须能够进行身份验证。

支持多个 Kerberos 域中的用户，因为每个用户都有属性 **KERBEROS_PRINCIPAL** 引用用于身份验证的 kerberos 主体，这用于进一步查找此用户。为了避免在 kerberos realm **A** 和 **B** 中用户 **john** 时存在冲突，Red Hat Single Sign-On 用户的用户名可能包含 kerberos 域小写。例如，用户名将是 **john@a**。仅在 realm 与配置的 **Kerberos 域** 匹配时，可能会从生成的用户名中省略 realm 后缀。例如，只要 Kerberos 提供程序上配置了 Kerberos **域是 A，即 Kerberos 主体 john @ A** 的 john。

8.4.6. 故障排除

如果出现问题，请启用额外的日志记录来调试问题：

- 在 Kerberos 或 LDAP 联合供应商的管理控制台中启用 **Debug** 标记
- 为类别 **org.keycloak** 启用 TRACE logging，以在服务器日志中接收更多信息
- 添加系统属性 **-Dsun.security.krb5.debug=true** 和 **-Dsun.security.spnego.debug=true**

8.5. X.509 客户端证书用户身份验证

如果您已将服务器配置为使用 mutual SSL 身份验证，则 Red Hat Single Sign-On 支持使用 X.509 客户端证书登录。

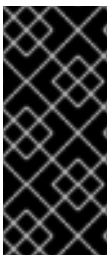
典型的工作流：

- 客户端通过 SSL/TLS 频道发送身份验证请求。

- 在 SSL/TLS 握手期间，服务器和客户端会交换其 x.509/v3 证书。
- 容器(JBoss EAP)验证证书 PKIX 路径和证书到期日期。
- X.509 客户端证书验证器使用以下方法之一验证客户端证书：
  - 使用 CRL 或 CRL 分发点检查证书撤销状态。
  - 使用 OCSP （在线证书状态协议）检查证书撤销状态。
  - 验证证书中的密钥是否与预期的密钥匹配。
  - 验证证书中的扩展密钥是否与预期的扩展密钥匹配。
- 如果这些检查中有任何一个失败，x.509 身份验证会失败。否则，身份验证器会提取证书身份并将其映射到现有用户。

当证书映射到现有用户时，行为会根据身份验证流来剥离：

- 在浏览器流中，服务器会提示用户确认其身份或使用用户名和密码登录。
- 在 Direct Grant 流中，服务器会登录到用户。



重要

请注意，Web 容器负责验证证书 PKIX 路径。Red Hat Single Sign-On 一侧的 X.509 验证器只提供对检查证书过期、证书撤销状态和密钥使用的额外支持。如果您在使用在反向代理后部署的 Red Hat Single Sign-On，请确保您的反向代理配置为验证 PKIX 路径。如果您不使用反向代理和用户直接访问 JBoss EAP，您应该象 JBoss EAP 一样确保验证 PKIX 路径，只要它进行了配置，如下所示：

8.5.1. 功能

支持的证书身份源：

- 使用正则表达式匹配 SubjectDN
- X500 Subject 的 email 属性
- X500 使用者电子邮件地址来自主题备用名称(RFC822Name 常规名称)
- X500 对象来自主题备用名称扩展的其他名称。这个其他名称通常是 User Principal Name (UPN)。
- X500 Subject 的 Common Name 属性
- 使用正则表达式匹配 IssuerDN
- 证书序列号
- 证书 Serial Number 和 IssuerDN
- SHA-256 证书指纹
- PEM 格式的完整证书

8.5.1.1. 正则表达式

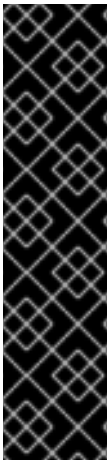
Red Hat Single Sign-On 使用正则表达式作为过滤器从 Subject DN 或 Issuer DN 中提取证书身份。例如，此正则表达式与 email 属性匹配：

```
emailAddress=(.*?)(?:,|$)
```

如果将 **Identity Source** 设置为 **Match SubjectDN using regular expression** 或 **Match IssuerDN using regular expression**，会应用正则表达式过滤。

8.5.1.1.1. 将证书身份映射到现有用户

证书身份映射可以将提取的用户身份映射到现有用户的用户名、电子邮件或自定义属性，其值与证书身份匹配。例如，将 **身份源** 设置为 *Subject 的电子邮件* **或用户映射方法为** *Username 或 email*，使 X.509 客户端证书验证器使用证书主题 DN 中的 email 属性作为搜索条件，因为按用户名或电子邮件搜索现有用户。



重要

- 如果您在 realm 设置中禁用 **带有电子邮件** 的登录，则同一规则适用于证书身份验证。用户无法使用 email 属性登录。
- 使用 证书 **Serial Number 和 IssuerDN** 作为身份源需要两个序列号和 IssuerDN 的自定义属性。
- **SHA-256 证书指纹** 是 SHA-256 证书指纹的小写十六进制表示。
- 使用 **PEM 格式的完整证书** 作为身份源仅限于映射到外部联合源（如 LDAP）的自定义属性。由于有长限制，Red Hat Single Sign-On 无法将证书存储在其数据库中，因此，如果是 LDAP，您必须启用 **Always Read Value From LDAP**。

8.5.1.1.2. 扩展证书验证

- 使用 CRL 撤销状态检查。
- 使用 CRL/Distribution Point 撤销状态检查。
- 使用 OCSP/Responder URI 撤销状态检查。
- 证书密钥验证.
- 证书扩展密钥验证.

8.5.2. 启用 X.509 客户端证书用户身份验证

以下小节介绍了如何配置 JBoss EAP/Undertow 和红帽单点登录服务器以启用 X.509 客户端证书身份验证。

8.5.2.1. 在 JBoss EAP 中启用 mutual SSL

如果在 JBoss EAP 中启用 SSL 的说明，请参阅启用 SSL。

- 打开 RHSSO_HOME/standalone/configuration/standalone.xml，再添加一个新域：

```
<security-realms>
```

```
    <security-realm name="ssl-realm">
      <server-identities>
        <ssl>
          <keystore path="servercert.jks"
                    relative-to="jboss.server.config.dir"
                    keystore-password="servercert password"/>
        </ssl>
      </server-identities>
      <authentication>
        <truststore path="truststore.jks"
                    relative-to="jboss.server.config.dir"
                    keystore-password="truststore password"/>
      </authentication>
    </security-realm>
</security-realms>
```

ssl/keystore

ssl 元素包含 **密钥存储** 元素，其中包含从 JKS 密钥存储加载服务器公钥对的详细信息。

ssl/keystore/path

JKS 密钥存储的路径。

ssl/keystore/relative-to

密钥存储路径相对于的路径。

ssl/keystore/keystore-password

打开密钥存储的密码。

SSL/keystore/alias （可选）

密钥存储中的条目的别名。如果密钥存储包含多个条目，则设置。

SSL/keystore/key-password （可选）

私钥密码与密钥存储密码不同。

authentication/truststore

定义如何加载信任存储，以验证入站/外连接的远程端提供的证书。通常，truststore 包含一组可信 CA 证书。

authentication/truststore/path

JKS 密钥存储的路径，包含可信证书颁发机构的证书。

authentication/truststore/relative-to

truststore 路径的路径是相对路径的。

authentication/truststore/keystore-password

打开信任存储的密码。

8.5.2.2. 启用 HTTPS 侦听器

有关在 WildFly 中启用 HTTPS 的说明，请参阅 HTTPS Listener。

- 添加 <https-listener> 元素。

```
<subsystem xmlns="urn:jboss:domain:undertow:12.0">
 ....
   <server name="default-server">
    <https-listener name="default"
                    socket-binding="https"
```

```
                security-realm="ssl-realm"
                verify-client="REQUESTED"/>
    </server>
</subsystem>
```

https-listener/security-realm

这个值必须与上一节中的域的名称匹配。

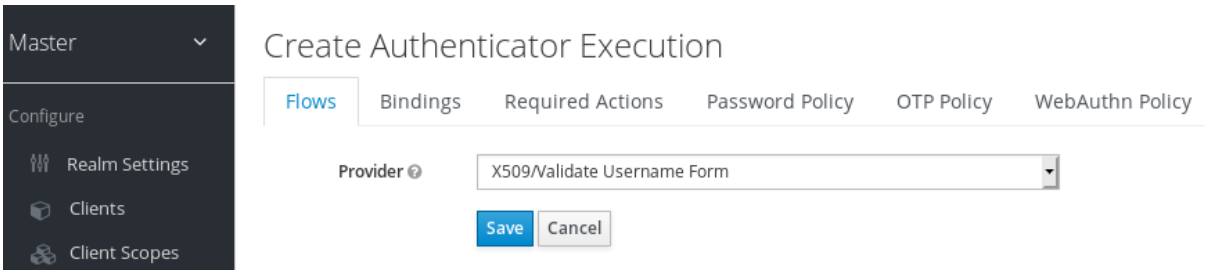
https-listener/verify-client

如果设置为 **REQUESTED**，则服务器会选择性地询问客户端证书。如果设置为 **REQUIRED**，则服务器将拒绝入站连接（如果没有提供客户端证书）。

8.5.3. 在浏览器流中添加 X.509 客户端证书验证

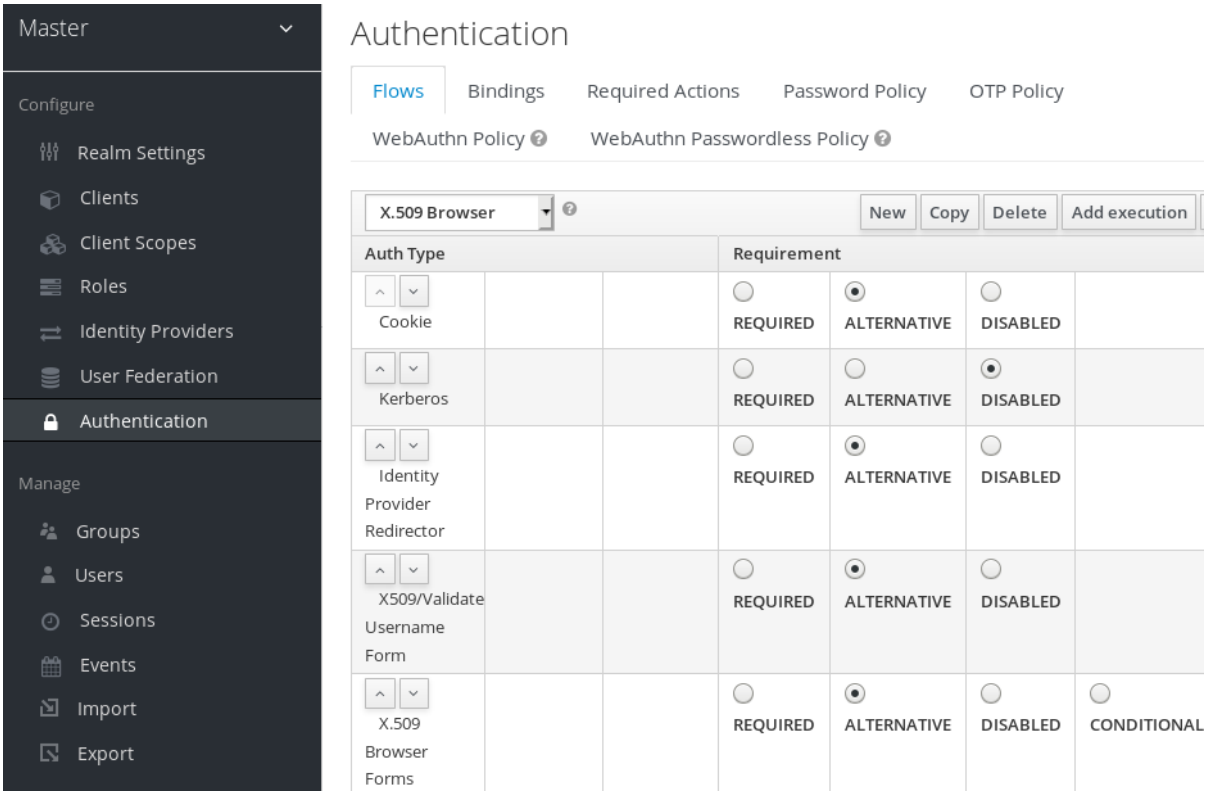
1. 在菜单中，单击 **Authentication**。
2. 点 "Browser" 流。
3. 单击 **Copy**，以制作内置"Browser"流的副本。
4. 输入副本的名称。
5. 点 **确定**。
6. 点 **Add policy** 下拉列表中的 copy。
7. 单击 **Add execution**。
8. 点击 "X509/Validate Username Form"。
9. 点 **Save**。

X509 执行



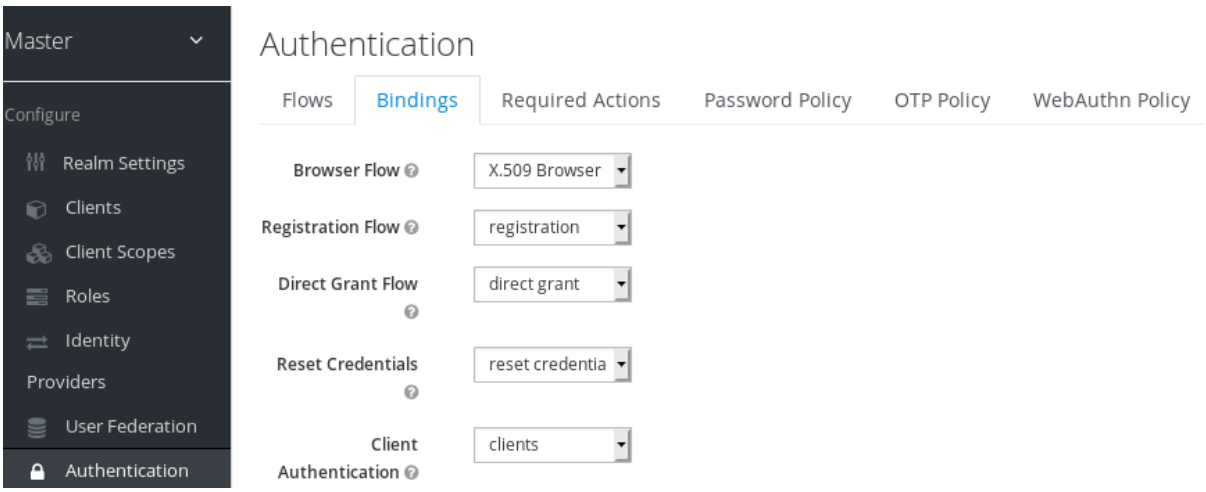
10. 点击上箭头按钮将"X509/Validate Username Forms"移到"Browser Forms"执行。
11. 将要求设置为"ALTERNATIVE"。

X509 浏览器流



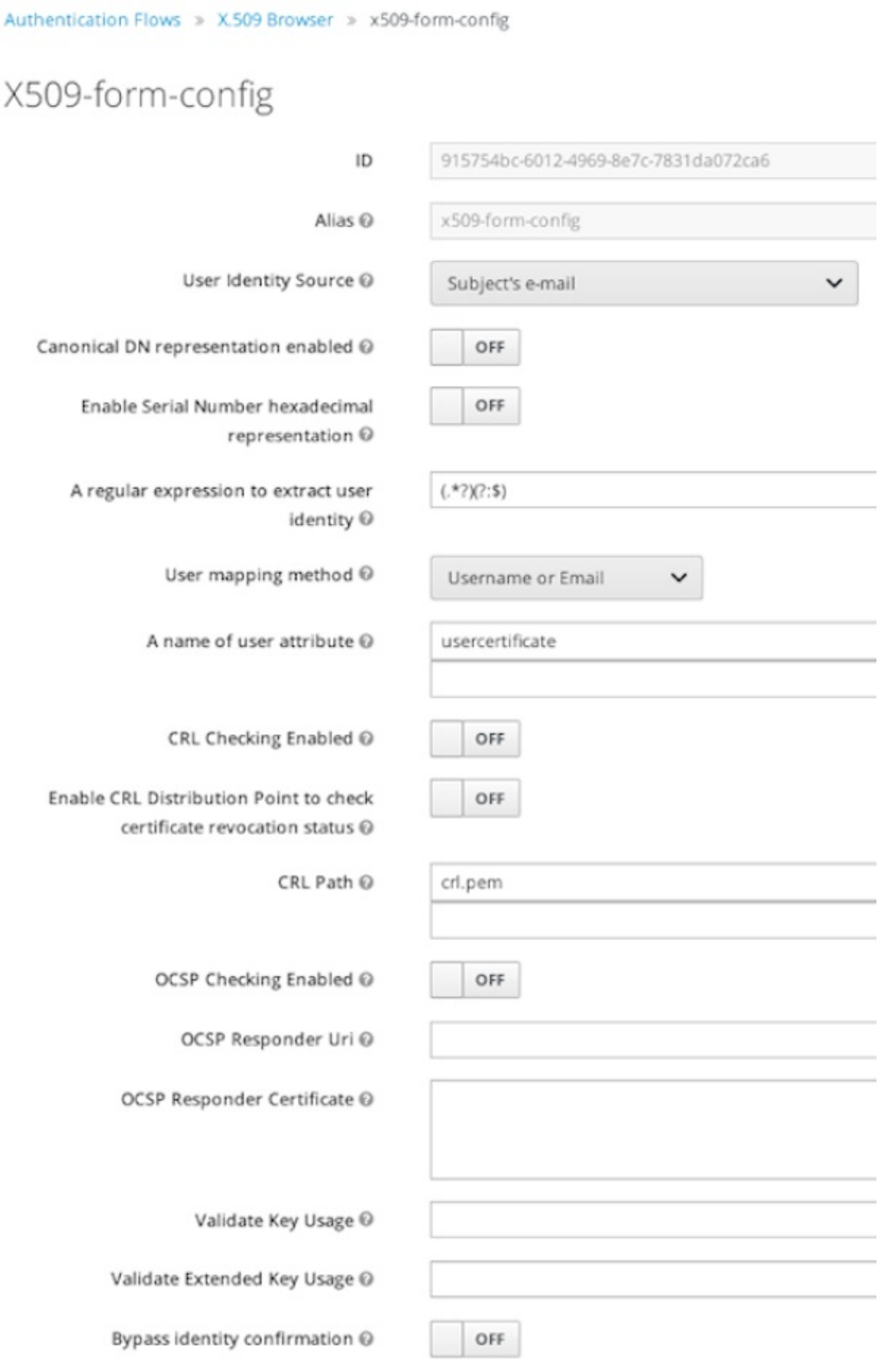
12. 单击 **Bindings** 选项卡。
13. 点击 **Browser Flow** 下拉列表。
14. 从下拉列表中选择浏览器流的副本。
15. 点 **Save**。

X509 浏览器流绑定



8.5.4. 配置 X.509 客户端证书验证

X509 配置


Authentication Flows » X.509 Browser » x509-form-config
X509-form-config
ID
915754bc-6012-4969-8e7c-7831da072ca6
Alias
x509-form-config
User Identity Source
Subject's e-mail
Canonical DN representation enabled
OFF
Enable Serial Number hexadecimal representation
OFF
A regular expression to extract user identity
(.*?)(?:$)
User mapping method
Username or Email
A name of user attribute
usercertificate
CRL Checking Enabled
OFF
Enable CRL Distribution Point to check certificate revocation status
OFF
CRL Path
crl.pem
OCSP Checking Enabled
OFF
OCSP Responder Uri
OCSP Responder Certificate
Validate Key Usage
Validate Extended Key Usage
Bypass identity confirmation
OFF

用户身份源

定义用于从客户端证书提取用户身份的方法。

启用规范 DN 表示

定义是否使用规范格式来确定可识别的名称。官方 Java API 文档描述了格式。这个选项 *使用正则表达式对两个用户身份来源 Match SubjectDN* 的影响，仅使用 *正则表达式*。在设置新的 Red Hat Single Sign-On 实例时启用这个选项。禁用这个选项，以保持与现有 Red Hat Single Sign-On 实例的向后兼容性。

启用串行数十六进制表示

以十六进制表示 序列号。设置了 sign 位的序列号必须为 1，必须留使用 00 八进制数。例如，根据 RFC5280，带有十进制值 *161* 或十六进制的 *a1* 的序列号被编码为 *00a1*。如需了解更多详细信息，请参阅 RFC5280, appendix-B。

正则表达式

用作提取证书身份的过滤器的正则表达式。表达式必须包含单个组。

用户映射方法

定义与现有用户身份匹配的方法。*根据用户名或电子邮件* 搜索现有用户的用户名或电子邮件.*自定义属性映射程序* 会搜索具有与证书身份匹配的自定义属性的现有用户。自定义属性的名称是可配置的。

名称 user 属性

其值与证书身份匹配的自定义属性。如果属性映射与多个值相关，请使用多个自定义属性，例如：'Certificate Serial Number 和 IssuerDN'。

启用 CRL 检查

使用 Certificate Revocation List 检查证书的吊销状态。列表的位置在 **CRL 文件路径** 属性中定义。

启用 CRL Distribution Point 来检查证书撤销状态

使用 CDP 检查证书撤销状态。大多数 PKI 颁发机构在证书中包含 CDP。

CRL 文件路径

包含 CRL 列表的文件路径。如果启用了 **CRL Checking Enabled** 选项，则该值必须是到有效文件的路径。

启用 OCSP 检查

使用在线证书状态协议检查证书撤销状态。

OCSP Fail-Open Behavior

默认情况下，OCSP 检查必须返回正响应，以便继续成功验证。有时，这个检查可能会排除：例如，OCSP 服务器无法访问，超载，或者客户端证书可能不包含 OCSP 响应器 URI。当启用此设置时，只有在 OCSP 响应程序收到明确的负响应并且证书绝对被撤销时，才会拒绝身份验证。如果一个有效的 OCSP 响应不是无效的验证尝试，则会接受验证尝试。

OCSP Responder URI

覆盖证书中的 OCSP 响应者 URI 值。

验证密钥使用

验证是否已设置证书的密钥扩展位。例如："数字签名,KeyEncipherment"验证是否设置了 KeyUsage 扩展中的位 0 和 2。保留此参数为空以禁用 Key Usage 验证。如需更多信息，请参阅 RFC5280, Section-4.2.1.3。当键使用不匹配时，红帽单点登录会引发错误。

验证扩展密钥使用

验证扩展密钥使用扩展中定义的一个或多个目的。如需更多信息，请参阅 RFC5280, Section-4.2.1.12。将此参数设置为空以禁用扩展密钥使用验证。当被发出 CA 和密钥使用扩展不匹配时，Red Hat Single Sign-On 会引发一个错误。

验证证书策略

验证证书策略扩展中定义的一个或多个策略 OID。请参阅 RFC5280、Section-4.2.1.4。将参数留空，以禁用证书策略验证。使用逗号分隔多个策略。

证书策略验证模式

当在 **Validate Certificate Policy** 设置中指定了多个策略时，它决定匹配是否应检查所有请求的策略，或者一个匹配度足以成功身份验证。默认值为 **All**，表示所有请求的策略都应存在于客户端证书中。

绕过身份确认

如果启用，X.509 客户端证书身份验证不会提示用户确认证书身份。红帽单点登录在成功身份验证后对用户进行签名。

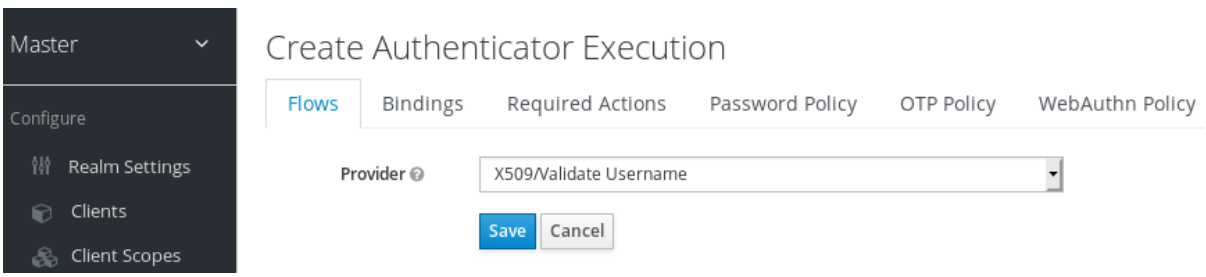
Revalidate 客户端证书

如果设置，客户端证书信任链总是使用配置的信任存储中存在的证书在应用程序级别进行验证。如果底层 Web 服务器没有强制验证客户端证书链验证，例如，它位于非值负载均衡器或反向代理后面，或者当允许 CA 的数量太大时，对于 mutual SSL 协商（大多数浏览器上限的 SSL 协商大小上限为 32767 字节，它对应于 200 公告的 CA）。默认情况下，这个选项是 off。

8.5.5. 在直接授予流中添加 X.509 客户端证书验证

1. 在菜单中，单击 **Authentication**。
2. 点 "Direct Grant" 流。
3. 单击 **Copy**，以制作"Direct Grant"流的副本。
4. 输入副本的名称。
5. 点 **确定**。
6. 单击"用户名验证"的 **Actions** 链接，**再单击删除**。
7. 点 **Delete**。
8. 单击"密码"的 **Actions** 链接，**再单击删除**。
9. 点 **Delete**。
10. 单击 **Add execution**。
11. 点 "X509/Validate Username"。
12. 点 **Save**。

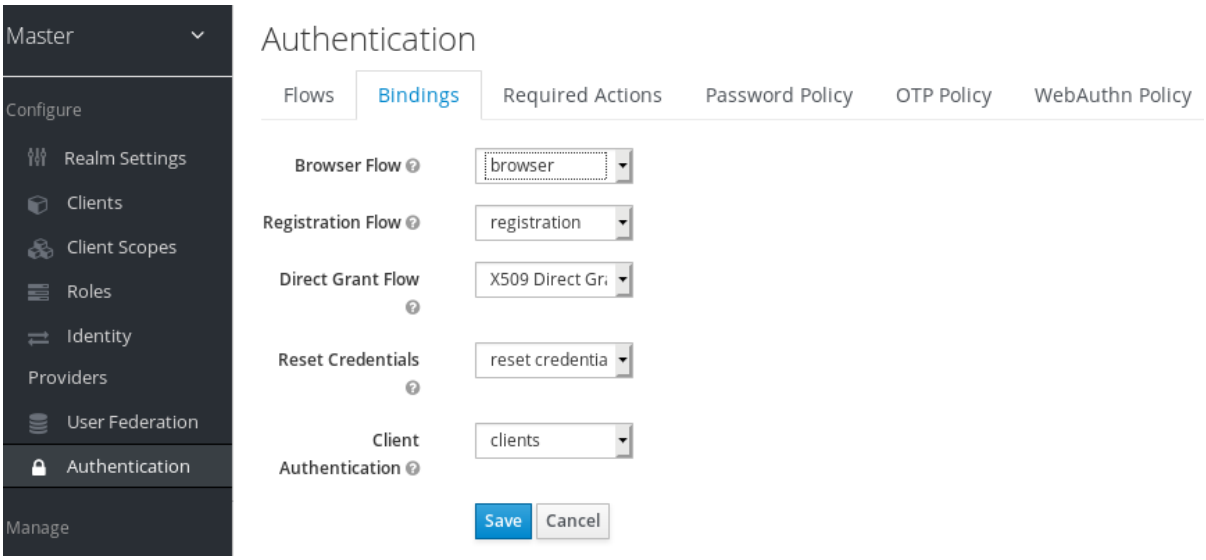
X509 直接授权执行



13. 按照 x509 浏览器流部分中介绍的步骤来设置 x509 身份验证配置。
14. 单击 **Bindings** 选项卡。
15. 点击 **Direct Grant Flow** 下拉列表。
16. 点新创建的 "x509 Direct Grant" 流。

17. 点 **Save**。

X509 直接授权流绑定



8.5.6. 客户端证书查找

当红帽单点登录服务器收到直接 HTTP 请求时，JBoss EAP undertow 子系统建立 SSL 握手并提取客户端证书。JBoss EAP 将客户端证书保存到 HTTP 请求的 **javax.servlet.request.X509Certificate** 属性，如 servlet 规范中所述。Red Hat Single Sign-On X509 验证器可以从此属性中查找证书。

但是，当 Red Hat Single Sign-On 服务器监听负载均衡器或反向代理后面的 HTTP 请求时，代理服务器可以提取客户端证书并建立相互 SSL 连接。反向代理通常将经过身份验证的客户端证书放在底层请求的 HTTP 标头中。代理将请求转发到后端 Red Hat Single Sign-On 服务器。在这种情况下，Red Hat Single Sign-On 必须从 HTTP 标头而不是 HTTP 请求的属性中查找 X.509 证书链。

如果 Red Hat Single Sign-On 位于反向代理后，您通常需要在 RHSSO_HOME/standalone/configuration/standalone.xml 中配置 **x509cert-lookup** SPI 的替代供应商。当 **默认** 供应商查找 HTTP 标头证书时，存在两个其他内置供应商： **haproxy** 和 **apache**。

8.5.6.1. HAProxy 证书查找供应商

当 Red Hat Single Sign-On 服务器位于 HAProxy 反向代理后面时，您可以使用此供应商。为您的服务器使用以下配置：

```
<spi name="x509cert-lookup">
    <default-provider>haproxy</default-provider>
    <provider name="haproxy" enabled="true">
        <properties>
            <property name="sslClientCert" value="SSL_CLIENT_CERT"/>
            <property name="sslCertChainPrefix" value="CERT_CHAIN"/>
            <property name="certificateChainLength" value="10"/>
        </properties>
    </provider>
</spi>
```

在本示例配置中，客户端证书是从 HTTP 标头、**SSL_CLIENT_CERT** 以及其链中其他证书中查找，如 **CERT_CHAIN_0** 到 **CERT_CHAIN_9**。属性 **certificateChainLength** 是链的最大长度，因此最后一个属性是 **CERT_CHAIN_9**。

有关为客户端证书和客户端证书链配置 HTTP 标头的详细信息，请参阅 HAProxy 文档。

8.5.6.2. Apache 证书查找供应商

当 Red Hat Single Sign-On 服务器位于 Apache 反向代理后面时，您可以使用此供应商。为您的服务器使用以下配置：

```
<spi name="x509cert-lookup">
    <default-provider>apache</default-provider>
    <provider name="apache" enabled="true">
        <properties>
            <property name="sslClientCert" value="SSL_CLIENT_CERT"/>
            <property name="sslCertChainPrefix" value="CERT_CHAIN"/>
            <property name="certificateChainLength" value="10"/>
        </properties>
    </provider>
</spi>
```

此配置与 **haproxy** 供应商相同。如需了解有关配置客户端证书和客户端证书链的 HTTP 标头的详细信息，请参阅 mod_ssl 和 mod_headers 上的 Apache 文档。

8.5.6.3. NGINX 证书查找供应商

当 Red Hat Single Sign-On 服务器位于 NGINX 反向代理后面时，您可以使用此供应商。为您的服务器使用以下配置：

```
<spi name="x509cert-lookup">
    <default-provider>nginx</default-provider>
    <provider name="nginx" enabled="true">
        <properties>
            <property name="sslClientCert" value="ssl-client-cert"/>
            <property name="sslCertChainPrefix" value="USELESS"/>
            <property name="certificateChainLength" value="2"/>
        </properties>
    </provider>
</spi>
```

注意

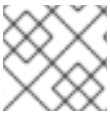
NGINX SSL/TLS 模块 不会公开客户端证书链。Red Hat Single Sign-On 的 NGINX 证书查找提供程序通过使用 Keycloak 信任存储来重建该提供程序。使用 keytool CLI 及所有 root 和中间 CA 用于重建客户端证书链，以此填充 Red Hat Single Sign-On 信任存储。

有关为客户端证书配置 HTTP 标头的详细信息，请参阅 NGINX 文档。

NGINX 配置文件示例：

```
...
server {
    ...
    ssl_client_certificate      trusted-ca-list-for-client-auth.pem;
    ssl_verify_client           optional_no_ca;
    ssl_verify_depth            2;
```

```
    ...
    location / {
      ...
      proxy_set_header ssl-client-cert        $ssl_client_escaped_cert;
      ...
    }
    ...
}
```



注意

trusted-ca-list-for-client-auth.pem 中的所有证书都必须添加到 Keycloak 信任存储中。

8.5.6.4. 其他反向代理实现

Red Hat Single Sign-On 没有可用于其他反向代理实现的内置支持。但是，您可以使其他反向代理的行为与 **apache** 或 **haproxy** 类似。如果没有这些工作，请创建您的 **org.keycloak.services.x509.X509ClientCertificateLookupFactory** 和 **org.keycloak.services.x509.X509ClientCertificateLookup** 提供程序。有关如何添加您的供应商的详细信息，请参阅 服务器开发人员指南。

8.5.7. 故障排除

转储 HTTP 标头

若要查看反向代理向 Keycloak 发送的内容，启用 **RequestDumpingHandler** Undertow 过滤器并查阅 **server.log** 文件。

在 logging 子系统下启用 TRACE 日志记录

```
...
    <profile>
        <subsystem xmlns="urn:jboss:domain:logging:8.0">
...
            <logger category="org.keycloak.authentication.authenticators.x509">
                <level name="TRACE"/>
            </logger>
            <logger category="org.keycloak.services.x509">
                <level name="TRACE"/>
            </logger>
```

警告

不要在生产环境中使用 RequestDumpingHandler 或 TRACE 日志记录。

使用 X.509 直接授予身份验证

要执行这个身份验证，您需要以下内容：

- root CA 和中间 CA

- 用户签名的证书请求，使用 root CA/intermediate CA 签名

您可以使用以下模板，使用 Resource Owner Password Credentials Grant 来请求令牌：

```
$ LOC=<install-dir>/intermediate1/user-certificate

$ curl --insecure https://localhost:8443/auth/realms/X509_demo/protocol/openid-connect/token --data
"grant_type=password" -E $LOC/user1.crt --key $LOC/user1.key --cacert $LOC/intermediate-ca-
chain.crt -d client_id=account -d client_secret=BNm5AQPJGEtbayIAoiKUetr0lkXKSlF4 | jq
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 2097 100 2013 100 84 25481 1063 -::- -::- -::- 26544
{
"access_token":
"eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJ1OUNpN0tzUjBIOEFCQXEtQ1Z4SEFDSUo1
M1hNYWVhclJrYkw4cFd1VW4wIn0.eyJleHAiOjE2Njc4MzA5NjAsImlhdCI6MTY2NzgzMDY2MCwianRpI
oiNDU5YzE2OGMtODU3ZS00OWRjLTgxYjItZjVhM2M3M2MwODMzIiwiaXNzIjoiaHR0cHM6Ly9sb2Nh
bGhvc3Q6ODQ0My9hdXRoL3JlYWxtcy9YNTA5X2RlbW8iLCJzdWIiOiIwODZiMTgyZC00MzdhLTQzZ
DItYTRmZS05ZGZmYTNmOTBiZDAiLCJ0eXAiOiJCZWFyZXIiLCJhenAiOiJhY2NvdW50Iiwic2Vzc2lvbl
9zdGF0ZSI6ImMwYjNiMTJjLTM5YmEtNGQ0Ni1iNDNlLTZkMTM0MGJmNTA5OCIsImFjciI6IjEiLCJyZX
NvdXJjZV9hY2Nlc3MiOnsiYWNjb3VudCI6eyJyb2xlcyI6WyJtYW5hZ2UtYWNjb3VudCIsIm1hbmFnZS1h
Y2NvdW50LWxpbmtzIiwidmlldy1wcm9maWxlIl19fSwic2NvcGUiOiJwcm9maWxlIGVtYWlsIiwic2lkIjoiYzE
M2IxMmMtMzliYS00ZDQ2LWI0M2UtNmQxMzQwYmY1MDk4IiwiZW1haWxfdmVyaWZpZWQiOmZhb
HNlLCJwcmVmZXJyZWRfdXNlcm5hbWUiOiJ1c2VyMSIsImVtYWlsIjoidXNlckByZWRoYXQuY29tIn0.CD
tltEkmITloDpqU5alq4U1JopqEJVeoglT-
wA43edQ_DfeWSgefL0BIrPlt1SKhFMOVitwyq_9XZvfiS5ZiObE33cDmhr6eohbUtDPibU2GuEIYP9WjlV
pZDMaSKQVu5SwM91m6yei22PtH-
ApPOBeG4Ru0xZtNXjwGQpuIJEi_H1rZdPY3I4U2lPuQo4Uono5gnF7re_nUvf90FJi0uaOOrsvUhUkj1x
EwQ0Diy1oIymcbrDL0Ek7B30StBcjn-fe3-
0GpLttLQju0OGTkwD7Eb0UWTKoWAwspMlgpf9NaIGj8rmBsz6eBlGIGWBN2Qg6v3PzbJ2NXKvq435f
9Zg",
"expires_in": 300,
"refresh_expires_in": 1800,
"refresh_token":
"eyJhbGciOiJIUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICIyMmFkZDdhMy0xN2RjLTQ5NmQtYTk4NS05Y
WZhNGZhODVhMTEifQ.eyJleHAiOjE2Njc4MzI0NjAsImlhdCI6MTY2NzgzMDY2MCwianRpIjoiZWU4MjJ
hMzYtMWEzMS00ZGEzLWIxMGEtNmY1ODkxYmI0MzlhIiwiaXNzIjoiaHR0cHM6Ly9sb2NhbGhvc3Q6O
DQ0My9hdXRoL3JlYWxtcy9YNTA5X2RlbW8iLCJhdWQiOiJodHRwczovL2xvY2FsaG9zdDo4NDQzL2F
1dGgvcmVhbG1zL1g1g1MDlfZGVtbyIsInN1YiI6IjA4NmIxODJkLTQzN2EtNDNkMi1hNGZlLTlkZmZhM2Y5
MGJkMCIsInR5cCI6IlJlZnJlc2giLCJhenAiOiJhY2NvdW50Iiwic2Vzc2lvbl9zdGF0ZSI6ImMwYjNiMTJjLTM
5YmEtNGQ0Ni1iNDNlLTZkMTM0MGJmNTA5OCIsInNjb3BlIjoicHJvZmlsZSBlbWFpbCIsInNpZCI6ImM
wYjNiMTJjLTM5YmEtNGQ0Ni1iNDNlLTZkMTM0MGJmNTA5OCJ9.MubgR9rvyrmSOcaq5ce-
qVTPenVQye1KsEHJr7nh9-A",
"token_type": "Bearer",
"not-before-policy": 0,
"session_state": "c0b3b12c-39ba-4d46-b43e-6d1340bf5098",
"scope": "profile email"
}
```

[host][:port]

远程 Red Hat Single Sign-On 服务器的主机和端口号。

user_cert.crt

用户的公钥。

user_cert.key

用户的私钥。这个密钥会验证公钥是否没有被伪造。私钥指向与公钥相同的哈希。

CLIENT_ID

客户端 ID。

CLIENT_SECRET

对于机密客户端，客户端机密。

8.6. W3C WEB 身份验证(WEBAUTHN)

红帽单点登录提供对 W3C Web 身份验证(WebAuthn) 的支持。Red Hat Single Sign-On 作为 WebAuthn 的 Rely ing party (RP) 工作。

注意

WebAuthn 的运营成功取决于用户的 WebAuthn 支持验证器、浏览器和平台。确保您的验证器、浏览器和平台支持 WebAuthn 规格。

8.6.1. 设置

对 2FA 支持的 WebAuthn 支持的设置过程如下：

8.6.1.1. 启用 WebAuthn authenticator 注册

1. 在菜单中，单击 **Authentication**。
2. 点 **所需的 Actions** 选项卡。
3. 点 **Register**。
4. 单击 **Required Action** 下拉列表。
5. 点 **Webauthn Register**。
6. 点 **确定**。

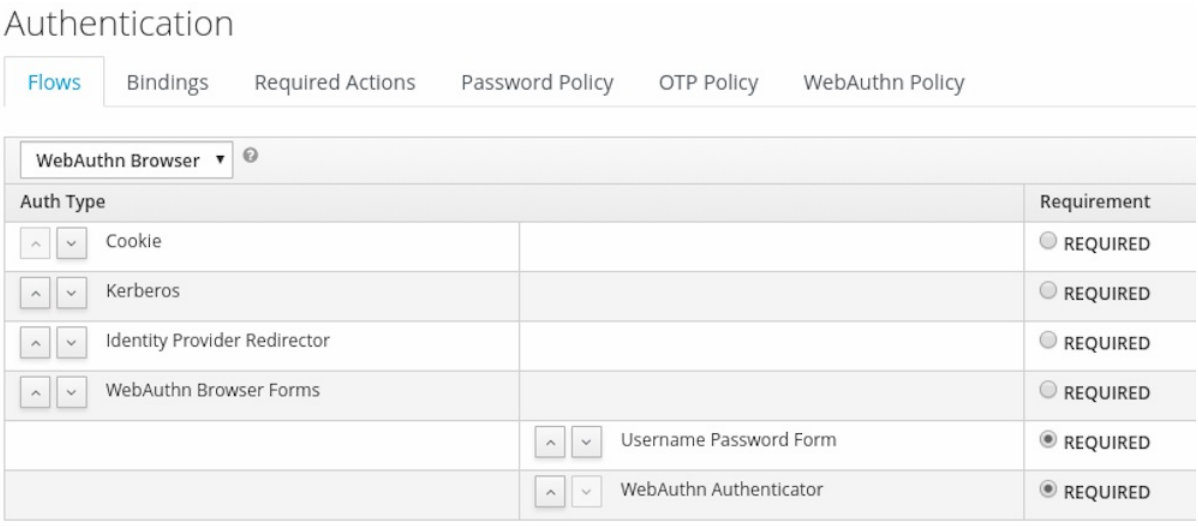
如果您希望所有新用户注册其 WebAuthn 凭据，请标记 **Default Action** 复选框。

8.6.1.2. 将 WebAuthn 身份验证添加到浏览器流

1. 在菜单中，单击 **Authentication**。
2. 单击 **浏览器** 流。
3. 单击 **Copy**，以制作内置 **浏览器** 流的副本。
4. 输入副本的名称。
5. 点 **确定**。
6. 为 **WebAuthn Browser Browser - Conditional OTP**点 *Actions*，点 *Delete*。
7. 点 **Delete**。

如果您需要所有用户的 WebAuthn：

1. 单击 **WebAuthn Browser Forms** 的 *Actions* 链接。
2. 单击 **Add execution**。
3. 点 **Provider** 下拉列表。
4. 单击 **WebAuthn Authenticator**。
5. 点 **Save**。
6. 为 **WebAuthn Authenticator** 点 *REQUIRED*



7. 单击 **Bindings** 选项卡。
8. 点击 **Browser Flow** 下拉列表。
9. 单击 **WebAuthn Browser**。
10. 点 **Save**。

注意

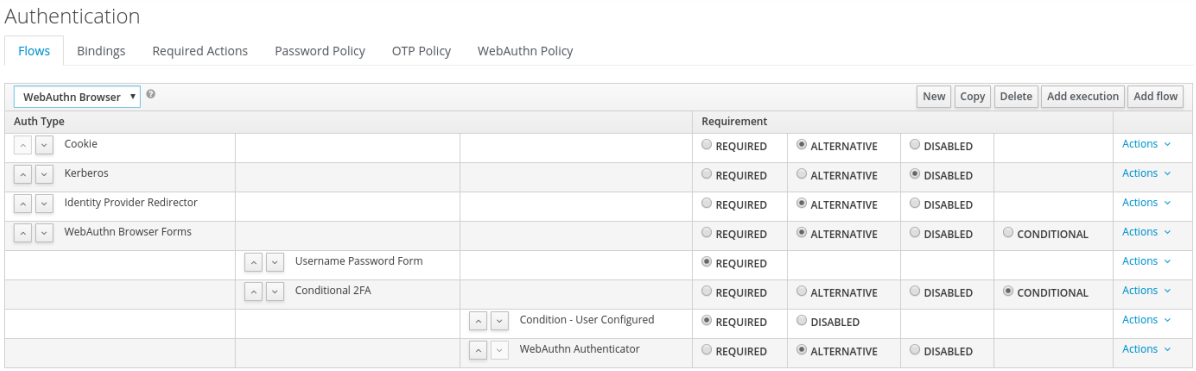
如果用户没有 WebAuthn 凭据，用户必须注册 WebAuthn 凭据。

如果用户仅注册了 WebAuthn，则用户可以通过 WebAuthn 进行登录。因此，**您可以** ：

流程

1. 单击 **WebAuthn Browser Forms** 的 *Actions* 链接。
2. 点 **Add flow**。
3. 在 *Alias* 字段中输入 "Conditional 2FA"。
4. 点 **Save**。
5. 为 **条件 2FA**点 *CONDITIONAL*
6. 点 **Conditional 2FA** 的 *Actions* 链接。

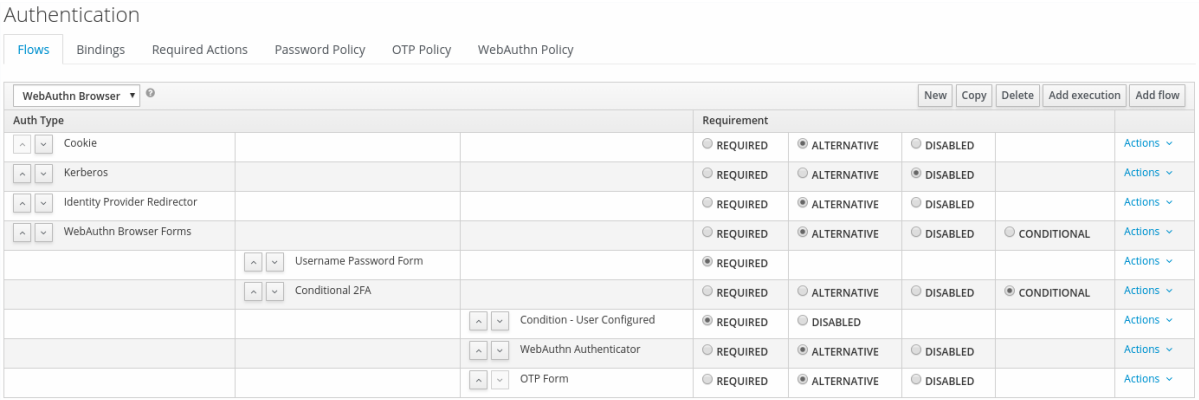
7. 单击 **Add execution**。
8. 点 **Provider** 下拉列表。
9. 点 **Condition - User Configured**。
10. 点 **Save**。
11. 为 **Conditional 2FA**点 *REQUIRED*
12. 点 **Conditional 2FA** 的 *Actions* 链接。
13. 单击 **Add execution**。
14. 点 **Provider** 下拉列表。
15. 单击 **WebAuthn Authenticator**。
16. 点 **Save**。
17. 为 **Conditional 2FA** 点 *ALTERNATIVE*



用户可以选择使用 WebAuthn 和 OTP 作为第二个因素：

流程

1. 为 **Conditional 2FA** 点 *Actions* 链接。
2. 单击 **Add execution**。
3. 点 **Provider** 下拉列表。
4. 点 **ITP Form**。
5. 点 **Save**。
6. 为 **Conditional 2FA** 点 *ALTERNATIVE*



8.6.2. 使用 WebAuthn authenticator 进行身份验证

在注册 WebAuthn authenticator 后，用户执行以下操作：

- 打开登录表单。用户必须使用用户名和密码进行身份验证。
- 用户的浏览器要求用户使用其 WebAuthn authenticator 进行身份验证。

8.6.3. 以管理员身份管理 WebAuthn

8.6.3.1. 管理凭证

Red Hat Single Sign-On 管理 WebAuthn 凭证与来自用户凭证管理的其他 凭证 类似：

- Red Hat Single Sign-On 为用户分配一个必要操作，以便从 **Reset Actions** 列表中删除 WebAuthn 凭证，然后选择 **Webauthn Register**。
- 管理员可以点**删除**来删除 WebAuthn 凭据。
- 管理员可以通过选择 **Show data...** 来查看凭证的数据，如 AAGUID。
- 管理员可以通过在 **User Label** 字段中设置值并保存数据，为凭证设置标签。

8.6.3.2. 管理策略

管理员可以将 WebAuthn 相关操作配置为每个域的 **WebAuthn 策略**。

流程

1. 在菜单中，单击 **Authentication**。
2. 单击 **WebAuthn Policy** 选项卡。
3. 在策略中配置项目（请参见下面的描述）。
4. 点 **Save**。

可配置项及其描述如下：

Configuration	描述
依赖实体名称	可读取的服务器名称作为 WebAuthn Relying unsupported。此项目是必需的，并适用于注册 WebAuthn authenticator。默认设置为 "keycloak"。如需了解更多详细信息，请参阅 WebAuthn 规格。
签名算法	该算法告知 WebAuthn 身份验证器用于 公共密钥凭据的签名算法。红帽单点登录使用公共密钥凭据来签名和验证身份 验证身份。如果没有算法，则默认 ES256 被调整。ES256 是应用于 WebAuthn authenticator 注册的可选配置项。如需了解更多详细信息，请参阅 WebAuthn 规格。
依赖第三方 ID	用于确定公钥凭据范围的 WebAuthn 的 ID。ID 必须是 origin 的有效域。此 ID 是应用于 WebAuthn authenticator 的注册的可选配置项。如果该条目为空，Red Hat Single Sign-On 会修改 Red Hat Single Sign-On 的基本 URL 的主机部分。如需了解更多详细信息，请参阅 WebAuthn 规格。
在测试过程中参考信息	浏览器上的 WebAuthn API 实施(WebAuthn Client)是生成 Attestation 语句的首选方法。这个首选项是应用到 WebAuthn authenticator 注册的可选配置项。如果没有 选项，其行为与选择"none"相同。如需了解更多详细信息，请参阅 WebAuthn 规格。
身份验证器附加	WebAuthn 客户端的 WebAuthn 验证器可接受的附加模式。这个模式是应用到 WebAuthn authenticator 注册的可选配置项。如需了解更多详细信息，请参阅 WebAuthn 规格。
需要相关密钥	选项要求 WebAuthn authenticator 将公钥凭据生成为 Client-side-resident Public Key Credential Source。此选项适用于 WebAuthn authenticator 的注册。如果留空，其行为与选择"No"相同。如需了解更多详细信息，请参阅 WebAuthn 规格。
用户验证要求	选项要求 WebAuthn 验证器确认用户验证。这是一个可选配置项，它应用到 WebAuthn authenticator 的注册，并通过 WebAuthn authenticator 对用户进行身份验证。如果没有 选项，其行为与选择"首选"相同。如需了解更多详细信息，请参阅 用于注册 WebAuthn authenticator 和 WebAuthn 规格 以通过 WebAuthn authenticator 验证用户身份的 WebAuthn 规格。

Configuration	描述
Timeout（超时）	超时值（以秒为单位）用于注册 WebAuthn authenticator 并使用 WebAuthn authenticator 验证用户身份。如果设置为零，其行为取决于 WebAuthn authenticator 的实现。默认值为 0。如需了解更多详细信息，请参阅 用于注册 WebAuthn authenticator 和 WebAuthn 规格 以通过 WebAuthn authenticator 验证用户身份的 WebAuthn 规格。
避免 Same Authenticator 注册	如果启用，Red Hat Single Sign-On 无法重新注册注册的 WebAuthn authenticator。
可接受的 AAGUID	一个 WebAuthn authenticator 必须针对的 AAGUIDs 的白色列表。

8.6.4. attestation 语句验证

在注册 WebAuthn authenticator 时，红帽单点登录会验证 WebAuthn authenticator 生成的 attestation 语句的可信赖性。Red Hat Single Sign-On 需要信任定位者的证书。Red Hat Single Sign-On 使用 Keycloak 信任存储，因此您必须事先将这些证书导入到其中。

要省略这个验证，请禁用此信任存储，或将 WebAuthn 策略的配置项"Attestation Conveyance Preference" 设置为 "none"。

8.6.5. 以用户身份管理 WebAuthn 凭证

8.6.5.1. 注册 WebAuthn authenticator

注册 WebAuthn 验证器的适当方法取决于用户是否已在红帽单点登录上注册帐户。

8.6.5.2. 新用户

如果 **WebAuthn Register** required action 在域中是 **Default Action**，则新用户在第一次登录后必须设置 WebAuthn 安全密钥。

流程

1. 打开登录表单。
2. 点 **Register**。
3. 在表单中填写项目。
4. 点 **Register**。

注册成功后，浏览器会询问用户输入其 WebAuthn authenticator 标签的文本。

8.6.5.3. 现有用户

如果根据第一个示例所示设置了 **WebAuthn Authenticator**，那么当现有用户尝试登录时，需要他们自动注册其 WebAuthn authenticator:

流程

1. 打开登录表单。
2. 在表单中输入项目。
3. 点 **Save**。
4. 单击 **登录**。

注册成功后，用户的浏览器会询问用户输入其 WebAuthn authenticator 标签的文本。

8.6.6. 免密码 WebAuthn 与两个因素一起

Red Hat Single Sign-On 使用 WebAuthn 进行双因素身份验证，但您可以使用 WebAuthn 作为第一因素身份验证。在本例中，具有 **无密码** WebAuthn 凭据的用户可以无需输入密码即可向红帽单点登录身份验证。Red Hat Single Sign-On 可以使用 WebAuthn 作为域上下文和单一身份验证流中的双因素身份验证机制。

管理员通常需要用户为 WebAuthn 免密码身份验证注册的安全密钥满足不同的要求。例如，安全密钥可能需要用户使用 PIN 或具有更强大的证书颁发机构的tests 或安全密钥在安全密钥进行身份验证。

因此，红帽单点登录允许管理员配置单独的 **WebAuthn 免密码策略**。必需的 **Webauthn Register Passwordless** action of type，单独的验证器类型为 **WebAuthn Passwordless Authenticator**。

8.6.6.1. 设置

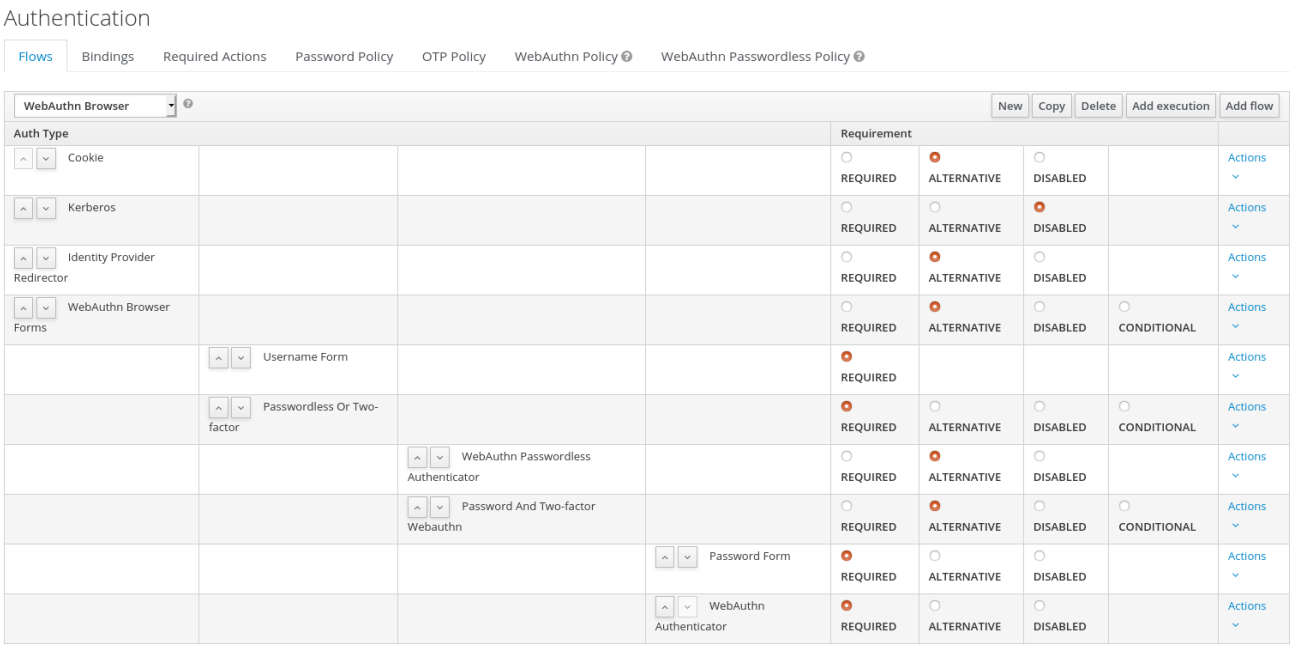
流程

设置 WebAuthn 免密码支持，如下所示：

1. 为 WebAuthn 免密码支持注册新的所需操作。使用 启用 WebAuthn Authenticator Registration 中所述的步骤。注册 **Webauthn Register Passwordless** 操作。
2. 配置策略。您可以使用 管理策略 中描述的步骤和 配置选项。在 **WebAuthn Passwordless Policy** 的选项卡中，执行管理控制台中的配置。通常，安全密钥的要求要高于双因素策略。例如，在配置免密码策略时，您可以将 **User Verification Requirement** 设置为 **Required**。
3. 配置身份验证流程。使用 **WebAuthn Browser** 流，如 Adding WebAuthn Authentication to a Browser Flow 所述。按如下方式配置流：
    - **WebAuthn Browser Forms** 子流包含 **Username Form**，作为第一个验证器。删除默认的 **Username Password Form** authenticator 并添加 **Username Form** authenticator。此操作要求用户提供用户名作为第一步。
    - 需要一个必需的子流，它可以命名为 **less Or Two-factor**，例如：此子流表示用户可以通过无密码的 WebAuthn 凭据或使用双因素身份验证进行身份验证。
    - 流包含 **WebAuthn Passwordless Authenticator** 作为第一个替代方案。
    - 第二个替代项是名为 **Password 的子流和双因素 Webauthn**，例如：此子流包含 **密码表单** 和 **WebAuthn Authenticator**。

流的最终配置类似：

免密码流



现在，您可以把 **WebAuthn Register Passwordless** 作为 Red Hat Sign-On （已称 Red Hat Sign-On）的所需操作来添加来进行测试。在第一次身份验证期间，用户必须使用密码和第二因素 WebAuthn 凭据。如果用户使用 WebAuthn Passwordless 凭证，则用户需要提供密码和第二因素 WebAuthn 凭据。

8.6.7. LoginLess WebAuthn

Red Hat Single Sign-On 使用 WebAuthn 进行双因素身份验证，但您可以使用 WebAuthn 作为第一因素身份验证。在这种情况下，具有 **无密码** WebAuthn 凭据的用户可以向 Red Hat Single Sign-On 进行身份验证，而无需提交登录名或密码。Red Hat Single Sign-On 可在域上下文中使用 WebAuthn 作为无登录/密码和双因素身份验证机制。

管理员通常需要用户为 WebAuthn 登录身份验证注册的安全密钥满足不同的要求。无登录身份验证要求用户对安全密钥进行身份验证（例如，使用 PIN 代码或指纹），并且与无登录凭证关联的加密密钥实际存储在安全密钥中。并非所有安全密钥都符合这种要求。如果您的设备支持"用户验证"和"重要密钥"，请联系您的安全关键厂商。请参阅支持的安全密钥。

Red Hat Single Sign-On 允许管理员以允许无登录身份验证的方式配置 **WebAuthn Passwordless Policy**。请注意，无登录身份验证只能使用 **WebAuthn Passwordless Policy** 和 **WebAuthn Passwordless** 凭证进行配置。可以在同一域上配置 WebAuthn 登录身份验证和 WebAuthn 免密码身份验证，但将共享相同的策略 **WebAuthn Passwordless 策略**。

8.6.7.1. 设置

流程

设置 WebAuthnless 支持，如下所示：

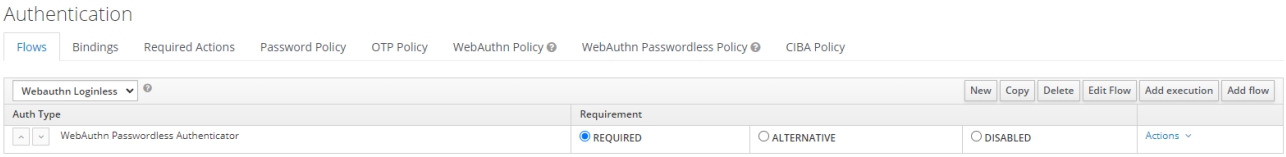
1. 为 WebAuthn 免密码支持注册新的所需操作。使用 启用 WebAuthn Authenticator Registration 中所述的步骤。注册 **Webauthn Register Passwordless** 操作。

2. 配置 **WebAuthn 免密码策略**。在 **WebAuthn Passwordless Policy** 选项卡的 Admin Console **Authentication** 部分中执行配置。在为策略配置无登录情况时，将 **User Verification Requirement** 设置为 **required**，**Require Resident Key** 设置为 **Yes**。请注意，由于没有专用登录策略，因此无法混合使用用户验证=no/resident key=no 和无登录情况（用户验证=yes/resident key=yes）。存储容量通常受到安全性键的限制，这意味着您无法将许多驻留在您的安全密钥中保存。

3. 配置身份验证流程。创建一个新的身份验证流，添加"WebAuthn Passwordless"执行，并将执行的 Requirement 设置为 **Required**

流的最终配置类似：

LoginLess flow



现在，您可以将所需的操作 **WebAuthn Register Passwordless** 添加到 Red Hat Single Sign-On 中已知的用户，以进行测试。配置所需操作的用户将需要进行身份验证（例如使用用户名/密码），然后系统将提示您注册要用于无登录身份验证的安全密钥。

8.6.7.2. 特定于厂商的 remarks

8.6.7.2.1. 兼容性检查列表

使用 Red Hat Single Sign-On 进行无登录身份验证需要安全密钥才能满足以下功能

- FIDO2 合规性：不要与 FIDO/U2F 混淆
- 用户验证：安全密钥验证用户身份的功能（第一条发现您的安全密钥的人员能够无密码验证和免密码验证）
- 常驻键：用于存储登录和与客户端应用程序关联的加密密钥的功能

8.6.7.2.2. Windows Hello

要使用 Windows Hello based credentials for Red Hat Single Sign-On 进行身份验证，请配置 **WebAuthn Passwordless Policy** 的 **Signature Algorithms** 设置以包含 **RS256** 值。请注意，一些浏览器不允许在私有窗口中访问平台安全密钥（如 Windows Hello）。

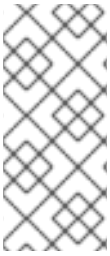
8.6.7.2.3. 支持的安全密钥

以下安全密钥已通过 Red Hat Single Sign-On 测试了成功进行无登录身份验证：

- Windows Hello (Windows 10 21H1/21H2)
- yubico Yubikey 5 NFC
- Feitian ePass FIDO-NFC

8.7. 恢复代码（恢复代码）

您可以通过在验证流中添加"Recovery Authentication Code Form"作为双因素验证器来配置恢复代码以实现双因素身份验证。有关配置此验证器的示例，请参阅 WebAuthn。



注意

RecoveryCodes **是技术预览**，没有被完全支持。此功能默认为禁用。

要使用 **-Dkeycloak.profile=preview** 或 **-Dkeycloak.profile.feature.recovery_codes=enabled** 来启用服务器。如需了解更多详细信息，请参阅 配置文件。

8.8. 条件流中的条件

如 执行要求 所述，*Condition* 执行只能包含在 *Conditional* 子流中。如果所有 *Condition* 执行都评估为 true，*则条件* 子流充当 *Required*。您可以在 *条件* 子流中处理下一次执行。如果 *Conditional* sub-flow 评估为 false 中包含某些执行，则整个子流被视为 *Disabled*。

8.8.1. 可用条件

condition - 用户角色

此执行能够确定该用户是否具有由 *User role* 字段定义的角色。如果用户具有所需的角色，则执行被视为 true，其他执行会被评估。管理员必须定义以下字段：

Alias

描述执行的名称，它将显示在验证流中。

用户角色

用户应当必须要执行这个流的角色。要指定应用角色，其语法为 **appname.approle** （例如 **myapp.myrole**）。

condition - 用户配置

这会检查是否为用户配置了流中的其他执行。Execution requirements 部分包括 OTP 表单的示例。

condition - 用户属性

这将检查用户是否已设置 required 属性。可能存在相关的输出，这意味着用户不应具有 属性。User Attributes 部分演示了如何添加自定义属性。您可以提供这些字段：

Alias

描述执行的名称，它将显示在验证流中。

属性名称

要检查的属性的名称。

预期属性值

属性中的预期值。

negate 输出

您可以设置输出。换句话说，属性不应存在。

8.8.2. 在条件流中明确拒绝/允许访问

您可以允许或拒绝访问条件流中的资源。两个验证器 **拒绝访问**，**并允许** 根据条件对资源进行访问。

允许访问

验证器始终将成功进行身份验证。这个验证器不可配置。

拒绝访问

访问将始终被拒绝。您可以定义错误消息，它将向用户显示。您可以提供这些字段：

Alias

描述执行的名称，它将显示在验证流中。

错误消息

将向用户显示的错误消息。错误消息可以作为特定消息或属性提供，以便将其用于本地化。（例如 "*您没有角色*". "*admin*'." , *my-property-deny* in messages 属性中）将默认消息留空来用作属性 **access-denied**。

以下示例说明了如何拒绝对没有角色 **role1** 的所有用户访问，并显示由属性 **deny-role1** 定义的错误消息。这个示例包括 **Condition - User Role** 和 **Deny Access** executions。

浏览器流

condition - 用户角色配置

Deny Access 的配置非常容易。您可以指定任意 Alias 和必需信息，如下所示：

最后一件事是在 login theme **messages_en.properties** （对于英语）中用错误消息定义属性：

```
deny-role1 = You do not have required role!
```

第 9 章 集成身份提供程序

Identity Broker 是使用身份提供程序连接服务提供商的中间服务。身份代理创建与外部身份提供程序的关系，以使用提供程序的身份访问服务提供商所公开的内部服务。

从用户的角度来看，身份代理提供了一种以用户为中心的集中式方式，以管理安全域和域的身份。您可以用来自身份提供程序的一个或多个身份链接帐户，或者根据它们的身份信息创建帐户。

身份提供程序派生自用来验证并向用户发送身份验证和授权信息的特定协议。它可以是：

- 社交供应商，如 Facebook、Google 或 Twitter。
- 用户需要访问您的服务的业务合作伙伴。
- 您希望集成基于云的身份服务。

通常，Red Hat Single Sign-Ons 身份提供程序在以下协议中：

- **SAML v2.0**
- **OpenID Connect v1.0**
- **OAuth v2.0**

9.1. BROKERING 概述

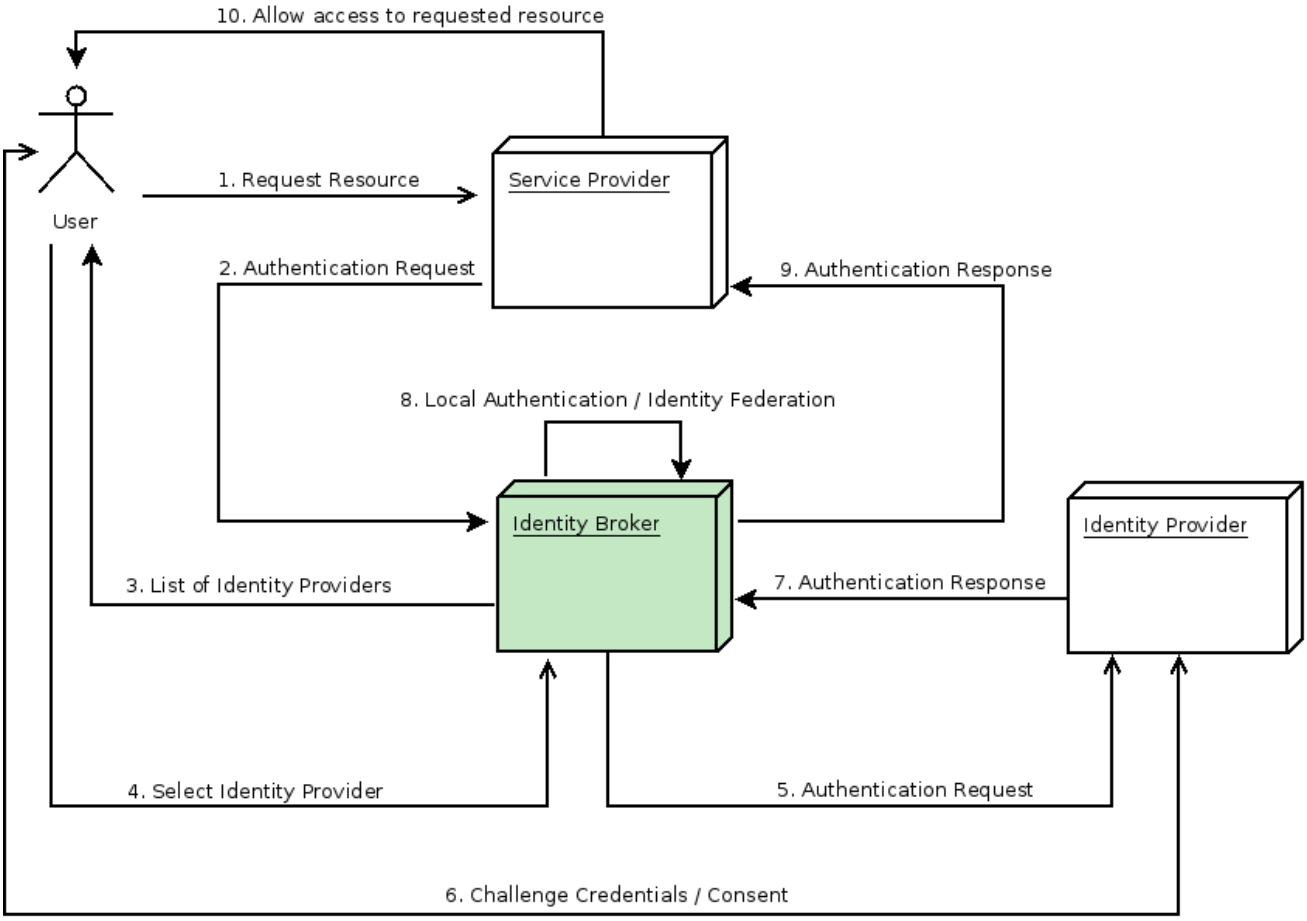
当将 Red Hat Single Sign-On 用作身份提供程序时，Red Hat Single Sign-On 不会强制用户提供他们在特定域中进行身份验证的凭证。Red Hat Single Sign-On 显示可以进行身份验证的身份提供商列表。

如果配置默认身份提供程序，Red Hat Single Sign-On 会将用户重定向到默认供应商。

注意

不同的协议可能需要不同的身份验证流程。Red Hat Single Sign-On 支持的所有身份提供程序都使用以下流：

身份代理流



1. 未经身份验证的用户在客户端应用程序中请求受保护的资源。
2. 客户端应用将用户重定向到 Red Hat Single Sign-On 以进行身份验证。
3. Red Hat Single Sign-On 将显示域中配置的身份提供程序列表。
4. 用户通过点击其按钮或链接来选择其中一个身份提供程序。
5. Red Hat Single Sign-On 会向目标身份提供程序发出身份验证请求，并将用户重定向到身份提供程序的登录页面。管理员已经为 Admin Console 的身份供应商设置了连接属性和其他配置选项。
6. 用户提供凭证或同意与身份提供程序进行身份验证。
7. 在身份验证成功后，用户通过身份验证响应重新重定向到 Red Hat Single Sign-On。通常，响应中包含一个安全令牌，供红帽单点登录用于信任身份提供程序的身份验证并检索用户信息。
8. Red Hat Single Sign-On 会检查来自身份提供程序的响应是否有效。如果有效，Red Hat Single Sign-On 导入并创建用户（如果用户尚不存在）。如果令牌不包含该信息，则 Red Hat Single Sign-On 可能会向身份提供程序寻求更多信息。这个行为是 *身份联合*。如果用户已存在，Red Hat Single Sign-On 可能会要求用户将从身份提供程序中返回的身份与现有帐户连接。这个行为是 *链接 的帐户*。使用红帽单点登录，您可以配置 *帐户链接并在* 第一个登录流 中指定它。在这一步中，Red Hat Single Sign-On 会验证用户并发出令牌来访问服务提供商中请求的资源。
9. 当用户验证时，Red Hat Single Sign-On 通过发送之前在本地身份验证期间发布的令牌，将用户重定向到服务提供商。
10. 服务提供商从 Red Hat Single Sign-On 接收令牌，并允许访问受保护的资源。

这种流程的变体是可能的。例如，客户端应用程序可以请求特定的身份提供程序，而不是显示它们的列表，也可以设置 Red Hat Single Sign-On 来强制用户在重新生成身份前提供其他信息。

在身份验证过程结束时，Red Hat Single Sign-On 会将令牌附加到客户端应用程序。客户端应用与外部身份提供程序分离，因此它们无法看到客户端应用的协议或如何验证用户身份。供应商仅需要了解 Red Hat Single Sign-On。

9.2. 默认身份提供程序

Red Hat Single Sign-On 可以重定向到身份提供程序，而不是显示登录表单。启用这个重定向：

流程

1. 在菜单中，单击 **Authentication**。
2. 单击 **浏览器** 流。
3. 从下拉列表中选择 **Identity Provider Redirector**。
4. **将默认身份提供程序** 设置为您要将用户重定向到的身份供应商。

如果 Red Hat Single Sign-On 没有找到配置的默认身份提供程序，则会显示登录表单。

此验证器负责处理 **kc_idp_hint** 查询参数。如需更多信息，请参阅客户端推荐的身份提供程序 部分。

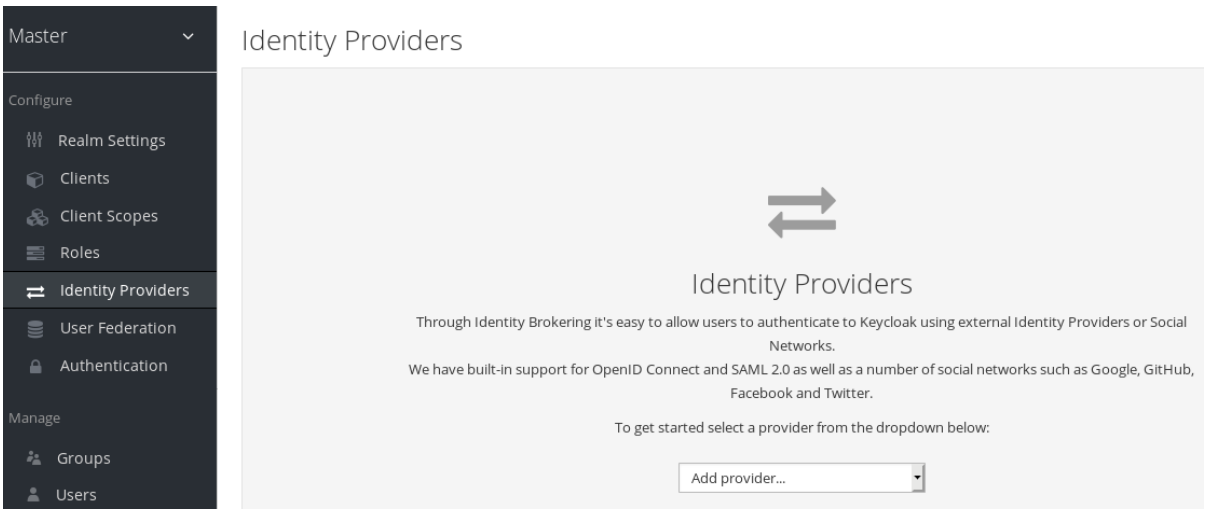
9.3. 常规配置

身份代理配置的基础是身份提供程序(IDP)。Red Hat Single Sign-On 为每个域创建身份提供程序，并默认为每个应用程序启用它们。在登录应用时，realm 中的用户可以使用任何注册的身份提供商。

流程

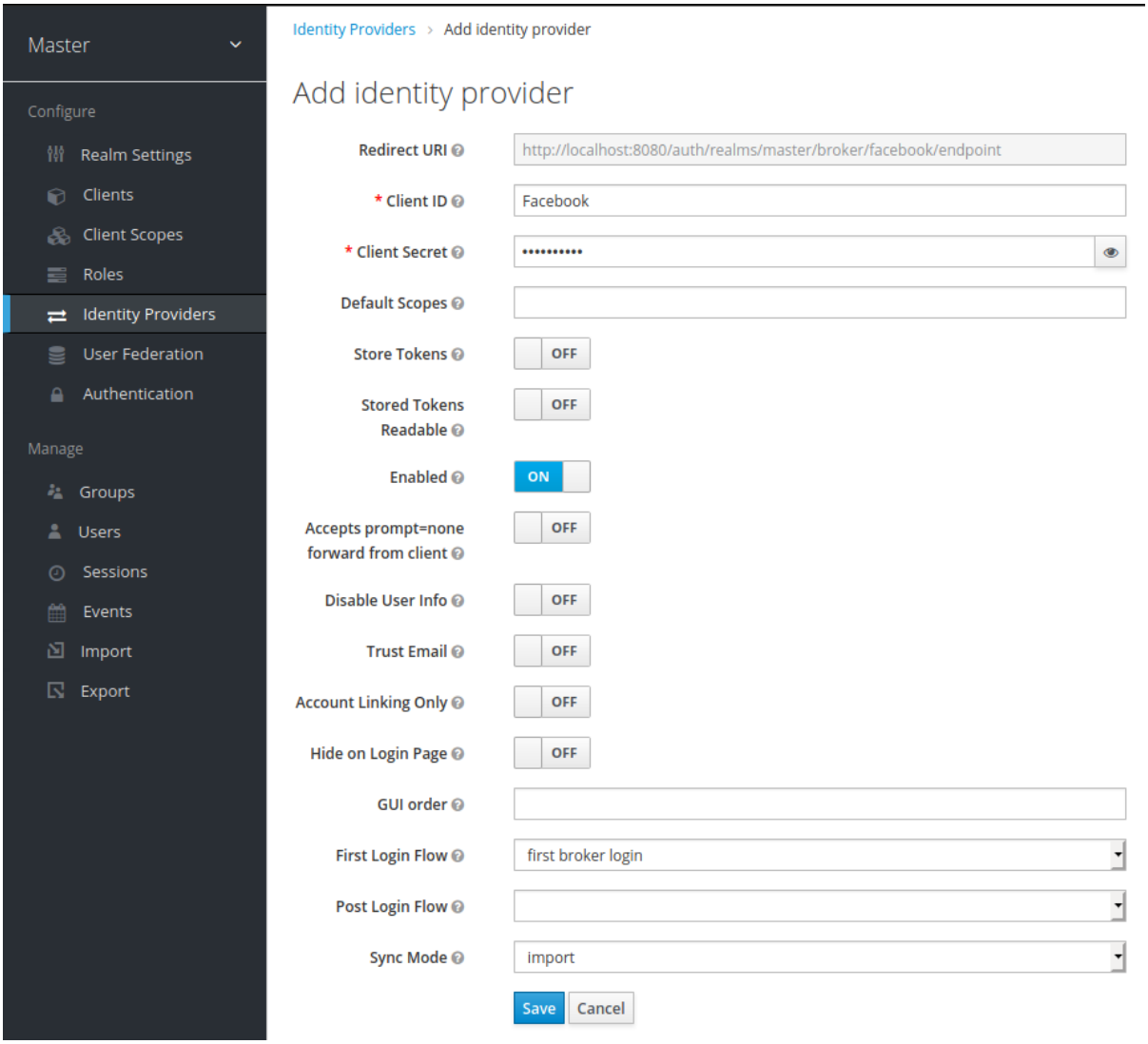
1. 在菜单中，单击 **Identity Providers**。

身份供应商



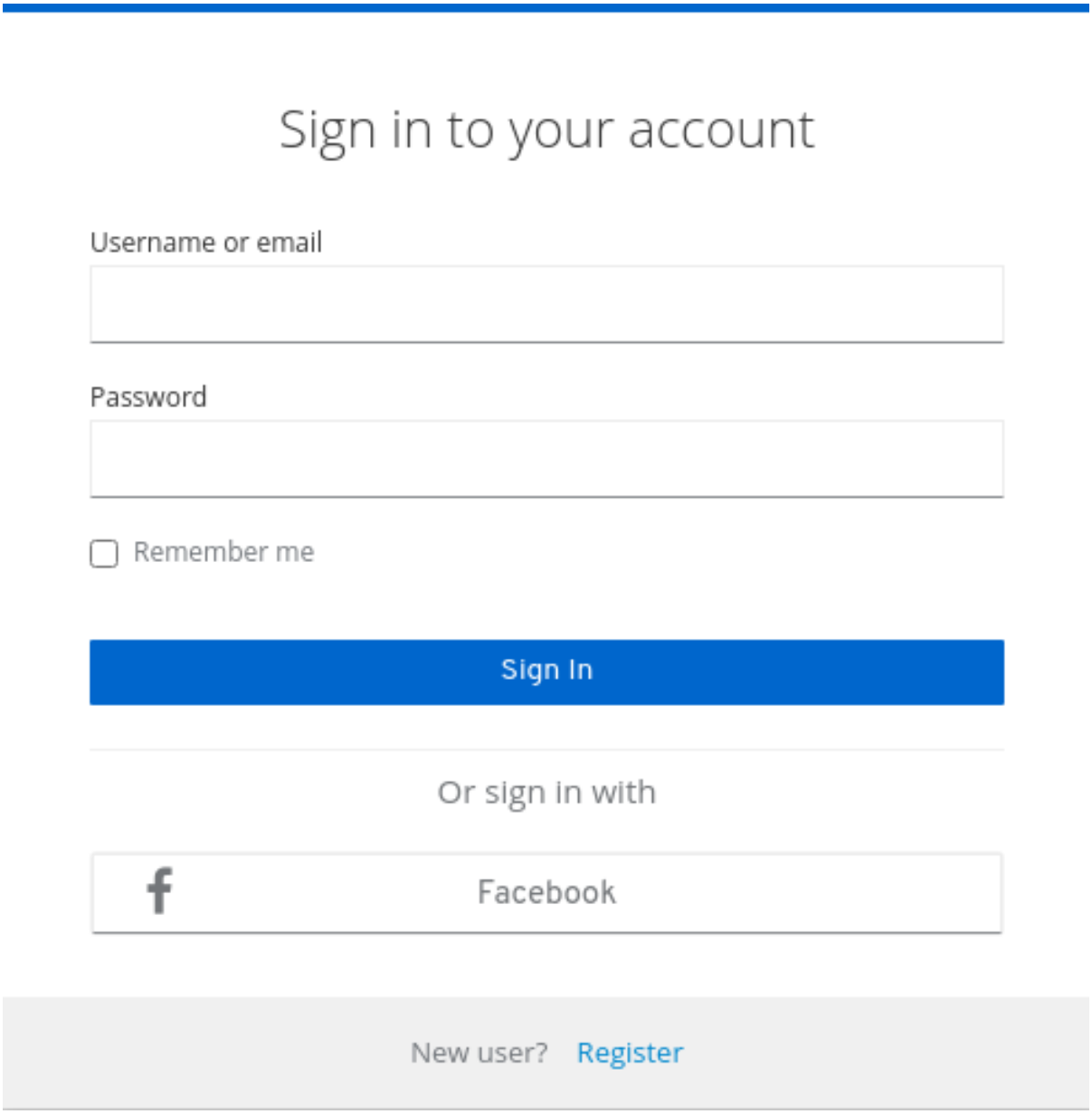
2. 从 **Add provider** 列表中，选择您要添加的身份提供程序。Red Hat Single Sign-On 显示您选择的身份提供程序的配置页面。

添加 facebook 身份提供程序



当您配置身份提供程序时，身份提供程序会作为选项出现在 Red Hat Single Sign-On 登录页面中。您可以将自定义图标放在每个身份提供程序的登录屏幕中。如需更多信息，请参阅自定义 图标。

IDP 登录页面



社交

社交提供程序在域中启用社交身份验证。通过红帽单点登录，用户可以使用社交网络帐户登录您的应用程序。支持的提供程序包括 Twitter、Facebook、Google、LinkedIn、Instagram、Microsoft、 PPal、 Openshift v3、GitHub、GitLab、Bitbucket 和 Stack Overflow。

基于协议

基于协议的供应商依赖特定的协议来验证和授权用户。使用这些供应商，您可以连接到符合特定协议的任何身份提供程序。Red Hat Single Sign-On 支持 SAML v2.0 和 OpenID Connect v1.0 协议。您可以根据这些开放标准配置并代理任何身份提供程序。

虽然每种类型的身份提供程序都有其配置选项，但所有共享一个通用配置。可用的配置选项如下：

表 9.1. 常见配置

Configuration	描述

Configuration	描述
Alias	别名是身份提供程序的唯一标识符，并引用内部身份提供程序。Red Hat Single Sign-On 使用别名为 OpenID Connect 协议构建重定向 URI，该协议需要重定向 URI 或回调 URL 与身份提供程序通信。所有身份提供程序都必须有一个别名。别名示例包括 **facebook**、**google** 和 **idp.acme.com**。
Enabled	切换提供程序 ON 或 OFF。
在登录页中隐藏	当 **ON** 时，Red Hat Single Sign-On 就不会显示此提供程序作为登录页面上的登录选项。客户端可以使用 URL 中的 'kc_idp_hint' 参数请求此提供程序。
仅限客户链接	当 **ON** 时，红帽单点登录将现有帐户链接到此提供程序。此提供程序无法登录用户，Red Hat Single Sign-On 不会将这个提供程序显示为登录页面上的选项。
存储令牌	当 **ON** 时，Red Hat Single Sign-On 将令牌从身份提供程序存储。
存储的令牌可读	**ON** 时，用户可以检索存储的身份提供程序令牌。此操作也适用于代理客户端 *级* 角色 *读取令牌*。
信任电子邮件	当 **ON** 时，Red Hat Single Sign-On 信任来自身份提供程序的电子邮件地址。如果域需要电子邮件验证，则从此身份提供程序登录的用户不需要执行电子邮件验证过程。
GUI 顺序	登录页面中可用身份提供程序的排序顺序。
第一个登录流	当用户第一次登录 Red Hat Single Sign-On 时，身份验证流 Red Hat Single Sign-On 会触发。
后登录流	当用户使用外部身份提供程序登录后，身份验证流 Red Hat Single Sign-On 会触发。
同步模式	通过映射程序从身份提供程序更新用户信息的策略。在选择 **传统的** 时，红帽单点登录将使用当前的行为。**导入** 不会更新用户数据，并尽可能 **强制更新** 用户数据。如需更多信息，请参阅身份提供程序映射程序。

9.4. 社交身份提供程序

社交身份提供程序可以将身份验证委派给可信的社交媒体帐户。红帽单点登录包括支持社交网络，如 Google、Facebook、Twitter、GitHub、LinkedIn、Microsoft 和 Stack Overflow。

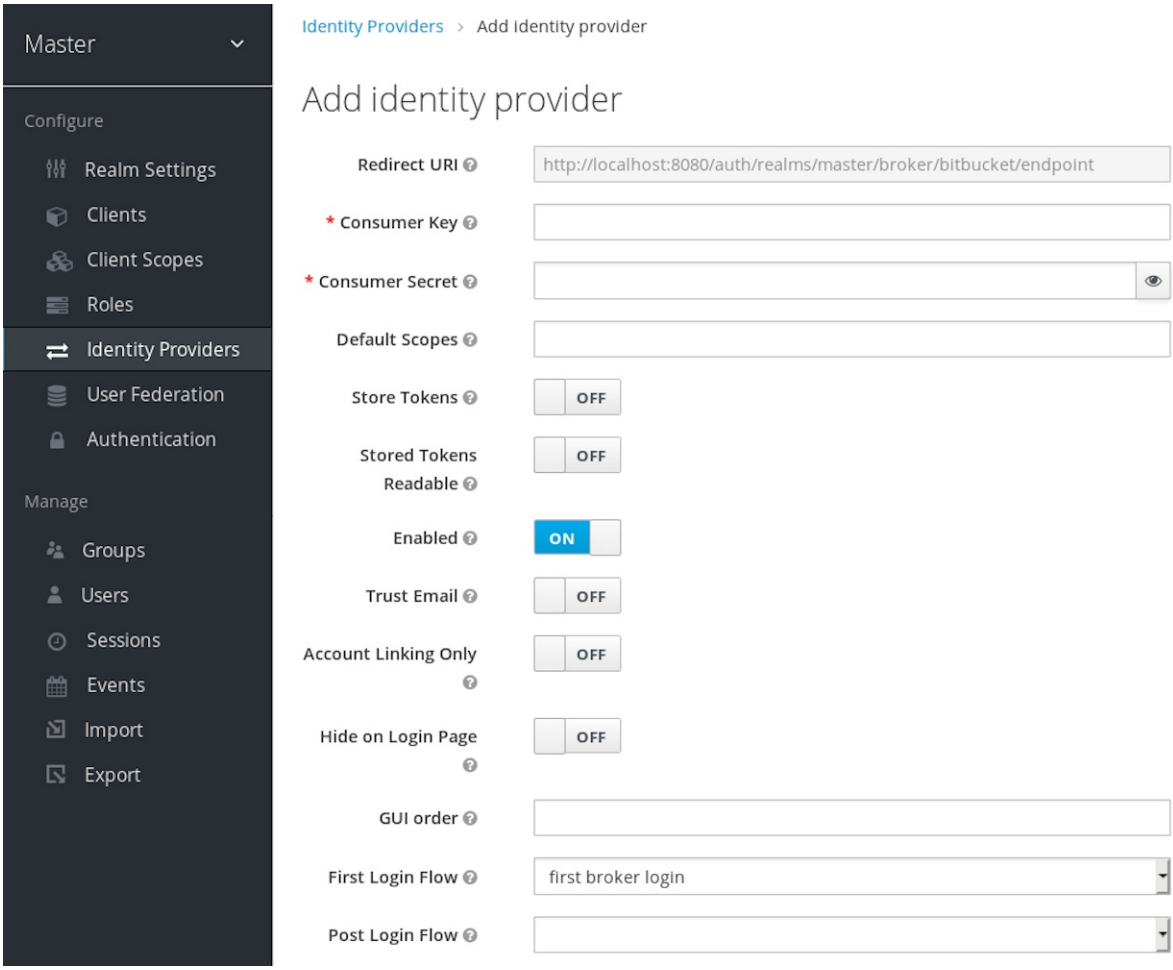
9.4.1. Bitbucket

要使用 Bitbucket 登录，请执行以下步骤。

流程

1. 在菜单中，单击 **Identity Providers**。
2. 从 **Add provider** 列表中，选择 **Bitbucket**。

添加身份提供程序



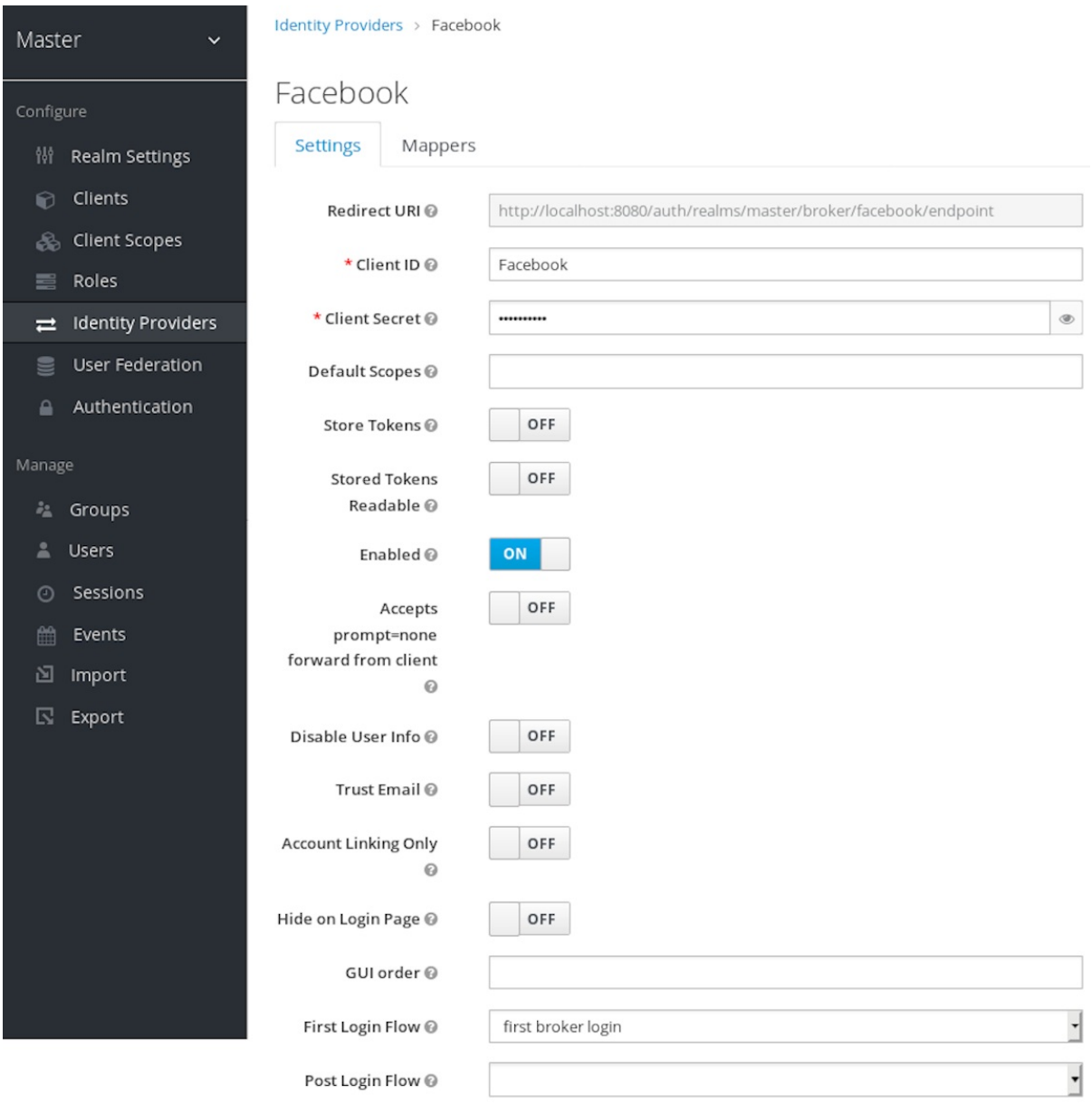
3. 将 **Redirect URI** 的值复制到您的剪贴板。
4. 在一个单独的浏览器标签页中，在 Bitbucket 云进程上执行 OAuth。当您点击 **Add Consumer** 时：
   a. 将 **重定向 URI** 的值粘贴到 **Callback URL** 字段中。
   b. 确保您在 **帐户** 部分中选择 **Email** 和 **Read**，以允许您的应用程序阅读电子邮件。
5. 请注意，创建消费者时的 **Key** 和 **Secret** 值 Bitbucket 会显示。
6. 在 Red Hat Single Sign-On 中，将 **Key** 的值粘贴到 **Client ID** 字段中。
7. 在 Red Hat Single Sign-On 中，将 **Secret** 的值粘贴到 **Client Secret** 字段中。
8. 点 **Save**。

9.4.2. Facebook

流程

1. 在菜单中，单击 **Identity Providers**。
2. 从 **Add provider** 列表中，选择 **Facebook**。Red Hat Single Sign-On 显示 Facebook 身份提供程序的配置页面。

添加身份提供程序



3. 将 **Redirect URI** 的值复制到您的剪贴板。
4. 在单独的浏览器选项卡中，按照 Facebook 开发人员指南 的说明在 Facebook 中创建项目和客户端。
   a. 确保您的应用程序是一个网站类型应用程序。
   b. 在 Facebook **Website** 设置块的 **Site URL** 中输入重定向 URI 的值。
   c. 确保该应用是公共的。

5. 在 Red Hat Single Sign-On 的 **Client ID** 和 **Client Secret** 字段中输入来自 Facebook 的 **Client ID** 和 **Client Secret** 值。

6. 在 **Default Scopes** 字段中输入所需的范围。默认情况下，Red Hat Single Sign-On 使用 **电子邮件** 范围。有关 Facebook 范围的更多信息，请参阅 Graph API。

Red Hat Single Sign-On 将配置集请求发送到 **graph.facebook.com/me?fields=id,name,email,first_name,last_name**。响应仅包含 id、name、mail、first_name 和 last_name 字段。要从 Facebook 配置集中获取附加字段，请添加对应的范围，并在 **Additional user 的 profile 字段配置**选项 字段中添加字段名称。

9.4.3. GitHub

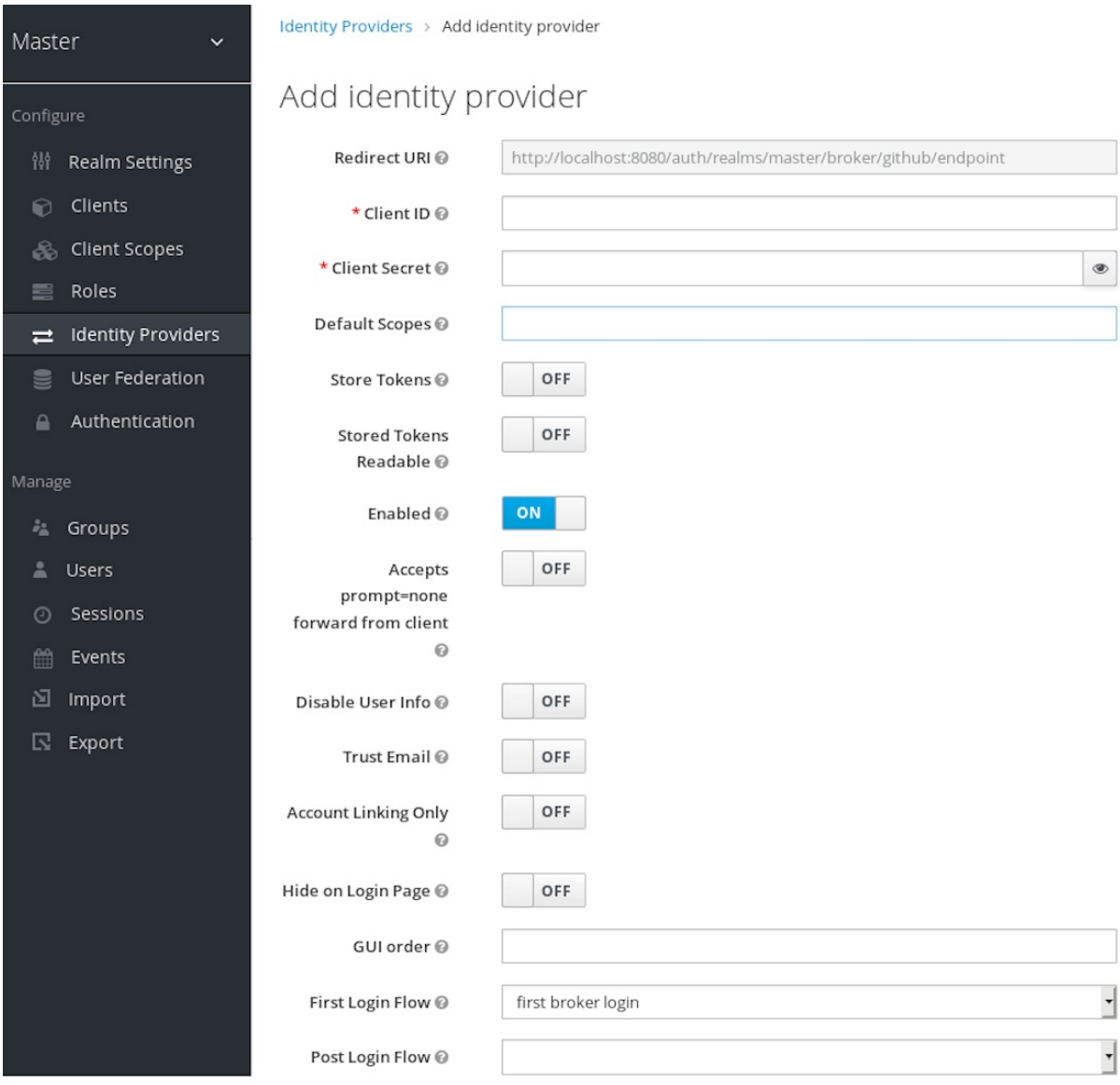
要使用 Github 登录，请执行以下步骤。

流程

1. 在菜单中，单击 **Identity Providers**。

2. 从 **Add provider** 列表中，选择 **Github**。

添加身份提供程序



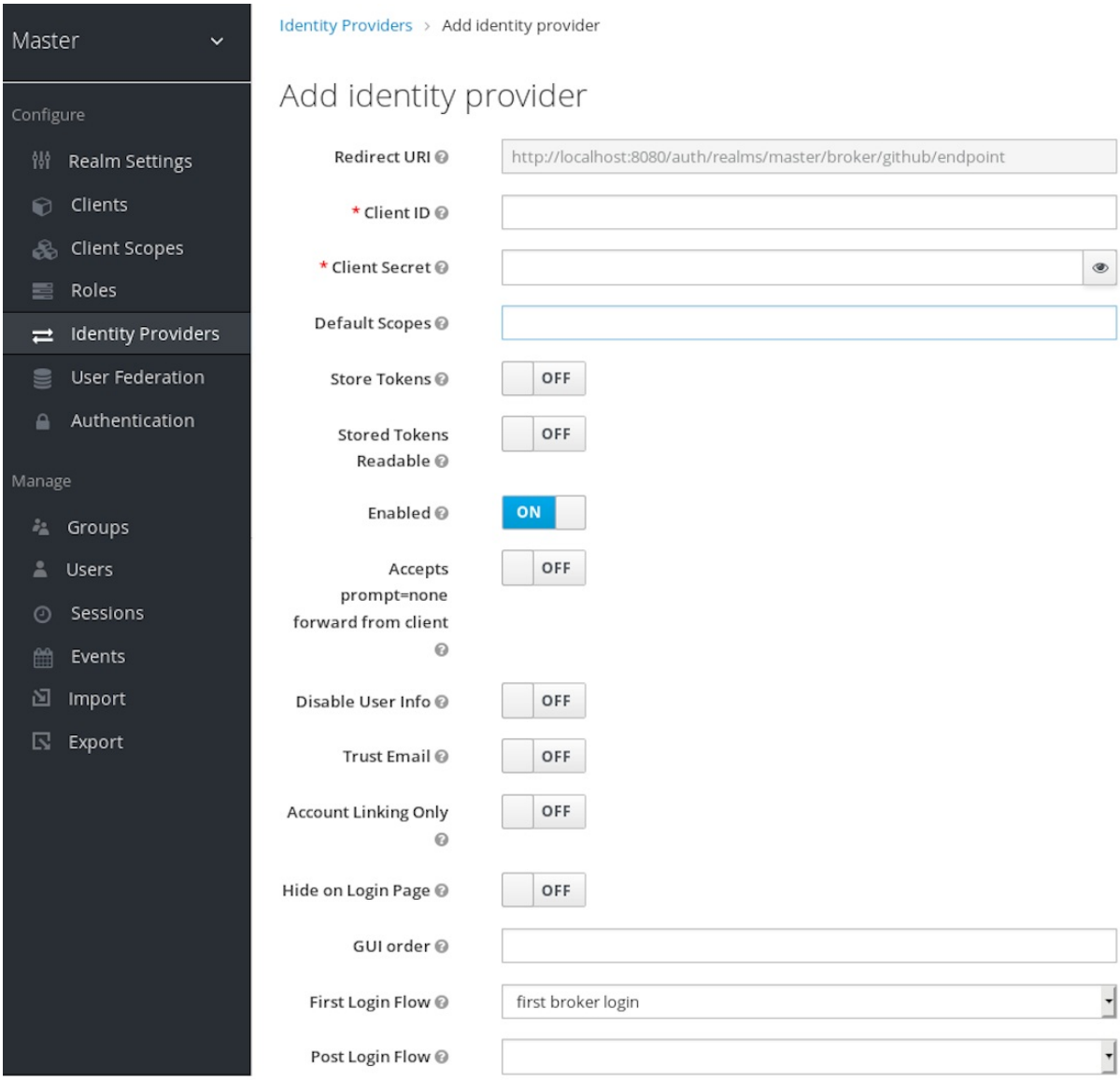
3. 将 **Redirect URI** 的值复制到您的剪贴板。
4. 在一个单独的浏览器标签页中，创建 OAUTH app。
   a. 在创建应用程序时，输入 **Redirect URI** 的值到 **Authorization callback URL** 字段。
   b. 请注意 OAUTH 应用管理页面中的客户端 ID 和客户端 secret。
5. 在 Red Hat Single Sign-On 中，将 **Client ID** 的值粘贴到 **Client ID** 字段中。
6. 在 Red Hat Single Sign-On 中，将 **Client Secret** 的值粘贴到 **Client Secret** 字段中。
7. 点 **Save**。

9.4.4. GitLab

流程

1. 在菜单中，单击 **Identity Providers**。
2. 从 **Add provider** 列表中，选择 **GitLab**。

添加身份提供程序



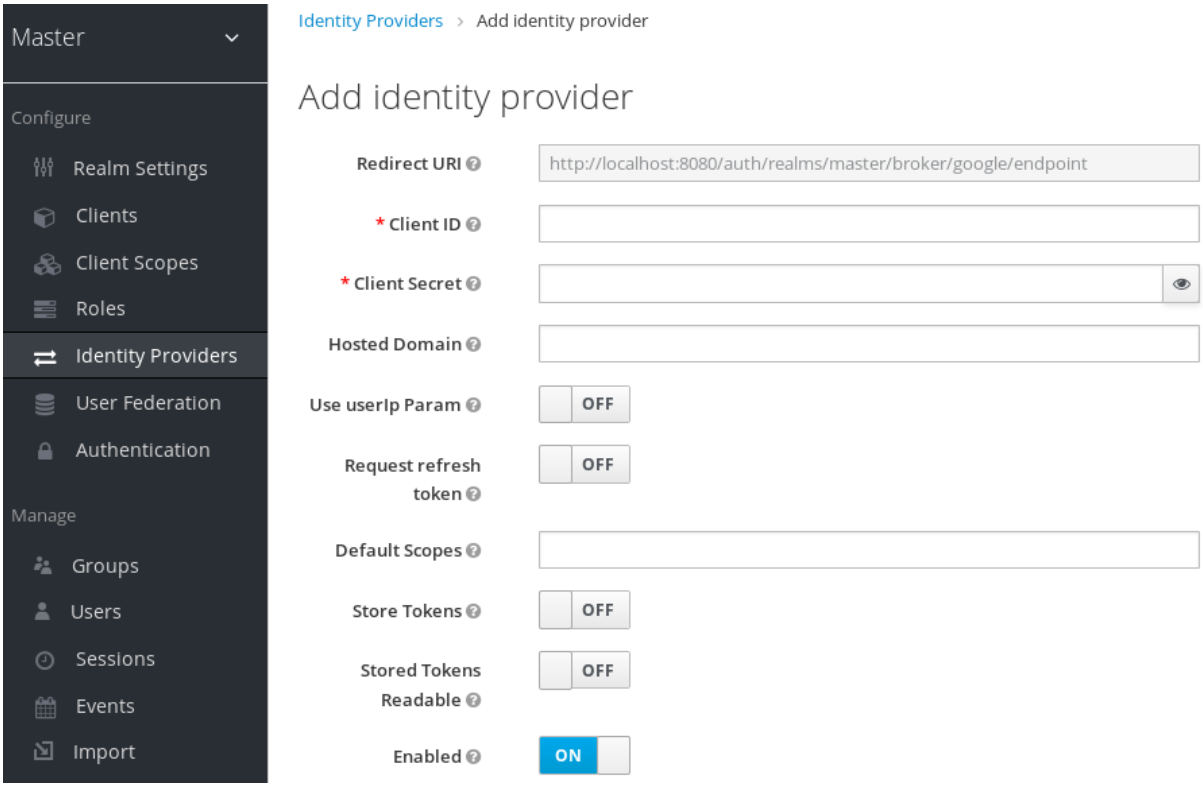
3. 将 **Redirect URI** 的值复制到您的剪贴板。
4. 在一个单独的浏览器选项卡中，添加新的 GitLab 应用程序。
   a. 使用剪贴板中的 **重定向 URI** 作为 **重定向 URI**。
   b. 保存应用程序时，记录下 **Client ID** 和 **Client Secret**
5. 在 Red Hat Single Sign-On 中，将 **Client ID** 的值粘贴到 **Client ID** 字段中。
6. 在 Red Hat Single Sign-On 中，将 **Client Secret** 的值粘贴到 **Client Secret** 字段中。
7. 点 **Save**。

9.4.5. Google

流程

1. 在菜单中，单击 **Identity Providers**。
2. 从 **Add provider** 列表中，选择 **Google**。

添加身份提供程序



3. 在一个单独的浏览器选项卡中，打开 Google Cloud Platform 控制台。
4. 在 Google 应用的 Google 仪表板中，点 **OAuth consent 屏幕** 菜单。创建一个同意屏幕，确保用户类型同意屏幕是外部的。
5. 在 Google 仪表板中：
   a. 点 **Credentials** 菜单。
   b. 单击 **CREATE CREDENTIALS** - **OAuth Client ID**。

c. 从 **Application type** 列表中，选择 **Web 应用程序**。

d. 点 **Create**。

e. 记录 **您的客户端 ID** 和 **您的客户端 Secret**。

6. 在 Red Hat Single Sign-On 中，将您的客户端 **ID** 的值粘贴到 **Client ID** 字段中。

7. 在 Red Hat Single Sign-On 中，将您的 **Client Secret** 的值粘贴到 **Client Secret** 字段中。

8. 在 **Default Scopes** 字段中输入所需的范围。默认情况下，Red Hat Single Sign-On 使用以下范围： **openid 配置集 电子邮件**。如需 Google 范围列表，请参阅 OAuth Playground。

9. 若要仅限制您 GSuite 组织成员可以进行访问，在 **Hosted Domain** 字段中输入 G Suite 域。

10. 点 **Save**。

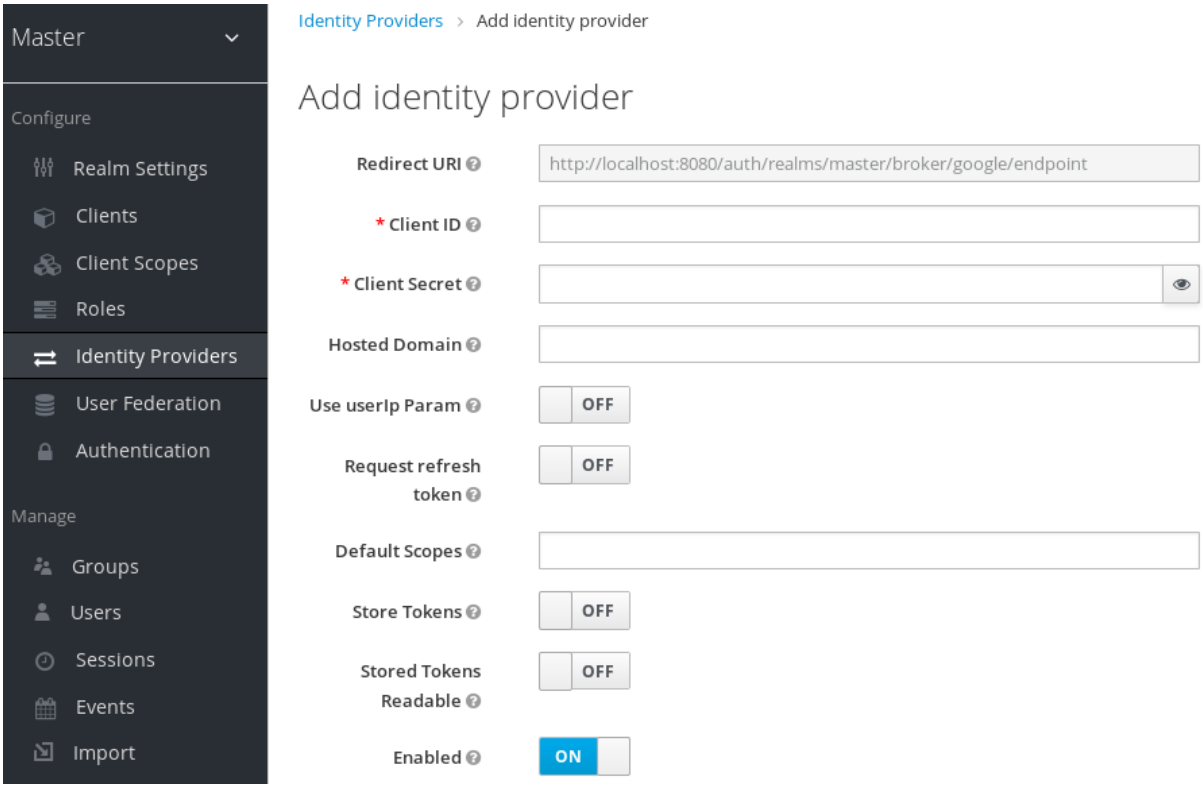
9.4.6. LinkedIn

流程

1. 在菜单中，单击 **Identity Providers**。

2. 从 **Add provider** 列表中，选择 **LinkedIn**。

添加身份提供程序



3. 将 **Redirect URI** 的值复制到您的剪贴板。

4. 在一个单独的浏览器标签页中，创建一个应用程序。

a. 创建应用程序后，点 **Auth** 选项卡。

b. 在 **Authorized redirect URLs for your app**字段中输入 **Redirect URI** 的值。

c. 记录 **您的客户端 ID** 和 **您的客户端 Secret**。

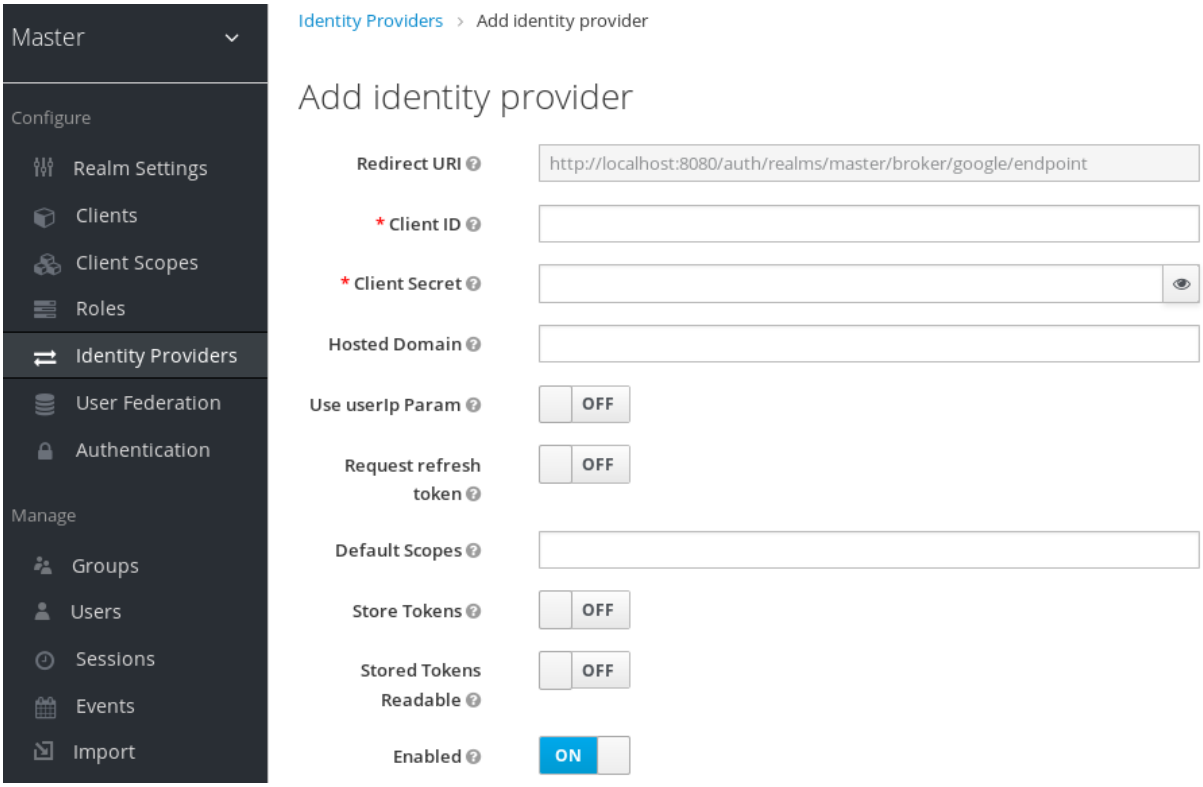
5. 在 Red Hat Single Sign-On 中，将 **Client ID** 的值复制到 **Client ID**。
6. 在 Red Hat Single Sign-On 中，将 **Client Secret** 的值粘贴到 **Client Secret** 字段中。
7. 点 **Save**。

9.4.7. Microsoft

流程

1. 在菜单中，单击 **Identity Providers**。
2. 从 **Add provider** 列表中，选择 **Microsoft**。

添加身份提供程序



3. 将 **Redirect URI** 的值复制到您的剪贴板。
4. 在单独的浏览器选项卡中，创建一个 Microsoft 应用程序。

a. 单击 **Add URL**，将重定向 URL 添加到 Microsoft 应用。

b. 请注意 **应用程序 ID** 和 **应用程序 Secret**。

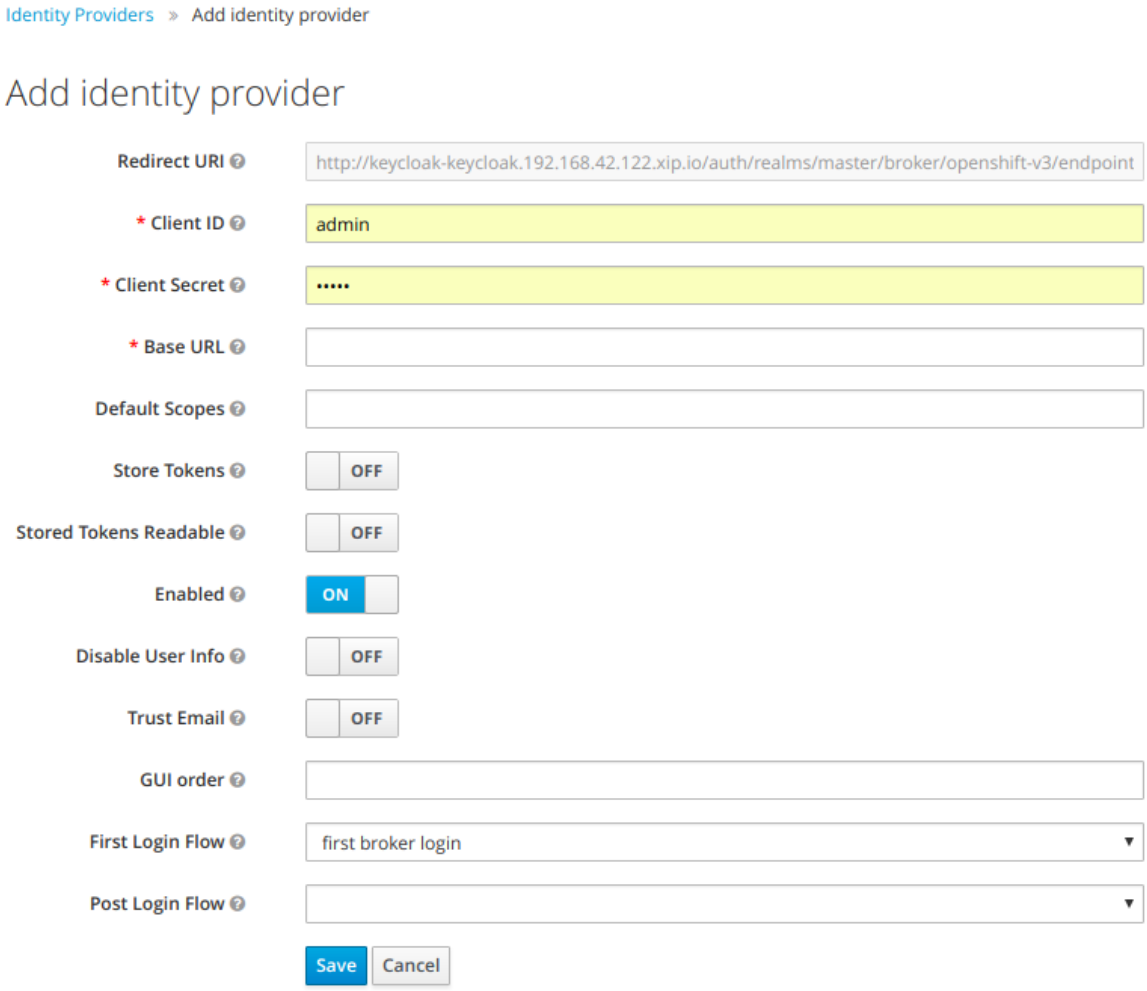
5. 在 Red Hat Single Sign-On 中，将 **Application ID** 的值粘贴到 **Client ID** 字段中。
6. 在 Red Hat Single Sign-On 中，将 **Application Secret** 的值粘贴到 **Client Secret** 字段中。
7. 点 **Save**。

9.4.8. OpenShift 3

流程

1. 在菜单中，单击 **Identity Providers**。
2. 从 **Add provider** 列表中，选择 **Openshift**。

添加身份提供程序



3. 将 **Redirect URI** 的值复制到您的剪贴板。
4. 使用 **oc** 命令行工具注册您的客户端。

```
$ oc create -f <(echo '
kind: OAuthClient
apiVersion: v1
metadata:
 name: kc-client (1)
secret: "..." (2)
redirectURIs:
 - "http://www.example.com/" (3)
grantMethod: prompt (4)
')
```

(1) OAuth 客户端 **的名称**。在向 <openshift_master> /oauth/ **authorize 和 <*openshift_master*>** / **oauth/ token** 发出请求时*传递为* ***client_*** **id** request 参数。

2 Red Hat Single Sign-On 用于 **client_secret** 请求参数的 **secret**。

3 对 < ***openshift_master>*** **; /oauth/authorize 和 &** lt ***;openshift_master>*** **; /oauth/token** 的请求中指定的 **redirect_uri** 参数必须等于 **redirectURIs** 中的某一 URI。您可以从身份提供程序屏幕中的 **Redirect URI** 字段获取它

4 **grantMethod** Red Hat Single Sign-On 使用 来确定当此客户端请求令牌但用户未授予访问权限时的操作。

1. 在 Red Hat Single Sign-On 中，将 **Client ID** 的值复制到 **Client ID**。
2. 在 Red Hat Single Sign-On 中，将 **Client Secret** 的值粘贴到 **Client Secret** 字段中。
3. 点 **Save**。

9.4.9. OpenShift 4

先决条件

1. 安装 jq。
2. 在容器中配置的 **X509_CA_BUNDLE**，并设置为 **/var/run/secrets/kubernetes.io/serviceaccount/ca.crt**。

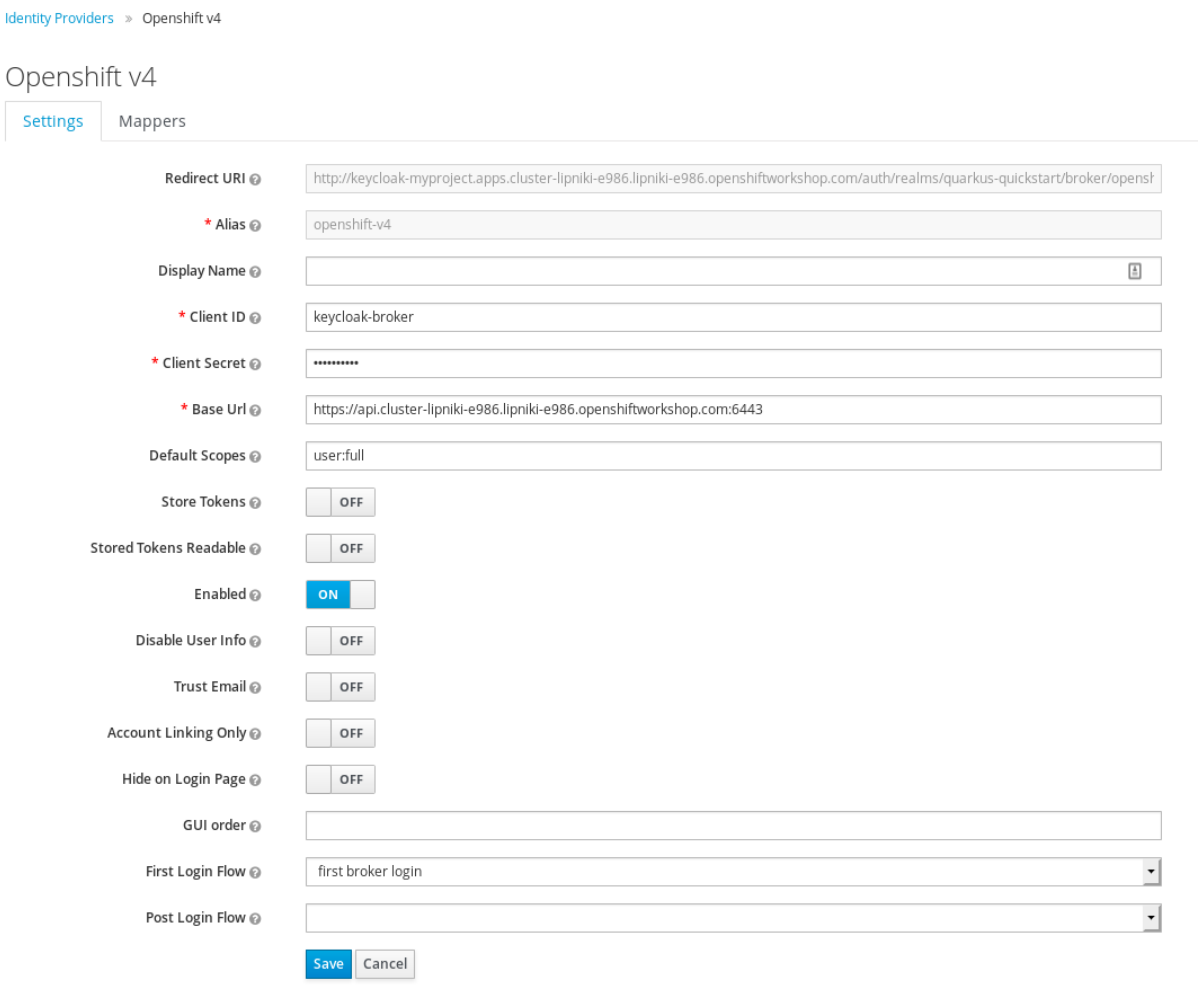
流程

1. 在命令行中运行以下命令并记录 OpenShift 4 API URL 输出。

```
curl -s -k -H "Authorization: Bearer $(oc whoami -t)" \https://<openshift-user-facing-api-url>/apis/config.openshift.io/v1/infrastructures/cluster | jq ".status.apiServerURL"
```

2. 在 Red Hat Single Sign-On 菜单中，单击 **Identity Providers**。
3. 从 **Add provider** 列表中，选择 **Openshift**。

添加身份提供程序



4. 将 **Redirect URI** 的值复制到您的剪贴板。

5. 使用 **oc** 命令行工具注册您的客户端。

```
$ oc create -f <(echo '
kind: OAuthClient
apiVersion: oauth.openshift.io/v1
metadata:
 name: keycloak-broker (1)
secret: "..." (2)
redirectURIs:
 - "<copy pasted Redirect URI from OpenShift 4 Identity Providers page>" (3)
grantMethod: prompt (4)
')
```

(1) OAuth 客户端 **的名称**。在向 <openshift_master> /oauth/ **authorize 和 *<openshift_master>* / oauth/ token** 发出请求时*传递为 client_* **id** request 参数。 **name** 参数在 **OAuthClient** 对象和 Red Hat Single Sign-On 配置中需要是相同的。

(2) Red Hat Single Sign-On 用于 **client_secret** 请求参数的 **secret**。

(3) 对 < ***openshift_master>* /oauth/authorize 和 &** lt ***;openshift_master>* /oauth/token** 的请求中指定的 **redirect_uri** 参数必须等于 **redirectURIs** 中的某一 URI。正确配置的最简单方式是从 Red Hat Single Sign-On OpenShift 4 Identity Provider 配置页面(**Redirect URI** 字段)中复制并粘贴它。

(4) **grantMethod** Red Hat Single Sign-On 使用 来确定当此客户端请求令牌但用户未授予访问权限时的操作。

1. 在 Red Hat Single Sign-On 中，将 **Client ID** 的值复制到 **Client ID**。
2. 在 Red Hat Single Sign-On 中，将 **Client Secret** 的值粘贴到 **Client Secret** 字段中。
3. 点 **Save**。

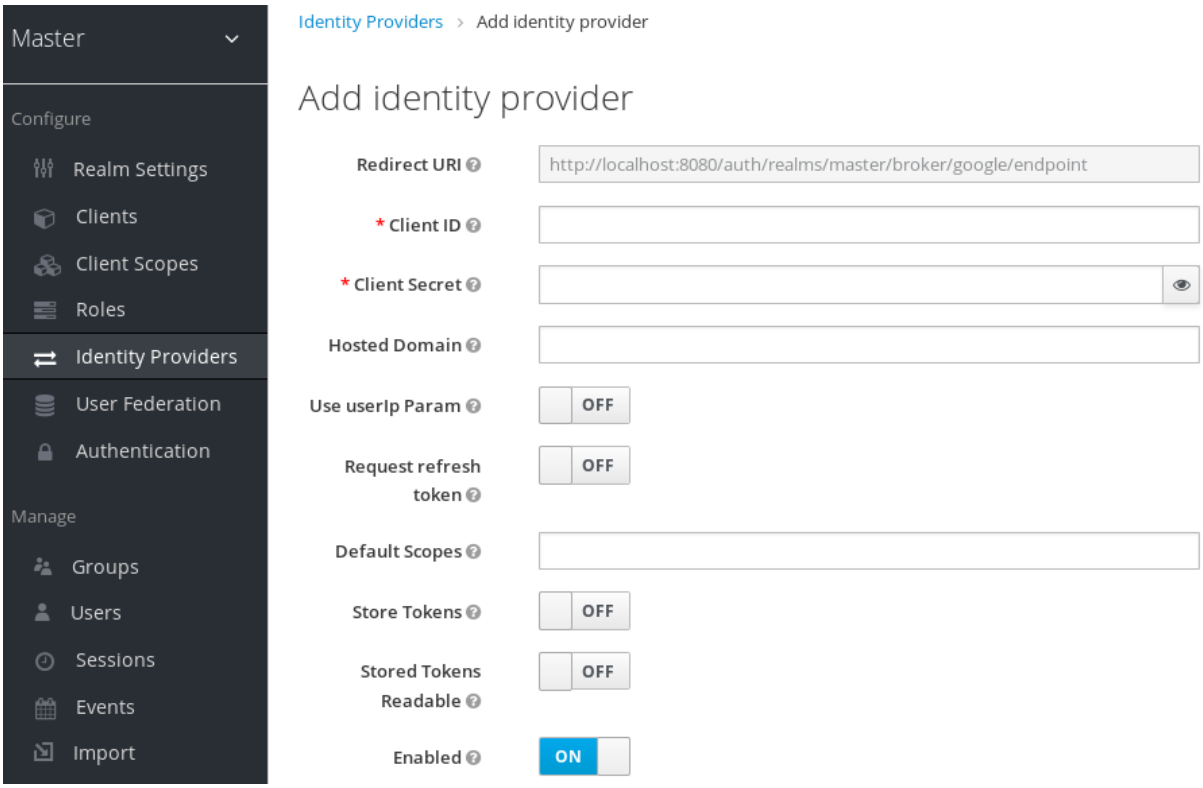
如需更多信息，请参阅 OpenShift 官方文档。

9.4.10. PayPal

流程

1. 在菜单中，单击 **Identity Providers**。
2. 从 **Add provider** 列表，选择 **PayPal**。

添加身份提供程序



3. 将 **Redirect URI** 的值复制到您的剪贴板。
4. 在一个单独的浏览器选项卡中，打开 PayPal Developer 应用程序区域。
   a. 点击 **Create App** 创建 PayPal 应用程序。
   b. 记下客户端 ID 和客户端 Secret。单击 **Show** 链接来查看 secret。
   c. 确保选中了 **与 PayPal 连接**。
   d. 将 return **URL** 字段的值设置为 Red Hat Single Sign-On 中的 **Redirect URI**。
5. 在 Red Hat Single Sign-On 中，将 **Client ID** 的值复制到 **Client ID**。
6. 在 Red Hat Single Sign-On 中，将 **Client Secret** 的值粘贴到 **Client Secret** 字段中。

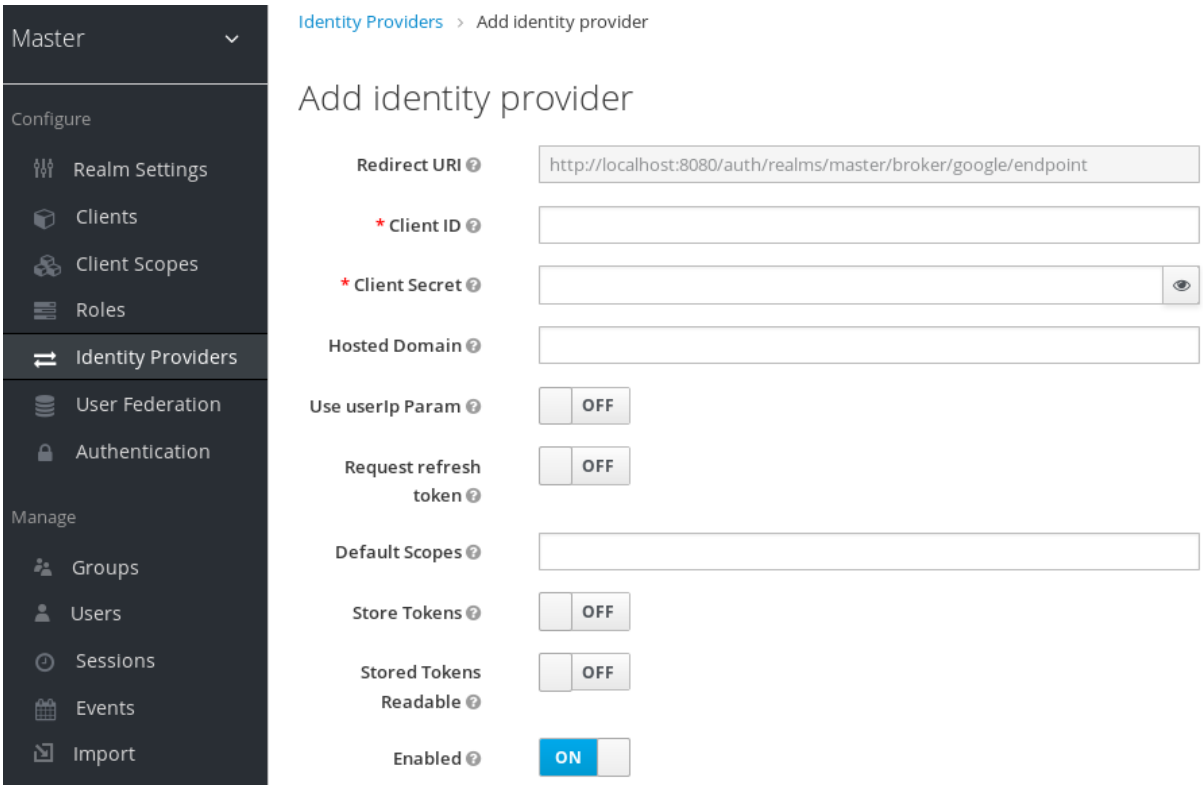
7. 点 **Save**。

9.4.11. 堆栈溢出

流程

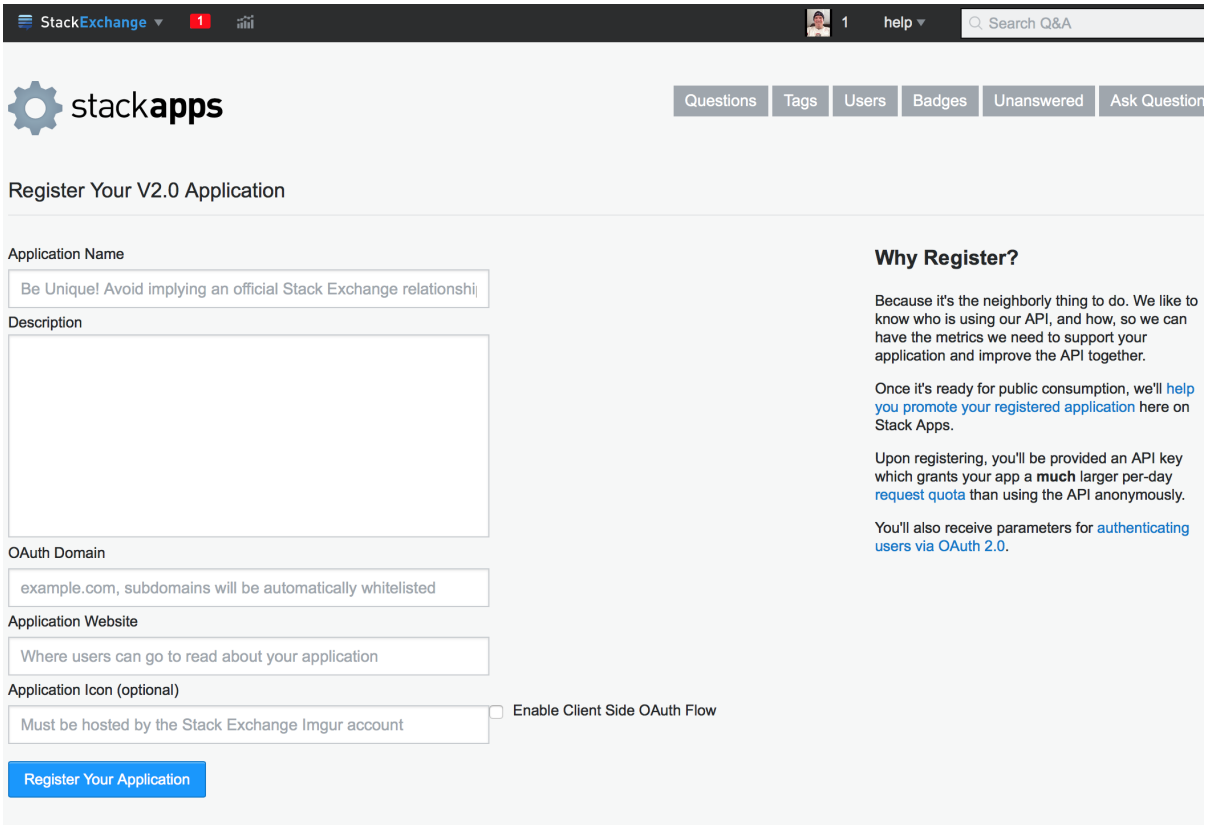
1. 在菜单中，单击 **Identity Providers**。
2. 从 **Add provider** 列表中，选择 **Stack Overflow**。

添加身份提供程序



3. 在一个单独的浏览器选项卡中，在 Stack Apps 上注册您的应用程序。

注册应用程序

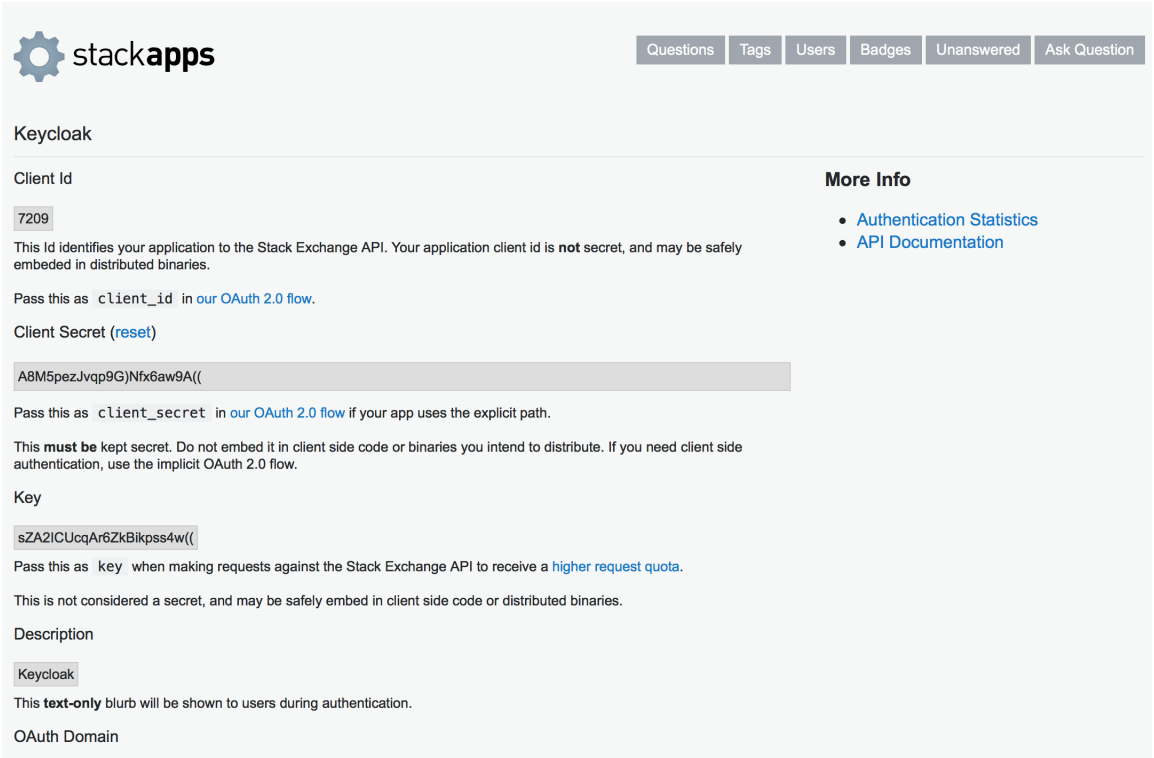


a. 在 **Application Name** 字段中输入应用程序名称。

b. 在 **OAuth Domain** 字段中，输入 OAuth 域。

c. 点 **Register Your Application**。

设置



4. 记下 **客户端 ID 和客户端 Secret**。

5. 在 Red Hat Single Sign-On 中，将 **Client ID** 的值复制到 **Client ID**。

6. 在 Red Hat Single Sign-On 中，将 **Client Secret** 的值粘贴到 **Client Secret** 字段中。
7. 点 **Save**。

9.4.12. Twitter

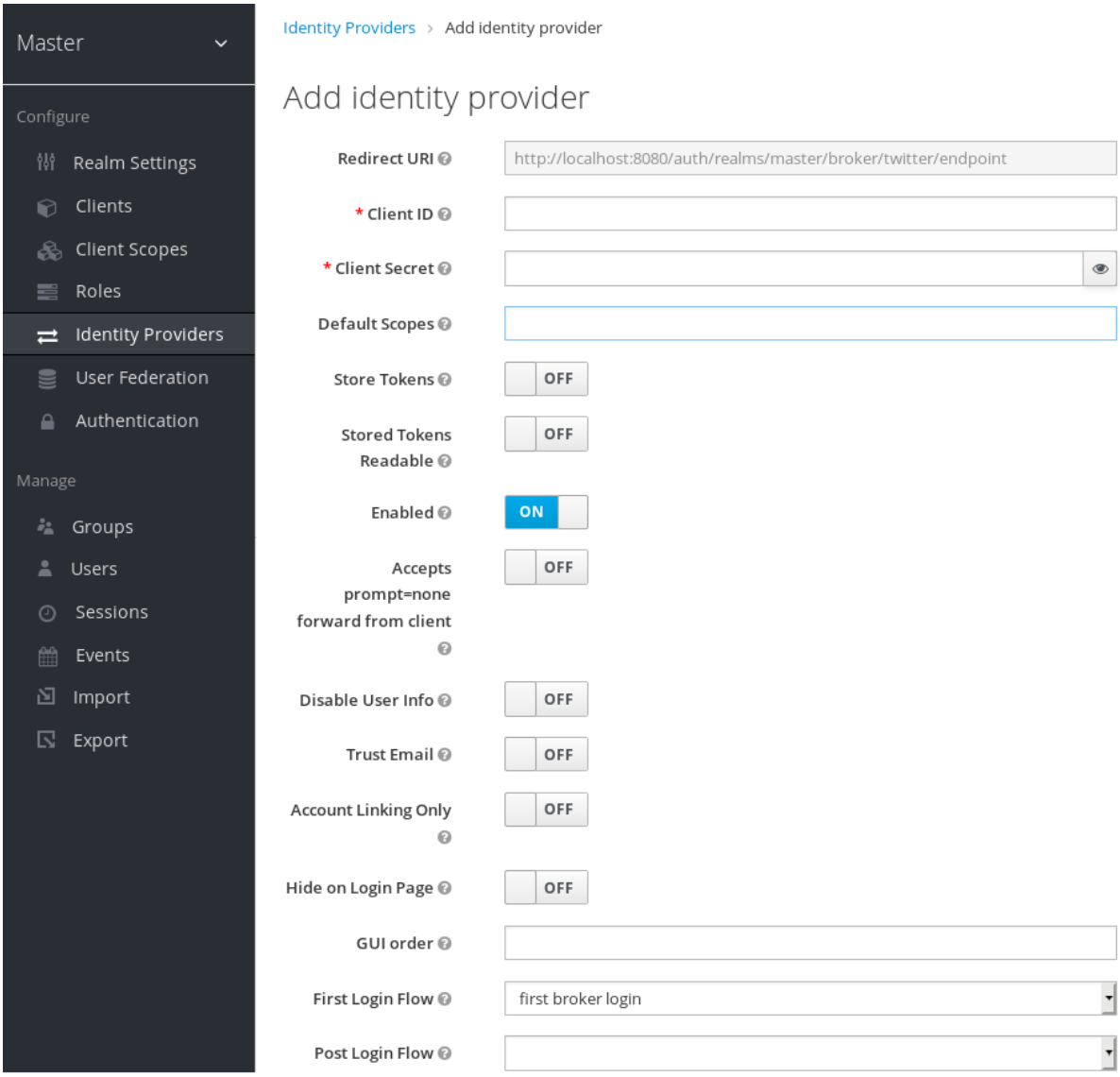
先决条件

1. Twitter 开发人员帐户。

流程

1. 在菜单中，单击 **Identity Providers**。
2. 从 **Add provider** 列表中，选择 **Twitter**。

添加身份提供程序



3. 将 **Redirect URI** 的值复制到您的剪贴板。
4. 在单独的浏览器选项卡中，在 Twitter 应用管理中创建应用。
   a. 为名称和描述输入任何值。

b. **Website** 的值可以是除 **localhost** 外的任何有效 URL。

c. 将 **重定向 URL** 的值粘贴到 **Callback URL** 字段中。

d. 在创建 Twitter 应用时，记录下 **Keys and Access Tokens** 中的 **Consumer Key** 和 **Consumer Secret** 的值。

5. 在 Red Hat Single Sign-On 中，将 **Consumer Key** 的值粘贴到 **Client ID** 字段中。

6. 在 Red Hat Single Sign-On 中，将 **Consumer Secret** 的值粘贴到 **Client Secret** 字段中。

7. 点 **Save**。

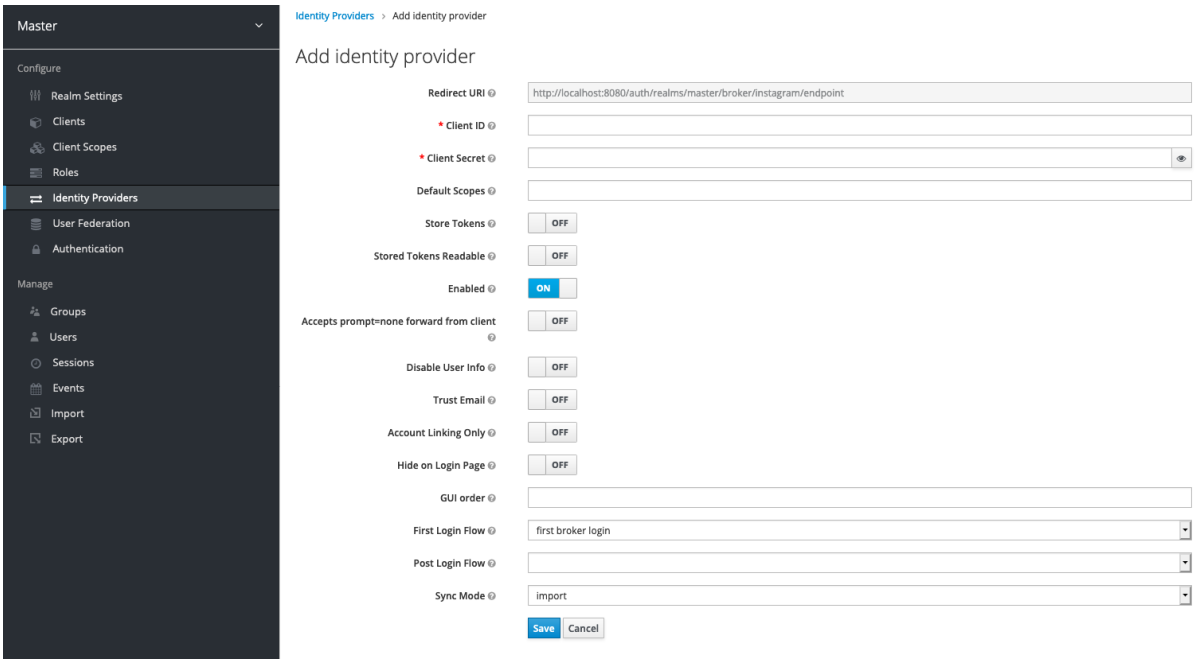
9.4.13. Instagram

流程

1. 在菜单中，单击 **Identity Providers**。

2. 从 **Add provider** 列表中，选择 **Instagram**。Red Hat Single Sign-On 显示 Instagram 身份提供程序的配置页面。

添加身份提供程序



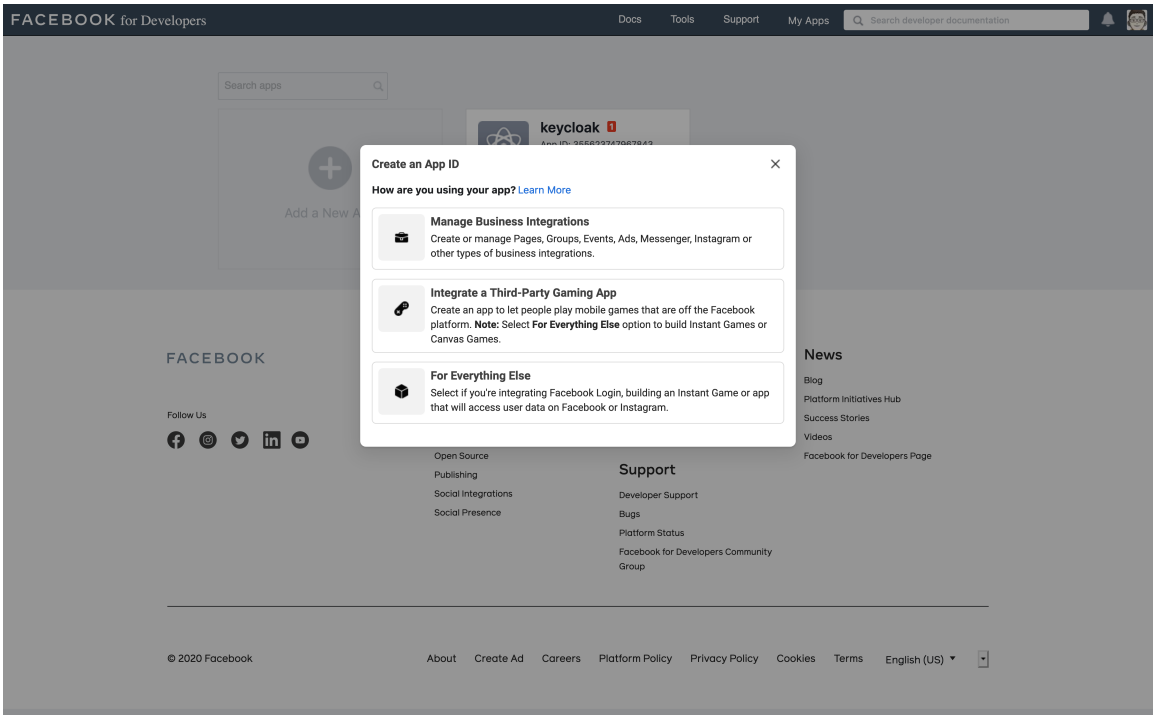
3. 将 **Redirect URI** 的值复制到您的剪贴板。

4. 在一个单独的浏览器选项卡中，打开 Facebook 开发人员控制台。

a. 点 **My Apps**。

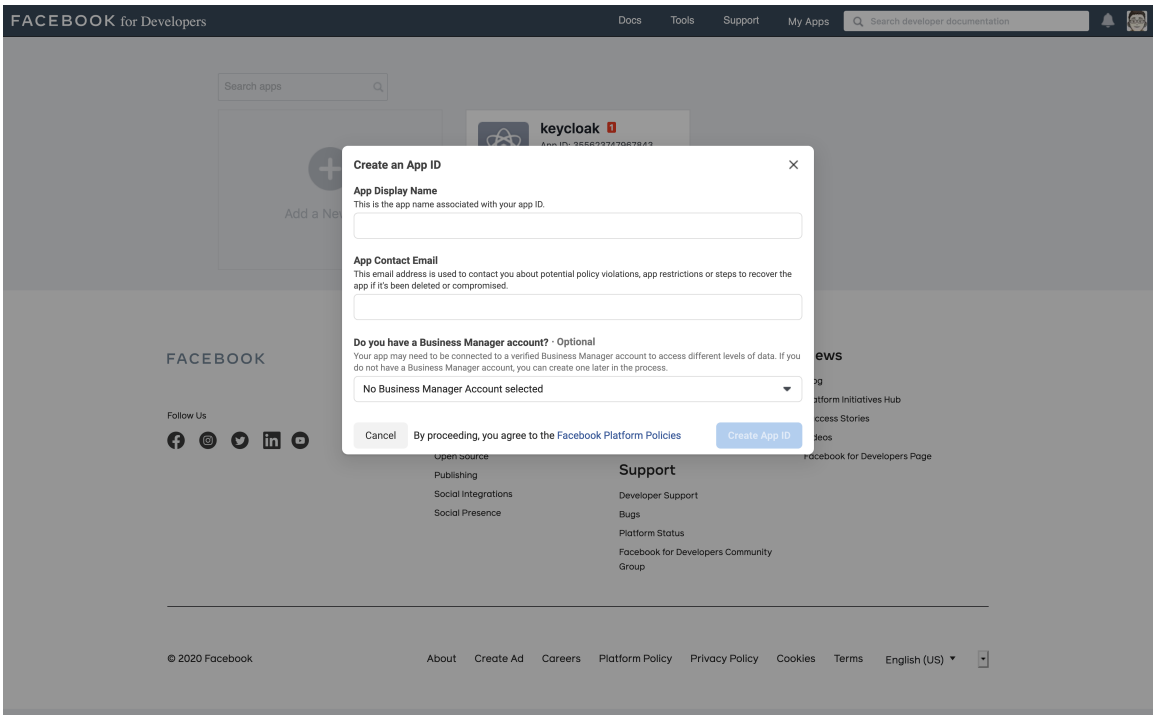
b. 选择 **Add a New App**。

添加新应用程序



c. 选择 **Forthing Else**。

创建新应用程序 ID

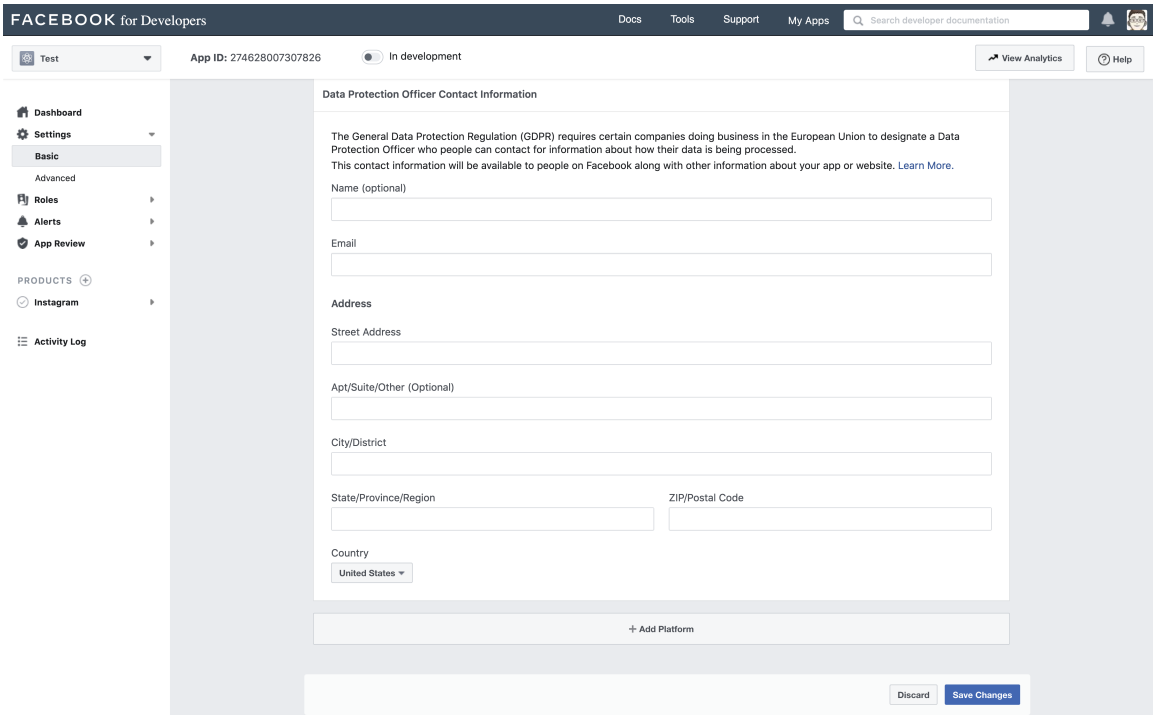


d. 填写所有必填字段。

e. 点 **Create App**。Facebook 然后将您带到仪表板。

f. 在导航面板中，选择 **Settings - Basic**。

添加平台

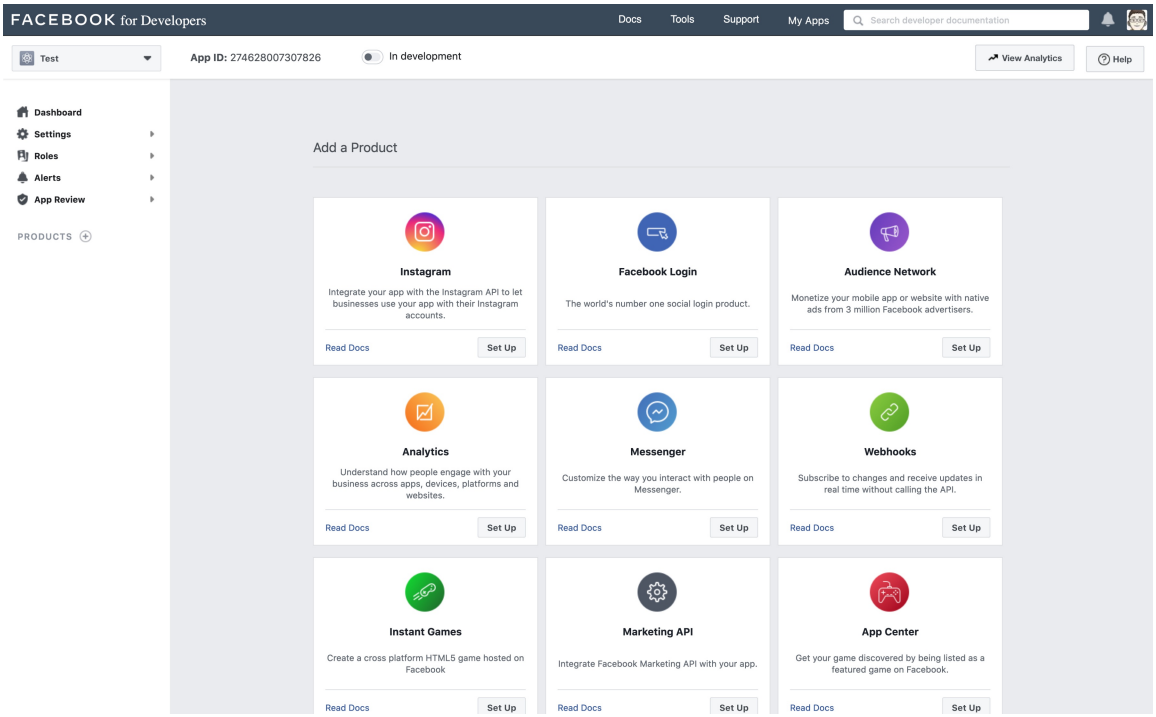


g. 选择 **+ Add Platform**。

h. 单击 **[Website]**。

i. 输入您站点的 URL。

添加产品



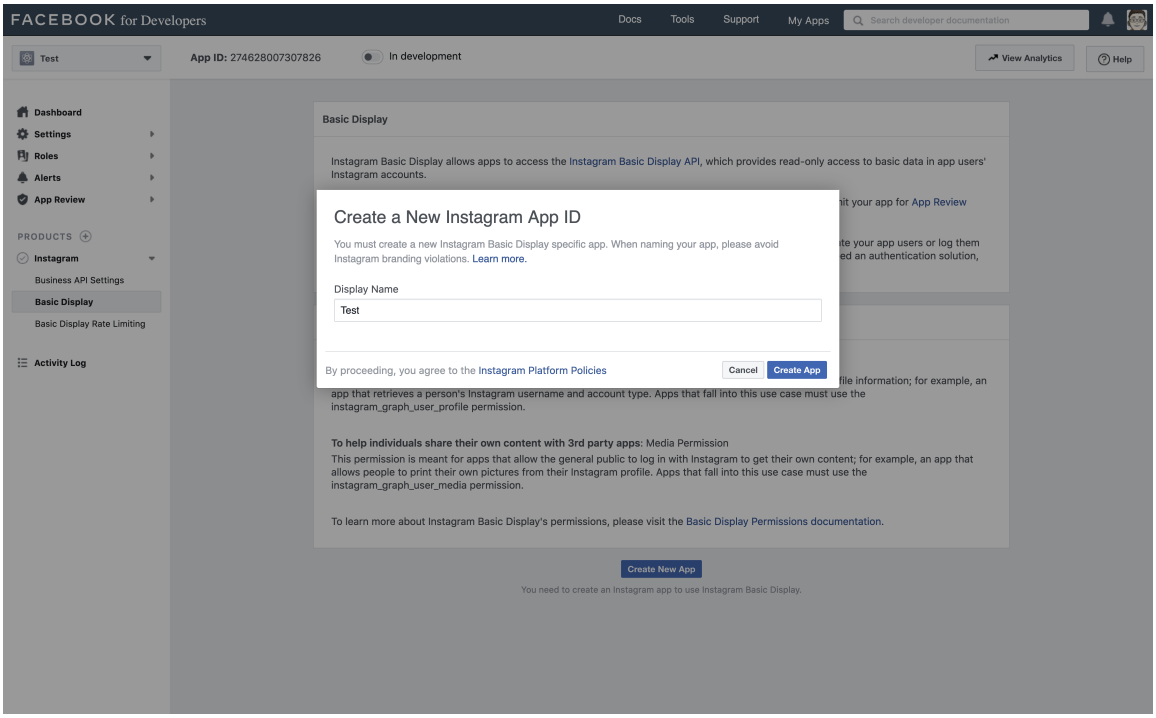
j. 从菜单中选择 **Dashboard**。

k. 在 Instagram 框中点 **Set Up**。

l. 从菜单中选择 **Instagram - Basic Display**。

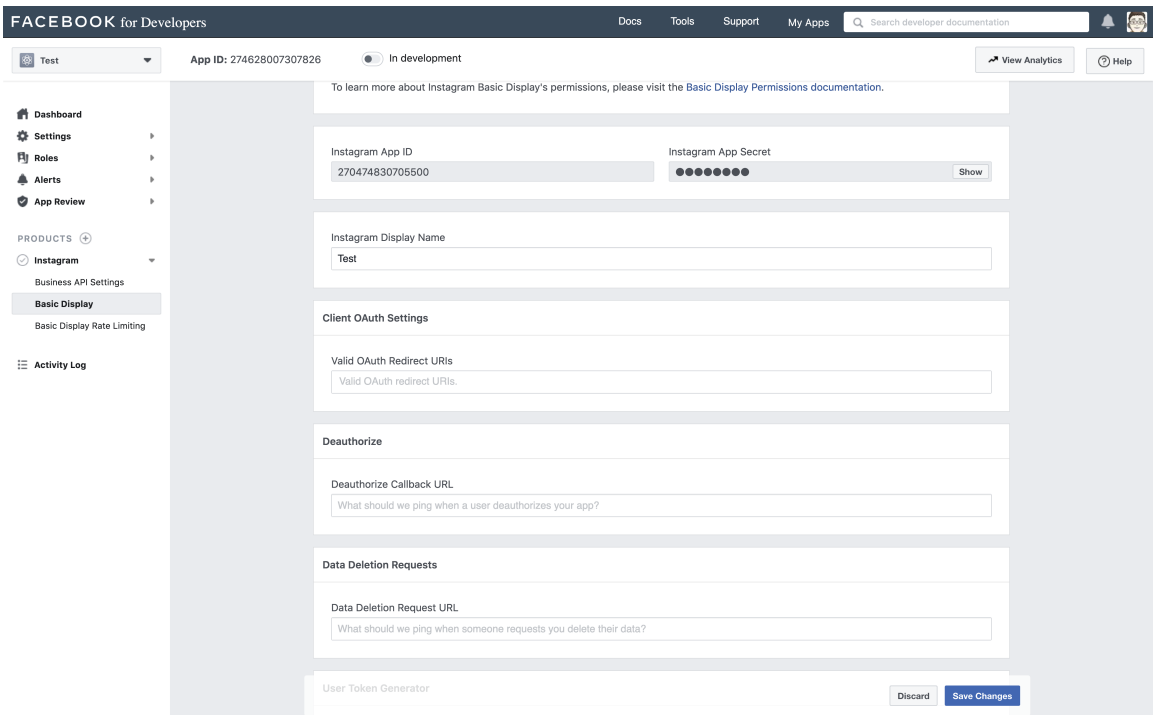
m. 点 **Create New App**。

创建新的 Instagram 应用 ID



n. 在 **Display Name** 中输入值。

设置应用程序



o. 将来自 Red Hat Single Sign-On 的 **Redirect URL** 粘贴到 **Valid OAuth Redirect URIs** 字段中。

p. 将来自 Red Hat Single Sign-On 的 **Redirect URL** 粘贴到 **Deauthorize Callback URL** 字段中。

q. 将来自 Red Hat Single Sign-On 的 **Redirect URL** 粘贴到 **Data Deletion Request URL** 字段中。

r. 在 **Instagram App Secret** 字段中，单击 **Show**。

s. 请注意 **Instagram App ID** 和 **Instagram App Secret**。

t. 点 **App Review** - **Requests**。

u. 按照屏幕上的说明进行操作。

5. 在 Red Hat Single Sign-On 中，将 **Instagram App ID** 的值粘贴到 **Client ID** 字段中。

6. 在 Red Hat Single Sign-On 中，将 **Instagram App Secret** 的值粘贴到 **Client Secret** 字段中。

7. 点 **Save**。

9.5. OPENID CONNECT V1.0 身份提供程序

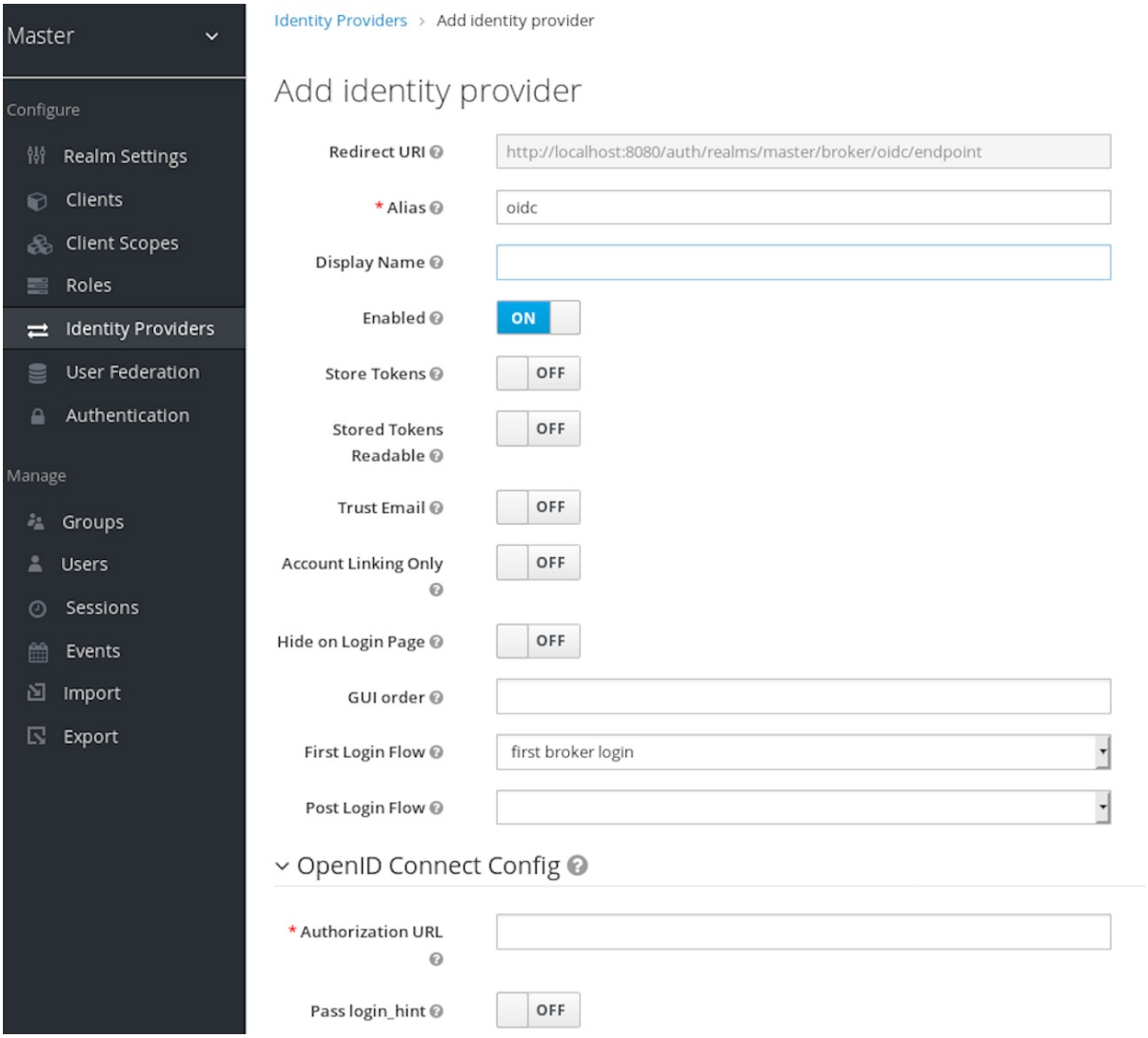
Red Hat Single Sign-On 代理基于 OpenID Connect 协议的身份供应商。这些身份提供程序(IDP)必须支持规格中定义的 授权代码流 来验证用户并授权访问权限。

流程

1. 在菜单中，单击 **Identity Providers**。

2. 从 **Add provider** 列表中，选择 **OpenID Connect v1.0**。

添加身份提供程序



3. 输入您的初始配置选项。有关配置选项的更多信息，请参阅 常规 IDP 配置。

表 9.2. OpenID 连接配置

Configuration	描述
授权 URL	OIDC 协议所需的授权 URL 端点。
令牌 URL	OIDC 协议所需的令牌 URL 端点。
注销 URL	OIDC 协议中的注销 URL 端点。这个值是可选的。
Backchannel Logout	一个后台，带外 REST 请求 IDP 以注销用户。有些 IDP 仅通过浏览器重定向执行注销，因为它们可能会使用浏览器 Cookie 来识别会话。
User Info URL	OIDC 协议定义的端点。此端点指向用户配置文件信息。

Configuration	描述
客户端身份验证	定义 Red Hat Single Sign-On 与授权代码流一起使用的客户端身份验证方法。如果 JWT 使用私钥签名，Red Hat Single Sign-On 会使用 realm 私钥。在其他情况下，定义客户端 secret。如需更多信息，请参阅客户端身份验证 规格。
客户端 ID	充当外部 IDP 的 OIDC 客户端的域。如果您使用授权代码流与外部 IDP 交互，则域必须具有 OIDC 客户端 ID。
Client Secret	来自外部 密码库的客户端机密。如果您使用授权代码流，则需要此 secret。
Client Assertion Signature Algorithm	创建 JWT 断言作为客户端身份验证的签名算法。如果 JWT 使用私钥或客户端机密签名为 jwt，则需要它。如果没有指定算法，则会调整以下算法。如果 JWT 使用私钥签名，则 **RS256** 被调整。**HS256** 以 jwt 的身份调整客户端机密。
Issuer	Red Hat Single Sign-On 根据此值验证来自 IDP 的签发者声明。
默认范围	OIDC 范围列表 Red Hat Single Sign-On 使用身份验证请求发送。默认值为 **openid**。每个范围分隔一个空格。
提示	OIDC 规格中的 prompt 参数。通过此参数，您可以强制重新验证和其他选项。如需了解更多详细信息，请参阅规格。

Configuration	描述
接受来自客户端的 prompt=none 转发	指定 IDP 是否接受包含 **prompt=none** 查询参数的身份验证请求。如果 realm 收到了带有 **prompt=none** 的 auth 请求，则域会检查用户当前是否通过身份验证，并在用户没有登录时返回 **login_required** 错误。当 Red Hat Single Sign-On 确定 auth 请求的默认 IDP （使用 **kc_idp_hint** 查询参数或者具有域的默认 IDP）时，您可以将带有 **prompt=none** 的 auth 请求转发到默认的 IDP。默认 IDP 检查用户那里的身份验证。因为并非所有 IDP 都支持带有 **prompt=none** 的请求，Red Hat Single Sign-On 会使用此切换来指示默认 IDP 支持参数，然后再重定向身份验证请求。 如果用户在 IDP 中未经身份验证的身份验证，客户端仍然会收到 **login_required** 错误。如果用户在 IDP 中是验证，如果 Red Hat Single Sign-On 必须显示需要 **用户交互的身份验证页面，则客户端仍然可以收到 interact_required** 错误。此身份验证包括所需的操作（如密码更改）、同意屏幕，以及 **first broker login** 流或 **post broker login** 流显示的屏幕。
验证签名	指定 Red Hat Single Sign-On 验证此 IDP 签名的外部 ID 令牌上的签名。如果 **ON**，Red Hat Single Sign-On 必须知道外部 OIDC IDP 的公钥。为了提高性能，Red Hat Single Sign-On 会缓存外部 OIDC 身份提供程序的公钥。如果您的身份提供程序的私钥被破坏，请更新您的密钥并清除密钥缓存。如需了解更多详细信息，请参阅 清除 cache 部分。
使用 JWKS URL	如果验证签名为 **ON，则此** 切换适用。如果使用 **JWKS URL** 为 **ON**，Red Hat Single Sign-On 从 JWKS URL 下载 IDP 的公钥。身份提供程序生成新密钥对时下载新密钥。如果 **OFF**，Red Hat Single Sign-On 使用其数据库的公钥（或证书），因此当 IDP 密钥对更改时，将新密钥导入到 Red Hat Single Sign-On 数据库。
JWKS URL	指向 IDP JWK 密钥位置的 URL。如需更多信息，请参阅 JWK 规格。如果您使用外部 Red Hat Single Sign-On 作为 IDP，如果您的 Red Hat Single Sign-On 在 http://broker-keycloak:8180 上运行，且其 realm 为 **test**，您可以使用 URL，比如 http://broker-keycloak:8180/auth/realms/test/protocol/openid-connect/certs。

Configuration	描述
验证公钥	Red Hat Single Sign-On 用来验证外部 IDP 签名的公钥。如果 **Use JWKS URL** 为 **OFF**，则适用此密钥。
验证公钥 Id	如果 **Use JWKS URL** 为 **OFF**，则适用此设置。此设置以 PEM 格式指定公钥的 ID。由于没有密钥计算密钥 ID 的标准方式，外部身份提供程序可以从 Red Hat Single Sign-On 使用的不同算法使用不同的算法。如果没有指定此字段的值，Red Hat Single Sign-On 会使用所有请求的公钥，无论外部 IDP 发送的密钥 ID 是什么。当 **ON** 时，此字段的值是 Red Hat Single Sign-On 用来验证来自供应商的签名的密钥 ID，并且必须与 IDP 指定的密钥 ID 匹配。

您可以通过提供指向 OpenID Provider 元数据的 URL 或文件来导入所有这些配置数据。如果您连接到 Red Hat Single Sign-On 外部 IDP，您可以从 < **root>/auth/realms/{realm-name}/.well-known/openid-configuration** 中导入 IDP 设置。此链接是一个 JSON 文档，描述有关 IDP 的元数据。

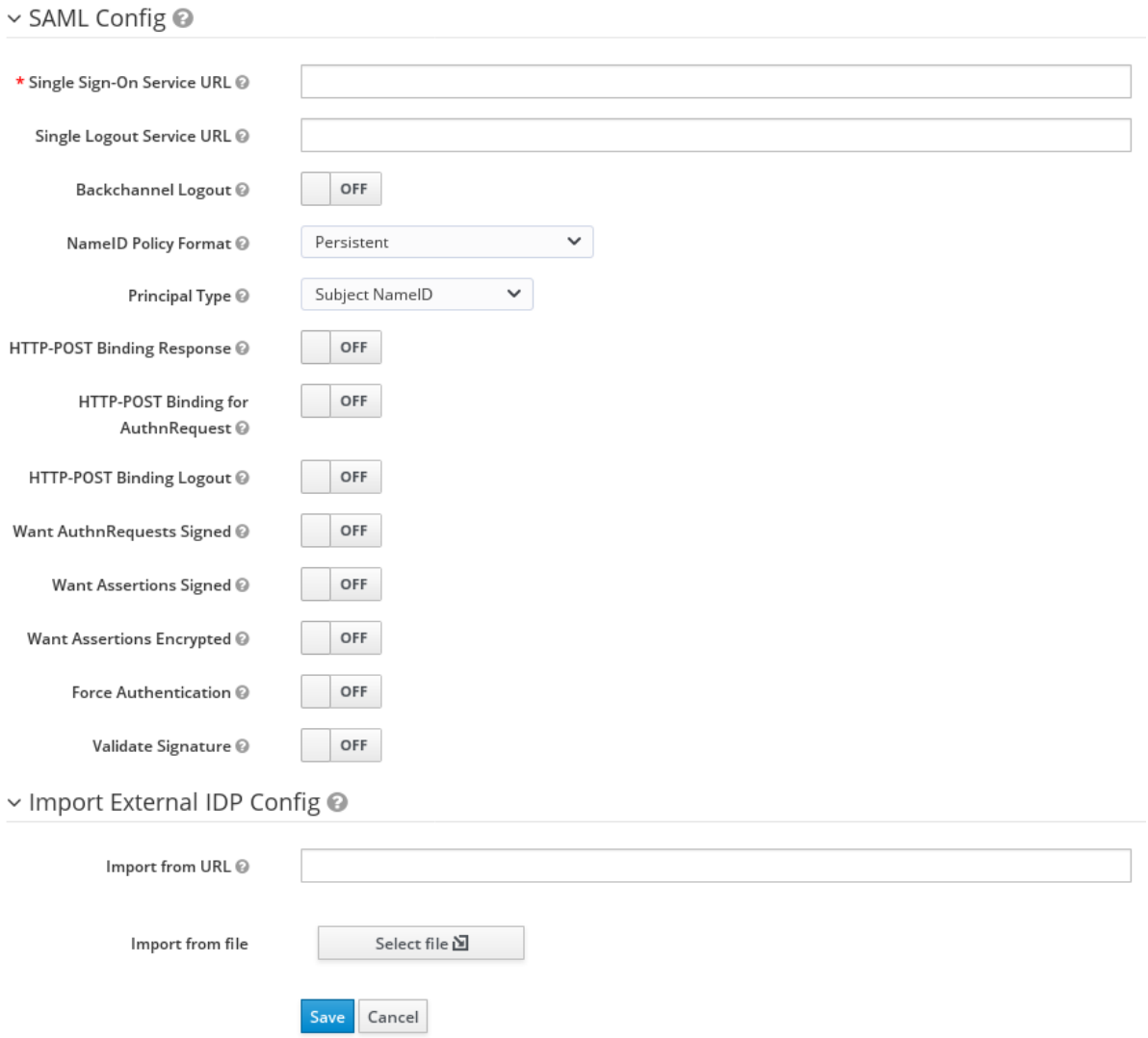
9.6. SAML V2.0 身份提供程序

红帽单点登录可以根据 SAML v2.0 协议代理身份提供程序。

流程

1. 在菜单中，单击 **Identity Providers**。
2. 从 **Add provider** 列表中，选择 **SAML v2.0**。

添加身份提供程序



3. 输入您的初始配置选项。有关配置选项的更多信息，请参阅 常规 IDP 配置。SAML 配置

Configuration	描述
服务提供商实体 ID	远程身份提供程序用于识别来自此服务提供商的请求的 SAML 实体 ID。默认情况下，此设置设置为 realms 基础 URL < **root>/auth/realms/{realm-name}**。
身份提供程序实体 ID	用于在接收 SAML 断言时验证颁发者元素的 SAML 实体 ID。如果为空，则不会执行颁发者验证。如果您的 SAML IDP 发布 IDP 实体描述符，则会在那里指定此字段的值。
单点登录服务 URL	启动身份验证过程的 SAML 端点。如果您的 SAML IDP 发布 IDP 实体描述符，则会在那里指定此字段的值。
单一 Logout 服务 URL	SAML 注销端点。如果您的 SAML IDP 发布 IDP 实体描述符，则会在那里指定此字段的值。

Configuration	描述
Backchannel Logout	如果您的 SAML IDP 支持后端频道注销，请将此开关切换为 **ON**。
NameID 策略格式	对应于名称标识符格式的 URI 引用。默认情况下，Red Hat Single Sign-On 将它设置为 **urn:oasis:names:tc:SAML:2.0:nameid-format:persistent**。
主体类型	指定 SAML 断言哪个部分将用于识别和跟踪外部用户身份。可以是 Subject NameID 或 SAML 属性（按名称或友好名称）。subject NameID 值不能与 'urn:oasis:names:tc:SAML:2.0:nameid-format:transient' NameID Policy Format 值一起设置。
主体属性	如果主体类型不是空的，该字段指定了用于标识属性的名称("Attribute [Name]")或友好名称("Attribute [Friendly Name]")。
allow create	允许外部身份提供程序创建新标识符来代表主体。
HTTP-POST 绑定响应	控制 SAML 绑定以响应外部 IDP 发送的任何 SAML 请求。当 **OFF** 时，Red Hat Single Sign-On 使用 Redirect Binding。
HTTP-POST Binding for AuthnRequest	在从外部 IDP 请求身份验证时控制 SAML 绑定。当 **OFF** 时，Red Hat Single Sign-On 使用 Redirect Binding。
想要 AuthnRequests Signed	当 **ON** 时，Red Hat Single Sign-On 使用域的密钥对签署发送到外部 SAML IDP 的请求。
签名算法	如果 **Want AuthnRequests Signed** 为 **ON**，要使用的签名算法。
SAML 签名密钥名称	使用 POST 绑定发送签名的 SAML 文档包含 **KeyName** 元素中的签名密钥识别（默认情况下，包含 Red Hat Single Sign-On 密钥 ID）。外部 SAML IDP 预期不同的密钥名称。此切换控制 **KeyName** 是否含有：* **KEY_ID** - Key ID。* **CERT_SUBJECT** - 来自与域密钥对应的证书的主题。Microsoft Active Directory Federation 服务预期 **CERT_SUBJECT**。* **NONE** - Red Hat Single Sign-On 在 SAML 信息中省略了密钥名称提示。
强制身份验证	用户必须在外部 IDP 中输入凭证，即使该用户已经登录。

Configuration	描述
验证签名	当 **ON** 时，域需要 SAML 请求和来自外部 IDP 的响应是数字签名。
验证 X509 证书	红帽单点登录使用公共证书来验证 SAML 请求的签名以及来自外部 IDP 的响应。
为服务提供商元数据签名	**ON** 时，红帽单点登录使用域的密钥对来签署 SAML 服务提供商元数据描述符。
pass subject	控制 Red Hat Single Sign-On 将 **login_hint** 查询参数转发到 IDP。Red Hat Single Sign-On 将此字段的值添加到 AuthnRequest 的 Subject 中的 login_hint 参数，以便目标供应商可以预先填充其登录表单。
Attribute Consuming Service Index	标识设置为请求远程 IDP 的属性。Red Hat Single Sign-On 会自动将身份提供程序配置中映射的属性添加到自动生成的 SP 元数据文档中。
属性假设服务名称	自动生成的 SP 元数据文档中公告的属性集的描述性名称。

您可以通过提供一个 URL 或指向外部 IDP 实体描述符的 URL 或文件来导入所有配置数据。如果您连接到 Red Hat Single Sign-On 外部 IDP，您可以从 URL < **root>/auth/realms/{realm-name}/protocol/saml/descriptor** 导入 IDP 设置。此链接是一个 XML 文档，描述了有关 IDP 的元数据。您还可以通过提供一个指向外部 SAML IDP 实体描述符的 URL 或 XML 文件来导入所有这些配置数据。

9.6.1. 请求特定的 AuthnContexts

身份提供程序有助于客户端指定验证用户身份的身份验证方法的限制。例如，询问 MFA、Kerberos 身份验证或安全要求。这些限制使用特定的 AuthnContext 标准。客户端可以询问一个或多个条件，并指定身份提供程序必须与请求的 AuthnContext、准确或满足其它等效项的匹配方式。

您可以通过在 Requested AuthnContext Constraints 部分中添加 ClassRefs 或 DeclRefs 来列出您的服务提供程序要求。通常，您需要提供 ClassRefs 或 DeclRefs，因此请检查您的身份提供程序文档受支持值。如果没有 ClassRefs 或 DeclRefs，身份提供程序不会强制实施额外的限制。

表 9.3. 请求的 AuthnContext 限制

Configuration	描述
比较	身份提供程序使用 方法来评估上下文要求。可用值有 **Exact**、**Minimum**、**Maximum** 或 **Better**。默认值为 **Exact**。
AuthnContext ClassRefs	描述所需标准的 AuthnContext ClassRefs。
AuthnContext DeclRefs	AuthnContext DeclRefs 描述所需标准。

9.6.2. SP Descriptor

当您访问供应商的 SAML SP 元数据时，在身份提供程序配置设置中查找 **Endpoints** 项。它包含 **SAML 2.0 服务提供商元数据** 链接，用于为服务提供商生成 SAML 实体描述符。您可以下载描述符或复制其 URL，然后将其导入到远程身份提供程序。

通过进入以下 URL 来公开此元数据：

```
http[s]://{host:port}/auth/realms/{realm-name}/broker/{broker-alias}/endpoint/descriptor
```

在访问描述符前，请确保您保存任何配置更改。

9.6.3. 在 SAML 请求中发送主题

默认情况下，指向 SAML 身份提供程序的社交按钮将用户重定向到以下登录 URL：

```
http[s]://{host:port}/auth/realms/${realm-name}/broker/{broker-alias}/login
```

在此 URL 中添加名为 **login_hint** 的查询参数，将参数值添加到 SAML 请求中作为 Subject 属性。如果此查询参数为空，Red Hat Single Sign-On 不会添加主题到请求。

启用 "Pass subject" 选项以在 SAML 请求中发送主题。

9.7. 客户端讨论的身份提供程序

OIDC 应用程序可以通过提示 Red Hat Single Sign-On 登录页面来绕过 Red Hat Single Sign-On 登录页面。您可以通过在授权代码流授权端点中设置 **kc_idp_hint** 查询参数来启用此功能。

使用 Red Hat Single Sign-On OIDC 客户端适配器，您可以在访问应用程序中的安全资源时指定这个查询参数。

例如：

```
GET /myapplication.com?kc_idp_hint=facebook HTTP/1.1
Host: localhost:8080
```

在这种情况下，您的域必须具有 **facebook** 别名的身份提供程序。如果此提供程序不存在，则会显示登录表单。

如果使用 **keycloak.js** 适配器，您也可以实现与以下相同的行为：

```
const keycloak = new Keycloak('keycloak.json');

keycloak.createLoginUrl({
 idpHint: 'facebook'
});
```

使用 **kc_idp_hint** 查询参数，如果您为 **Identity Provider Redirector** authenticator 配置了一个默认身份提供程序，客户端可以覆盖默认身份提供程序。通过将 **kc_idp_hint** 查询参数设置为空值，客户端可以禁用自动重定向。

9.8. 映射声明和断言

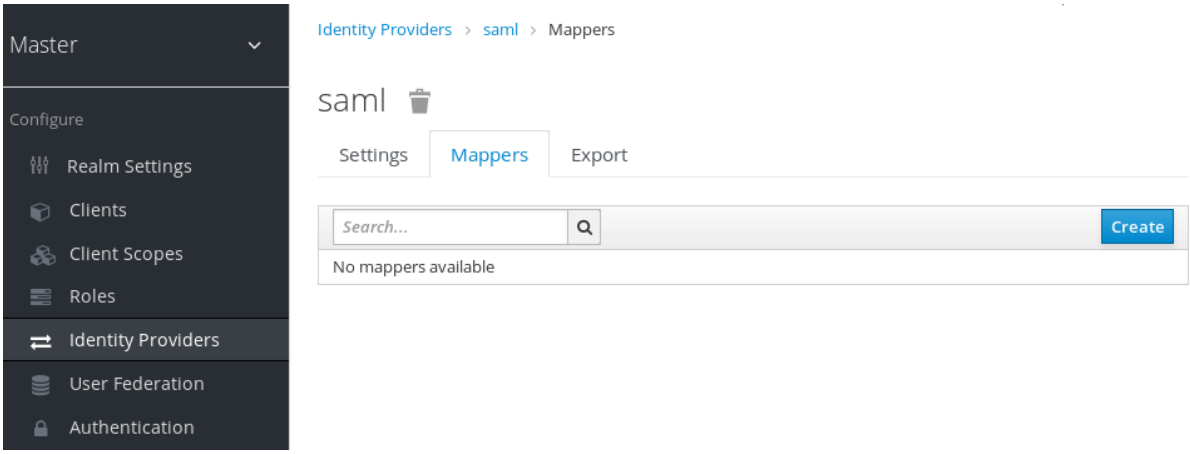
您可以将您通过身份验证的外部 IDP 提供的 SAML 和 OpenID Connect 元数据导入到域中。导入后，您可以提取用户配置集元数据和其他信息，以便您可供您的应用程序使用。

每个用户使用外部身份提供程序登录到您的域，在本地 Red Hat Single Sign-On 数据库中根据 SAML 或 OIDC 断言和声明的元数据，在本地 Red Hat Single Sign-On 数据库中都有一个条目。

流程

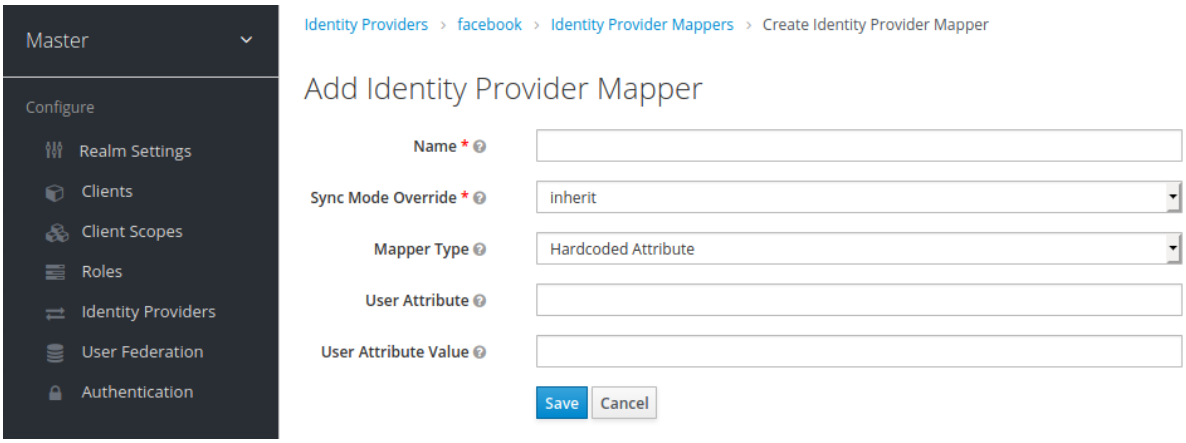
1. 在菜单中，单击 **Identity Providers**。
2. 从列表中选择其中一个身份提供程序。
3. 点 **Mappers** 选项卡。

身份提供程序映射器



4. 点 **Create**。

身份提供程序映射器



5. 为 **Sync Mode Override** 选择一个值。当用户根据此设置重复登录时，mapper 会更新用户信息。
   a. 选择 **legacy**，以使用上一个 Red Hat Single Sign-On 版本的行为。
   b. 选择 **导入** 以在首次使用特定身份提供程序登录 Red Hat Single Sign-On 中首次在 Red Hat Single Sign-On 中创建数据时，从 导入数据。
   c. 选择 **强制** 在每次用户登录时更新用户数据。
   d. 选择 **继承**，以使用身份提供程序中配置的同步模式。所有其他选项将覆盖此同步模式。

6. 从 **Mapper Type** 列表中选择一个映射程序。将鼠标悬停在 **映射程序类型上**，以获取映射程序和配置的说明，以输入 mapper。

7. 点 **Save**。

对于基于 JSON 的声明，您可以使用点表示法来嵌套和方括号来按索引访问数组字段。例如，**contact.address[0].country**。

要调查社交提供程序提供的用户配置集 JSON 数据的结构，您可以在服务器的 app-server 配置文件中 (domain.xml 或 standalone.xml)启用 **DEBUG** 级别日志记录器 **org.keycloak.social.user_profile_dump**。

9.9. 可用用户会话数据

用户从外部 IDP 登录后，Red Hat Single Sign-On 会存储用户会话记录您可以访问的数据。此数据可以传播到请求使用适当客户端映射程序传递给客户端的令牌或 SAML 断言登录的客户端。

identity_provider

用于执行登录的代理的 IDP 别名。

identity_provider_identity

当前验证的用户的 IDP 用户名。通常与 Red Hat Single Sign-On 用户名相同，但并不总是如此。例如，Red Hat Single Sign-On 可将用户 john' 链接到 Facebook 用户 **john123@gmail.com**。在这种情况下，用户会话备注的值是 **john123@gmail.com**。

您可以使用类型为 **User Session 注意的协议** 映射程序 将此信息传播到您的客户端。

9.10. 第一登录流程

当用户通过身份验证代理登录时，Red Hat Single Sign-On 导入用户在本地数据库中的用户链接。当 Red Hat Single Sign-On 成功通过外部身份提供程序验证用户时，会出现以下两种情况：

- Red Hat Single Sign-On 已导入并把用户帐户与经过身份验证的身份提供程序帐户相关联。在这种情况下，Red Hat Single Sign-On 以现有用户身份进行身份验证，再重新重定向到应用程序。

- 在 Red Hat Single Sign-On 中不存在这个用户的帐户。通常，您将注册并导入一个新帐户到 Red Hat Single Sign-On 数据库，但可能存在有一个具有相同电子邮件地址的现有红帽单点登录帐户。自动将现有本地帐户链接到外部身份提供程序是一个潜在的安全漏洞。您无法始终信任您从外部身份提供程序获取的信息。

在处理其中的一些情况时，不同的机构有不同的要求。在 Red Hat Single Sign-On 中，您可以使用 IDP 设置中的 **First Login Flow** 选项，为从外部 IDP 登录的用户选择首次登录 的工作流。默认情况下，**First Login Flow** 选项指向 **第一个代理登录流**，但您可以为不同的身份提供程序使用流或不同的流。

流位于 **Authentication** 选项卡下的管理控制台中。当您选择 **First Broker Login** 流时，您会看到默认使用的验证器。您可以重新配置现有的流。例如，您可以禁用某些验证器，将其部分标记为 **必需**，或者配置某些验证器。

9.10.1. 默认首次登录流身份验证

查看配置集

- 此验证器显示配置集信息页，因此用户可以检查其配置集，Red Hat Single Sign-On from a identity provider.

- 您可以在 Actions 菜单中设置 **Update Profile On First Login** 选项。
- 在 **ON** 时，用户会看到配置集页面，请求额外信息联合该用户的身份。
- **缺少** 时，如果身份提供程序未提供强制信息，则会在配置集页面中显示用户，如电子邮件、名字或姓氏。
- 当 **OFF** 时，配置集页面不会显示，除非用户在 **Confirm Link Existing Account** authenticator 显示的页面中的 **Review profile info** 链接中点击。

创建用户如果唯一性

此验证器将检查是否有现有的红帽单点登录帐户，该帐户具有相同的电子邮件或用户名，如来自身份提供程序的帐户。如果没有，则验证器只创建一个新的本地 Red Hat Single Sign-On 帐户，并将其与身份提供程序链接，整个流完成。否则，它会进入下一个 **Handle Existing Account** 子流。如果始终需要确保没有重复的帐户，您可以将这个验证器标记为 **REQUIRED**。在这种情况下，如果存在现有的 Red Hat Single Sign-On 帐户并且用户需要通过帐户管理链接身份提供程序帐户，用户会看到错误页面。

- 此验证器将验证是否有红帽单点登录帐户与身份提供程序帐户具有相同的电子邮件或用户名。
- 如果帐户不存在，则验证器会创建一个本地 Red Hat Single Sign-On 帐户，将这个帐户与身份提供程序链接，并终止流。
- 如果存在帐户，则验证器将实施下一个 **Handle Existing Account** 子流。
- 为确保没有重复的帐户，您可以将这个验证器标记为 **REQUIRED**。如果存在 Red Hat Single Sign-On 帐户，该用户会看到错误页面，用户必须通过帐户管理链接其身份提供程序帐户。

确认链接现有帐户

- 在信息页面中，用户会看到一个具有相同电子邮件的 Red Hat Single Sign-On 帐户。用户可以再次查看其配置集并使用不同的电子邮件或用户名。流重启并返回到 **Review Profile** 验证器。
- 另外，用户可以确认他们想将其身份提供程序帐户与其现有的红帽单点登录帐户相关联。
- 如果您不希望用户看到此确认页面，请禁用此验证器，并通过电子邮件验证或重新验证来直接链接身份提供程序帐户。

通过电子邮件验证现有帐户

- 这个验证器 **默认为** ALTERNATIVE。如果域配置了 SMTP 设置，Red Hat Single Sign-On 会使用这个验证器。
- 验证器将发送电子邮件给用户，以确认他们想将身份提供程序与红帽单点登录帐户链接。
- 如果您不想通过电子邮件确认链接，但希望用户使用密码重新进行身份验证，则禁用此验证器。

通过 Re-authentication 验证现有帐户

- 如果电子邮件验证器不可用，则使用此验证器。例如，您尚未为您的域配置 SMTP。此身份验证器显示一个登录屏幕，供用户通过身份提供程序链接其 Red Hat Single Sign-On 帐户。
- 用户还可重新与已链接到红帽单点登录帐户的另一个身份提供程序重新进行身份验证。

- 您可以强制用户使用 OTP。否则，如果您为用户帐户设置了 OTP，则它是可选的。

9.10.2. 自动链接现有的第一个登录流

警告

AutoLink 验证器在通用环境中具有危险，用户可使用任意用户名或电子邮件地址注册自己。请勿使用此验证器，除非您仔细策展用户注册并分配用户名和电子邮件地址。

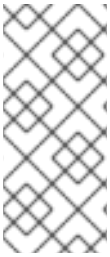
要配置第一个登录流，在不提示的情况下自动链接用户，请使用以下两个验证器创建一个新流：

创建用户如果唯一性

此验证器可确保 Red Hat Single Sign-On 处理唯一的用户。将验证器要求设置为 **Alternative**。

自动设置现有用户

此验证器将现有用户设置为身份验证上下文，无需验证。将验证器要求设置为 "Alternative"。



注意

此设置是可用的最简单设置，但也可以使用其他验证器。例如：如果您想要最终用户确认其配置集信息，您可以将 Review Profile 验证器添加到流的开头。* 您可以为这个流程添加验证机制，强制用户验证其凭证。添加验证机制需要复杂的流程。例如，您可以在 "Alternative" 子流中设置 "Automically Set Existing User" 和 "Password Form"，作为 "Alternative" 子流。

9.10.3. 禁用自动用户创建

默认第一个登录流查找与外部身份匹配的 Red Hat Single Sign-On 帐户，并提供链接它们。如果没有匹配的 Red Hat Single Sign-On 帐户，网络流会自动创建一个。

对于某些设置，这个默认行为可能不适合。例如，当您使用只读 LDAP 用户存储时，会预先创建所有用户。在这种情况下，您必须关闭自动用户创建。

禁用用户创建：

流程

1. 在菜单中，单击 **Authentication**。
2. 从列表中选择 **First Broker Login**。
3. 将 **Create User If unique** to **DISABLED** 设置为 DISABLED。
4. 将 **"确认链接现有账户"** 设置为 **DISABLED**。

此配置还意味着红帽单点登录本身无法确定哪个内部帐户与外部身份对应。因此，**Verify Existing Account By Re-authentication** authenticator 将要求用户提供用户名和密码。

9.10.4. 检测现有的用户首次登录流

要配置第一个登录流，请执行以下操作：

- 只有已在此域中注册的用户才能登录，
- 用户会自动链接，而不会被提示，

使用以下两个验证器创建一个新流：

检测现有代理用户

这个验证器可确保处理唯一的用户。将验证器要求设置为 **强制**。

自动设置现有用户

自动将现有用户设置为身份验证上下文，而无需验证。将验证器要求设置为 **强制**。

您必须将身份提供程序 **配置的第一个登录流** 设置为该流。如果要更新具有身份提供程序属性的用户配置集(Last Name, First Name...)，您也可以将 **Sync Mode** 设置为 **force**。

注意

如果要将身份委派给其他身份提供程序（如 github, facebook ...）但您希望管理可以登录的用户，可以使用这个流。

使用这个配置，Red Hat Single Sign-On 无法确定哪个内部帐户与外部身份对应。**Verify Existing Account By Re-authentication** authenticator 请求供应商获取用户名和密码。

9.11. 检索外部 IDP 令牌

使用 Red Hat Single Sign-On，您可以使用 IDP 设置页面上的 **存储** 令牌和来自身份验证流程的令牌和响应。

应用程序代码可以检索这些令牌和响应以导入额外的用户信息，或者安全地请求外部 IDP。例如，应用程序可以使用 Google 令牌来使用不同的 Google 服务和 REST API。要检索特定身份提供程序的令牌，请按如下所示发送请求：

```
GET /auth/realms/{realm}/broker/{provider_alias}/token HTTP/1.1
Host: localhost:8080
Authorization: Bearer <KEYCLOAK ACCESS TOKEN>
```

应用程序必须通过 Red Hat Single Sign-On 进行身份验证并接收访问令牌。此访问令牌必须设置 **broker** 客户端级角色 **read-token**，因此用户必须拥有此角色的角色映射，客户端应用程序必须在其范围内拥有该角色。在这种情况下，因为您要在 Red Hat Single Sign-On 中访问受保护的服务，请在用户身份验证期间发送红帽单点登录发布的访问令牌。您可以通过将 **Stored Tokens Readable** 开关设置为 **ON**，将这个角色分配给代理配置页面中新导入的用户。

这些外部令牌可以通过供应商再次登录或使用客户端发起的帐户链接 API 来重新建立。

9.12. IDENTITY BROKER LOGOUT

注销时，Red Hat Single Sign-On 会向外部身份提供程序发送请求，该供应商最初用来登录并注销此身份提供程序。您可以跳过此行为，避免从外部身份提供程序注销。如需更多信息，请参阅 适配器注销文档。

第 10 章 SSO 协议

本节讨论身份验证协议(Red Hat Single Sign-On 身份验证服务器)，以及红帽单点登录身份验证服务器保护应用程序的方式，并与这些协议交互。

10.1. OPENID CONNECT

OpenID Connect (OIDC)是一个身份验证协议，它是 OAuth 2.0 的扩展。

OAuth 2.0 是用于构建授权协议的框架，不完整。但是，OIDC 是一个完整的身份验证和授权协议，它使用 Json Web Token (JWT)标准。JWT 标准定义了身份令牌 JSON 格式，以及以紧凑和 Web 友好的方式对数据进行数字签名的方法。

通常，OIDC 实施两个用例。第一个情况是请求红帽单点登录服务器验证用户的应用程序。成功登录后，应用程序会收到 *身份令牌*和 *访问令牌*。*身份令牌* 包含用户信息，包括用户名、电子邮件和配置集信息。域 *对访问令牌* 进行数字签名，其中包含应用可用于决定用户在应用中可以访问的资源的访问信息（如用户角色映射）。

第二个用例是访问远程服务的客户端。

- 客户端从 Red Hat Single Sign-On 请求 *访问令牌*，以代表用户在远程服务上调用。
- Red Hat Single Sign-On 对用户进行身份验证，并要求用户同意授予请求客户端的访问权限。
- 客户端接收由域数字签名的 *访问令牌*。
- 客户端利用 *访问令牌* 在远程服务上发出 REST 请求。
- 远程 REST 服务提取 *访问令牌*。
- 远程 REST 服务验证令牌签名。
- 远程 REST 服务根据令牌中的访问信息（处理或拒绝请求）决定。

10.1.1. OIDC 身份验证流

OIDC 有几个方法或流，客户端或应用程序可以用来验证用户并接收 *身份和 访问令牌*。该方法取决于请求访问的应用或客户端的类型。

10.1.1.1. 授权代码流

授权代码流是基于浏览器的协议，适合验证和授权基于浏览器的应用程序。它使用浏览器重定向来获取*身份*和 *访问*令牌。

1. 用户使用浏览器连接至应用。应用程序检测到用户没有登录到应用程序。
2. 应用将浏览器重定向到红帽单点登录以进行身份验证。
3. 应用通过回调 URL 作为浏览器重定向中的查询参数传递。红帽单点登录在成功身份验证后使用参数。
4. Red Hat Single Sign-On 对用户进行身份验证，并创建了一次性、简短的临时代码。
5. Red Hat Single Sign-On 使用回调 URL 重定向到应用程序，并在回调 URL 中将临时代码添加为查询参数。

6. 该应用提取临时代码，并对红帽单点登录进行后台 REST 调用，以交换 *身份*、*访问*和*刷新*访问令牌的代码。为防止重播攻击，无法多次使用临时代码。

注意

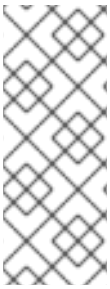
在那个令牌的生命周期中，系统会受到 stolen 令牌的影响。出于安全性和可扩展性的原因，访问令牌通常设置为快速过期，因此后续令牌请求会失败。如果令牌过期，应用程序可以使用由登录协议发送的额外 *刷新令牌* 获取新的访问令牌。

机密 客户端在交换令牌临时代码时提供客户端机密。不需要 *公共*客户端来提供客户端 secret。当 HTTPS 被严格强制时，*公共*客户端安全，并且严格控制为客户端注册的 URI。HTML5/JavaScript 客户端必须是 *公共客户端*，因为无法安全地将客户端 secret 传输到 HTML5/JavaScript 客户端。如需了解更多详细信息，请参阅管理客户端 一章。

Red Hat Single Sign-On 还支持 代码交换 规范的验证密钥。

10.1.1.2. 隐式流

Implicit Flow 是基于浏览器的协议。它与授权代码流类似，但请求较少，也没有刷新令牌。



注意

当令牌 *通过重定向* URI 传输时，访问令牌可能会泄漏（参见下文）。

另外，这个流不会为客户端提供刷新令牌。因此，访问令牌必须长期存在，或者用户在过期时必须重新验证。

我们不建议使用这个流程。支持这个流，因为它是 OIDC 和 OAuth 2.0 规范。

该协议如下：

1. 用户使用浏览器连接至应用。应用程序检测到用户没有登录到应用程序。
2. 应用将浏览器重定向到红帽单点登录以进行身份验证。
3. 应用通过回调 URL 作为浏览器重定向中的查询参数传递。Red Hat Single Sign-On 在成功身份验证后使用查询参数。
4. 红帽单点登录验证用户身份并创建 *身份和 访问令牌*。Red Hat Single Sign-On 使用回调 URL 重定向到应用程序，并 *另添加身份和 访问令牌* 作为回调 URL 中的查询参数。
5. 应用从回调 URL 中提取 *身份和 访问令牌*。

10.1.1.3. 资源所有者密码凭证授予(Direct Access Grants)

REST 客户端使用 *直接访问 Grants* 来代表用户获取令牌。这是一个 HTTP POST 请求，包含以下内容：

- 用户的凭据。凭据以表格参数形式发送。
- 客户端的 ID。
- 客户端机密（如果是一个机密客户端）。

HTTP 响应包含 *身份*、*访问 和刷新令牌*。

10.1.1.4. 客户端凭证授权

Client Credentials Grant 根据与客户端关联的服务帐户的元数据和权限来创建令牌，而不是获取代表外部用户有效的令牌。REST 客户端使用客户端 *凭据*。

如需更多信息，请参阅服务帐户章节。

10.1.1.5. 设备授权

这供在具有有限输入功能或缺少合适的浏览器的互联网连接设备上运行的客户端使用。以下是协议的简单概述：

1. 应用程序请求 Red Hat Single Sign-On a device code and user code。Red Hat Single Sign-On 创建了设备代码和用户代码。Red Hat Single Sign-On 将包括设备代码和用户代码返回响应。
2. 该应用为用户代码和验证 URI 提供用户。用户使用其他浏览器访问验证 URI 进行验证。
3. 应用程序重复轮询 Red Hat Single Sign-On 来查找用户是否完成了用户授权。如果完成了用户身份验证，应用程序会交换 *身份*、*访问*和*刷新*令牌的设备代码。

10.1.1.6. 发起的后向通道身份验证授权的客户端

想要直接与 OpenID 提供商通信的客户端使用此功能，而无需通过 OAuth 2.0 的授权代码授权等用户浏览器重定向。以下是协议的简单概述：

1. 客户端请求 Red Hat Single Sign-On a auth_req_id，用于标识客户端发出的身份验证请求。红帽单点登录创建 auth_req_id。
2. 在收到这个 auth_req_id 后，这个客户端需要重复轮询 Red Hat Single Sign-On 以获取 Access Token、Refresh Token 和 ID Token，以返回 auth_req_id，直到用户通过身份验证为止。

管理员可以将客户端初始后端身份验证(CIBA)相关操作配置为每个域的 **CIBA Policy**。

另请参阅 Red Hat Single Sign-On 文档的其他位置，如安全应用程序与服务指南中的 后端身份验证端点部分，以及 安全应用程序和服务指南的客户端初始身份验证评测部分。

10.1.1.6.1. CIBA 策略

管理员在 **Admin Console** 上执行以下操作：

- 打开 **Authentication** → **CIBA Policy** 选项卡。
- 配置项目，**再单击保存**。

可配置项及其描述如下。

Configuration	描述
Backchannel Token Delivery Mode	指定 CD （交换设备）如何获得验证结果和相关令牌。有三种模式，即 "poll" 和 "push"。Red Hat Single Sign-On 只支持 "poll"。默认设置为 "poll"。这个配置是必需的。如需了解更多详细信息，请参阅 CIBA 规格。

Configuration	描述
过期时间	自收到身份验证请求时，"auth_req_id"的过期时间（以秒为单位）。默认设置为 120。这个配置是必需的。如需了解更多详细信息，请参阅 CIBA 规格。
Interval（间隔）	CD (Consumption Device)需要等待轮询请求到令牌端点的时间间隔（以秒为单位）。默认设置为 5。此配置是可选的。如需了解更多详细信息，请参阅 CIBA 规格。
请求的身份验证用户 Hint	识别最终用户请求谁进行身份验证的方法。默认设置为 "login_hint"。有三种模式："login_hint"、"login_hint_token" 和 "id_token_hint"。Red Hat Single Sign-On only supports "login_hint".这个配置是必需的。如需了解更多详细信息，请参阅 CIBA 规格。

10.1.1.6.2. 供应商设置

CIBA 授权使用以下两个供应商。

1. 身份验证频道提供程序：提供 Red Hat Single Sign-On 和实际通过 AD 验证用户身份的实体（验证设备）之间的通信。
2. 用户 Resolver Provider：从客户端提供的信息中获得 Red Hat Single Sign-On 的 **UserModel**，以标识该用户。

Red Hat Single Sign-On 兼具默认提供程序。但是，管理员需要设置身份验证频道供应商，如下所示：

```
<spi name="ciba-auth-channel">
    <default-provider>ciba-http-auth-channel</default-provider>
    <provider name="ciba-http-auth-channel" enabled="true">
        <properties>
            <property name="httpAuthenticationChannelUri"
value="https://backend.internal.example.com/auth"/>
        </properties>
    </provider>
</spi>
```

可配置项及其描述如下。

Configuration	描述
httpAuthenticationChannelUri	指定实际通过 AD 验证用户身份的实体的 URI （身份验证设备）。

10.1.1.6.3. 身份验证频道供应商

CIBA 标准本规没有指定如何验证 AD 用户。因此，它可能会自行决定实施，例如单点登录将该身份验证

CIBA 标准文档没有指定如何验证 AD 用户。因此，它可能会自行决定实施。红帽单点登录将该身份验证委派给外部身份验证实体。要与身份验证实体通信，Red Hat Single Sign-On 提供身份验证频道提供程序。

其 Red Hat Single Sign-On 的实施假定身份验证实体由红帽单点登录管理员控制，以便红帽单点登录信任身份验证实体。不建议使用 Red Hat Single Sign-On 管理员无法控制的身份验证实体。

身份验证频道提供程序作为 SPI 供应商提供，以便 Red Hat Single Sign-On 的用户可以实施自己的供应商以满足其环境。Red Hat Single Sign-On 提供其默认供应商 HTTP Authentication Channel Provider，它使用 HTTP 与身份验证实体通信。

如果红帽单点登录用户希望使用 HTTP 身份验证频道提供商，他们需要知道其在红帽单点登录和由以下两部分组成的身份验证实体之间的合同。

身份验证委派请求/响应

红帽单点登录将身份验证请求发送到身份验证实体。

身份验证结果通知/ACK

身份验证实体通知 Red Hat Single Sign-On 的身份验证结果。

身份验证委派请求/响应由以下消息传递组成：

身份验证委派请求

请求从 Red Hat Single Sign-On 发送到身份验证实体，以要求它通过 AD 进行用户身份验证。

> POST [delegation_reception]

- Headers

Name	值	描述
Content-Type	application/json	消息正文为 json 格式。
授权	bearer [token]	当身份验证实体通知 Red Hat Single Sign-On 后，会使用 [token]。

- 参数

类型	Name	描述
路径	delegation_reception	身份验证实体提供的端点来接收委派请求

- Body

Name	描述

Name	描述
login_hint	它告知身份验证由 AD 验证的实体。 默认情况下，它是用户的"用户名"。 此字段是必需的，由 CIBA 标准文档定义。
scope	它告知身份验证实体从经过身份验证的用户获得同意的范围。 此字段是必需的，由 CIBA 标准文档定义。
is_consent_required	它显示身份验证实体是否需要获得关于范围经过身份验证的用户的同意。 此字段是必需的。
binding_message	它的值应该在 CD 和 AD 的 UI 中显示，以便用户识别 AD 的身份验证是由 CD 触发的。 这个字段是可选的，由 CIBA 标准文档定义。
acr_values	它告诉从 CD 请求身份验证上下文参考。 这个字段是可选的，由 CIBA 标准文档定义。

身份验证委派响应

响应从身份验证实体返回到 Red Hat Single Sign-On，以通知身份验证实体从 Red Hat Single Sign-On 收到身份验证请求。

- 响应

HTTP 状态代码	描述
201	它通知红帽单点登录以接收身份验证委派请求。

身份验证结果通知/ACK 包括以下消息传递：

身份验证结果通知

身份验证实体将身份验证请求的结果发送到 Red Hat Single Sign-On。

```
POST /auth/realms/[realm]/protocol/openid-connect/ext/ciba/auth/callback
```

- Headers

Name	值	描述
Content-Type	application/json	消息正文为 json 格式。
授权	bearer [token]	[token] 必须是从红帽单点登录请求中接收的身份验证实体的一个身份。

- 参数

类型	Name	描述
路径	realm	realm 名称

- Body

Name	描述
status	它告知 AD 进行用户身份验证的结果。 它必须是以下状态之一： SUCCEED： AD 的身份验证已成功完成。 UNAUTHORIZED： AD 的身份验证尚未完成。 CANCELLED： AD 的身份验证已被用户取消。

身份验证结果 ACK

从 Red Hat Single Sign-On 返回响应到身份验证实体，通知 Red Hat Single Sign-On 接收来自身份验证实体的结果。

- 响应

HTTP 状态代码	描述
200	它通知了接收验证结果通知的身份验证实体。

10.1.1.6.4. 用户解析器供应商

即使同一用户，其表示在每个 CD 中可能有所不同，Red Hat Single Sign-On 和身份验证实体也会不同。

对于 CD，Red Hat Single Sign-On 和身份验证实体可识别同一用户，这个用户解析器提供者会在其中转换自己的用户表示法。

用户 Resolver Provider 作为 SPI 提供程序提供，以便 Red Hat Single Sign-On 的用户可以实施自己的供应商以满足其环境。红帽单点登录提供名为 Default User Resolver Provider 的默认提供程序，它具有以下特征：

- 仅支持 **login_hint** 参数，默认使用。
- Red Hat Single Sign-On 中的 UserModel **用户名**用于代表 CD 中的用户、Red Hat Sign-On 和身份验证实体。

10.1.2. OIDC Logout

OIDC 有四个与注销机制相关的规格。这些规格采用草案状态：

1. 会话管理
2. RP-Initiated Logout

3. front-Channel Logout
4. back-Channel Logout

由于在 OIDC 规格中描述了所有这些操作，因此我们只在此处给出一个简略概述。

10.1.2.1. 会话管理

这是基于浏览器的注销。该应用定期从 Red Hat Single Sign-On 获取会话状态信息。当会话在 Red Hat Single Sign-On 中终止时，应用程序将注意到并触发它自己的注销。

10.1.2.2. RP-Initiated Logout

这也是一个基于浏览器的注销，从该注销开始，首先将用户重定向到 Red Hat Single Sign-On 的特定端点。当用户点击部分应用程序页面上的 **Log Out** 链接时，通常使用 Red Hat Single Sign-On 来验证用户时，通常会进行此重定向。

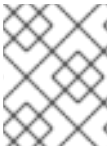
用户重新定向到注销端点后，Red Hat Single Sign-On 将向客户端发送注销请求，以便他们使他们无法使本地用户会话无效，并可能会在注销过程完成后将用户重定向到一些 URL。用户可能会被选择请求，在不使用 **id_token_hint** 参数时确认注销。注销后，只要它作为参数提供，用户会自动重定向到指定的 **post_logout_redirect_uri**。请注意，在包含 **post_logout_redirect_uri** 时，您需要包含 **client_id** 或 **id_token_hint** 参数。另外，**post_logout_redirect_uri** 参数需要匹配客户端 **配置中指定的 Valid Post Logout Redirect URI** 之一。

根据客户端配置，注销请求可以通过前端频道或通过后端通道发送到客户端。对于 frontend 浏览器客户端（依赖于上一节中所述的 Session Management），Red Hat Single Sign-On 不需要向它们发送任何注销请求；这些客户端会自动检测到浏览器中的 SSO 会话。

10.1.2.3. Frontchannel Logout

要将客户端配置为通过前端通道接收登出请求，请查看 Front-Channel Logout 客户端设置。使用此方法时，请考虑以下事项：

- 由 Red Hat Single Sign-On 发送至客户端的登出请求依赖于浏览器，内嵌的 **iframes** 被呈现为登出页。
- 通过基于 **iframes**，前端注销可能会受到内容安全策略(CSP)和注销请求的影响。
- 如果用户在呈现注销页面或注销请求实际发送到客户端之前关闭浏览器，则客户端的会话可能无效。



注意

考虑使用 Back-Channel Logout，因为它提供了一个更加可靠且安全的方法来注销用户并终止其在客户端的会话。

如果没有通过前端注销启用客户端，则红帽单点登录将首先尝试使用 Back-Channel Logout URL 通过 back-Channel Logout URL 发送注销请求。如果没有定义，服务器将回退到使用 Admin URL。

10.1.2.4. Backchannel Logout

这是一个非基于浏览器的注销，使用 Red Hat Single Sign-On 和客户端之间的直接后端通信。Red Hat Single Sign-On 发送包含注销令牌的 HTTP POST 请求到登录到 Red Hat Single Sign-On 的所有客户端。这些请求发送到 Red Hat Single Sign-On 中的注册的 backchannel logout URL，并在客户端一侧触发注销。

10.1.3. Red Hat Single Sign-On 服务器 OIDC URI 端点

以下是 Red Hat Single Sign-On 发布的 OIDC 端点列表。当非 Red Hat Single Sign-On 客户端适配器使用这些端点来与身份验证服务器通信时，可以使用这些端点。它们都是相对的 URL。URL 根由 HTTP (S) 协议、主机名和可选的路径组成：例如

```
https://localhost:8080/auth
```

/realms/{realm-name}/protocol/openid-connect/auth

用于获取授权代码流中的临时代码，或使用 Implicit Flow、Direct Grants 或 Client Grants 获取令牌。

/realms/{realm-name}/protocol/openid-connect/token

授权代码流用于将临时代码转换为令牌。

/realms/{realm-name}/protocol/openid-connect/logout

用于执行注销。

/realms/{realm-name}/protocol/openid-connect/userinfo

用于 OIDC 规格中描述的 User Info 服务。

/realms/{realm-name}/protocol/openid-connect/revoke

用于 OAuth 2.0 令牌吊销(RFC7009 所述)。

/realms/{realm-name}/protocol/openid-connect/certs

用于 JSON Web Key Set (JWKS)，包含用于验证任何 JSON Web Token (jwks_uri)的公钥。

/realms/{realm-name}/protocol/openid-connect/auth/device

用于设备授权授权来获取设备代码和用户代码。

/realms/{realm-name}/protocol/openid-connect/ext/ciba/auth

这是 Client Initiated Backchannel Authentication Grant 的 URL 端点，以获取 auth_req_id，用于标识客户端发出的身份验证请求。

/realms/{realm-name}/protocol/openid-connect/logout/backchannel-logout

这是执行后端的 URL 端点，在 OIDC 规格中描述。

在所有这些情况下，将 {realm-name} 替换为域的名称。

10.2. SAML

SAML 2.0 与 OIDC 类似，但更成熟。它从 SOAP 和 Web 服务消息传递规格中分离，因此通常比 OIDC 更详细。SAML 2.0 是一个身份验证协议，可在身份验证服务器和应用程序间交换 XML 文档。XML 签名和加密用于验证请求和响应。

通常，SAML 实施两个用例。

第一个用例是请求 Red Hat Single Sign-On 服务器验证用户的应用程序。成功登录后，应用程序将收到 XML 文档。本文档包含指定用户属性的 SAML 断言。域对文档进行数字签名，该文档包含应用用于访问应用的访问信息（如用户角色映射）。

第二个用例是访问远程服务的客户端。客户端请求来自 Red Hat Single Sign-On 的 SAML 断言，以代表用户调用远程服务。

10.2.1. SAML 绑定

红帽单点登录支持三种绑定类型：

10.2.1.1. 重定向绑定

重定向 绑定使用一系列浏览器重定向 URI 来交换信息。

1. 用户使用浏览器连接至应用。应用程序检测到用户没有通过身份验证。
2. 应用程序生成 XML 身份验证请求文档，并将其编码为 URI 中的查询参数。URI 用于重定向到 Red Hat Single Sign-On 服务器。根据您的设置，应用程序也可以对 XML 文档进行数字签名，并将签名作为查询参数包含在重定向 URI 到 Red Hat Single Sign-On 中。此签名用于验证发送请求的客户端。
3. 浏览器重定向到 Red Hat Single Sign-On。
4. 如果需要，服务器提取 XML 身份验证请求文档并验证数字签名。
5. 用户输入其身份验证凭据。
6. 身份验证后，服务器会生成 XML 身份验证响应文档。本文档包含包含用户的 SAML 断言，其中包含有关用户名、地址、电子邮件以及用户所拥有的角色映射的元数据。文档通常使用 XML 签名进行数字签名，也可以加密。
7. XML 身份验证响应文档在重定向 URI 中以查询参数形式编码。URI 使浏览器返回到应用程序。数字签名还包含为查询参数。
8. 应用程序接收重定向 URI，并提取 XML 文档。
9. 应用程序会验证 realm 的签名，以确保它收到有效的身份验证响应。SAML 断言内的信息用于做出访问决策或显示用户数据。

10.2.1.2. POST 绑定

POST 绑定与 *Redirect* 绑定类似，但 *POST* 绑定使用 POST 请求交换 XML 文档，而不是使用 GET 请求。*POST* Binding 使用 JavaScript 使浏览器在交换文档时向 Red Hat Single Sign-On 服务器或应用程序发送 POST 请求。HTTP 使用 HTML 文档响应，其中包含有嵌入式 JavaScript 的 HTML 表单。当页面加载时，JavaScript 会自动调用表单。

建议因为两个限制而使用 *POST* 绑定：

- **安全** baseDomain-sandboxedWith *Redirect* 绑定，SAML 响应是 URL 的一部分。不太安全，因为可以在日志中捕获响应。
- 在 HTTP 有效负载中调整文档的大小提供了比在有限 URL 中大量数据的范围。

10.2.1.3. ECP

增强的客户端或代理(ECP)是一个 SAML v.2.0 配置集，它允许在 Web 浏览器上下文外交换 SAML 属性。它通常由 REST 或 SOAP 的客户端使用。

10.2.2. Red Hat Single Sign-On Server SAML URI Endpoints

Red Hat Single Sign-On 对所有 SAML 请求都有一个端点。

http(s)://authserver.host/auth/realms/{realm-name}/protocol/saml

所有绑定都使用此端点。

10.3. 与 SAML 相比 OPENID CONNECT

下表列出了在选择协议时需要考虑的诸多因素。

在大多数情况下，Red Hat Single Sign-On 建议使用 OIDC。

OIDC

- OIDC 专门用于使用 Web。
- OIDC 适用于 HTML5/JavaScript 应用程序，因为比 SAML 更容易实现。
- OIDC 令牌采用 JSON 格式，使其更易于使用 Javascript。
- OIDC 具有简化安全实施的功能。例如，请查看规格用于决定用户登录状态的 iframe 难度。

SAML

- SAML 设计为一个层，可在 web 之上工作。
- SAML 比 OIDC 更详细。
- 用户通过 OIDC 选择 SAML，因为存在成熟问题。
- 用户通过 OIDC 对其安全保护的现有应用程序选择 SAML。

10.4. DOCKER REGISTRY V2 身份验证

注意

Docker 身份验证默认为禁用。要启用 docker 身份验证，请查看 配置集。

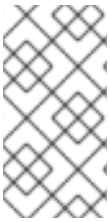
Docker Registry V2 身份验证是一个与 OIDC 类似，用于针对 Docker registry 验证用户的协议。红帽单点登录实现此协议可让 Docker 客户端使用 Red Hat Single Sign-On 身份验证服务器与注册表进行身份验证。这个协议使用标准令牌和签名机制，但它与真正的 OIDC 实现不同。它通过对请求和响应使用非常具体的 JSON 格式，并将存储库名称和权限映射到 OAuth 范围机制。

10.4.1. Docker 身份验证流程

Docker API 文档 中介绍了身份验证流程。以下是 Red Hat Single Sign-On 身份验证服务器摘要：

- 执行 Docker 登录。
- Docker 客户端从 Docker 注册表请求资源。如果资源受到保护，且没有在请求中，Docker 注册表服务器使用 401 HTTP 消息响应 401 HTTP 消息，其中包含一些有关所需权限的信息以及授权服务器的位置。
- Docker 客户端根据 Docker 注册表中的 401 HTTP 消息构建身份验证请求。客户端使用本地缓存的凭据（从 **docker login** 命令）作为发送到 Red Hat Single Sign-On 身份验证服务器的HTTP 基本身份验证 请求的一部分。
- Red Hat Single Sign-On 身份验证服务器会尝试验证用户，并返回包含 OAuth 风格的 Bearer 令牌的 JSON 正文。
- Docker 客户端从 JSON 响应中收到 bearer 令牌，并在授权标头中使用它请求受保护的资源。

- Docker 注册表从 Red Hat Single Sign-On 服务器获取受保护资源的新请求。registry 验证令牌并授予对请求资源的访问权限（如果适用）。



注意

在成功使用 Docker 协议进行身份验证后，Red Hat Single Sign-On 不会创建浏览器 SSO 会话。浏览器 SSO 会话不使用 Docker 协议，因为它无法刷新令牌或从 Red Hat Single Sign-On 服务器获取令牌或会话状态，因此不需要浏览器 SSO 会话。如需了解更多详细信息，请参阅 临时会话 部分。

10.4.2. Red Hat Single Sign-On Docker Registry v2 Authentication Server URI 端点

Red Hat Single Sign-On 对所有 Docker auth v2 请求都有一个端点。

http(s)://authserver.host/auth/realms/{realm-name}/protocol/docker-v2

第 11 章 控制对管理控制台的访问

在 Red Hat Single Sign-On 上创建的每个域都有一个专用管理控制台，该控制台可从中管理该域。主 域是一个特殊的域，它允许管理员管理系统上的多个域。您还可以定义不同域中用户的精细访问来管理服务器。本章在这款方案中都进行了介绍。

11.1. MASTER REALM 访问控制

Red Hat Single Sign-On 中的 master 域是一个特殊的域，它和其它域不同。Red Hat Single Sign-On master 域中的用户有权管理在 Red Hat Single Sign-On 服务器上部署的零个或更多域。创建域后，Red Hat Single Sign-On 会自动创建不同的角色，授予精细权限来访问该新域。可以通过将这些角色映射到主域中的用户来控制管理控制台和 Admin REST 端点 的访问权限。可以创建多个超级用户，以及只能管理特定域的用户。

11.1.1. 全局角色

master 域中有两个 realm 级别的角色。以下是：

- admin
- create-realm

具有 admin 角色的用户是超级用户，并具有完全访问权限来管理服务器上的任何域。具有 create-realm 角色的用户可以创建新的域。它们将被授予他们所创建的任何新域的完全访问权限。

11.1.2. realm 特定角色

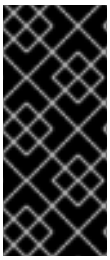
master realm 中的 admin 用户可以获得系统的一个或多个其他域的管理权限。Red Hat Single Sign-On 中的每个域都由主域中的客户端表示。客户端的名称为 < realm name>-realm。这些客户端各自定义了客户端级别的角色，定义不同的访问级别来管理单个域。

可用的角色有：

- view-realm
- view-users
- view-clients
- view-events
- manage-realm

- manage-users
- create-client
- manage-clients
- manage-events
- view-identity-providers
- manage-identity-providers
- 模拟

为用户分配您要的角色，并且他们只能使用管理控制台中的特定部分。



重要

具有 manage-users 角色的管理员只能将 admin 角色分配给他们自己具有的用户。因此，如果管理员具有 manage-users 角色，但没有 manage-realm 角色，则无法分配此角色。

11.2. 专用域管理控制台

每个域都有一个专用管理控制台，可通过转至 url /auth/admin/{realm-name}/console 进行访问。通过分配特定的用户角色映射，可以授予该域管理权限。

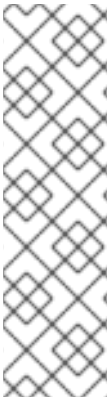
每个 realm 都有一个内置客户端，称为 realm-management。您可以通过转至域的 客户端 左侧菜单项来查看此客户端。此客户端定义客户端级角色，用于指定可以授予管理该域的权限。

- view-realm

- view-users
- view-clients
- view-events
- manage-realm
- manage-users
- create-client
- manage-clients
- manage-events
- view-identity-providers
- manage-identity-providers
- 模拟

为用户分配您要的角色，并且他们只能使用管理控制台中的特定部分。

11.3. 精细的管理权限



注意

精细管理权限 是技术预览，不被完全支持。此功能默认为禁用。

使用 -Dkeycloak.profile=preview 或 -Dkeycloak.profile.feature.admin_fine_grained_authz=enabled 来启用服务器。如需了解更多详细信息，请参阅 配置文件。

有时，management-realm 或 manage-users 等角色过于起见，而且您希望创建具有更细化权限的受限 admin 帐户。Red Hat Single Sign-On 允许您定义和分配用于管理域的受限访问策略。比如：

- 管理一个特定客户端
- 管理属于特定组的用户
- 管理组成员资格
- 有限的用户管理。
- 精细模拟控制
- 可以为用户分配一组特定的受限角色。
- 能够将特定的受限角色集合分配给复合角色。
- 能够将特定的受限角色集合分配给客户端的范围。
- 用于查看和管理用户、组、角色和客户端的新常规策略。

请注意有关精细的 admin 权限的一些重要事项：

- 细致的管理权限是在授权服务之上 实施。强烈建议您先读取这些功能，然后再进入相关权限。

- 精细的权限仅在 专用 管理控制台和那些域中定义的管理员中可用。您无法定义跨域细调权限。

- 细粒度权限用于授予额外权限。您无法覆盖在 admin 角色中构建的默认行为。

11.3.1. 管理一个特定客户端

首先，让我们看一个管理员仅管理一个客户端和一个客户端。在我们的示例中，我们有一个名为 test 的域，以及名为 sales-application 的客户端。在域测试中，我们将为该域权限授予用户，以便仅管理该应用程序。

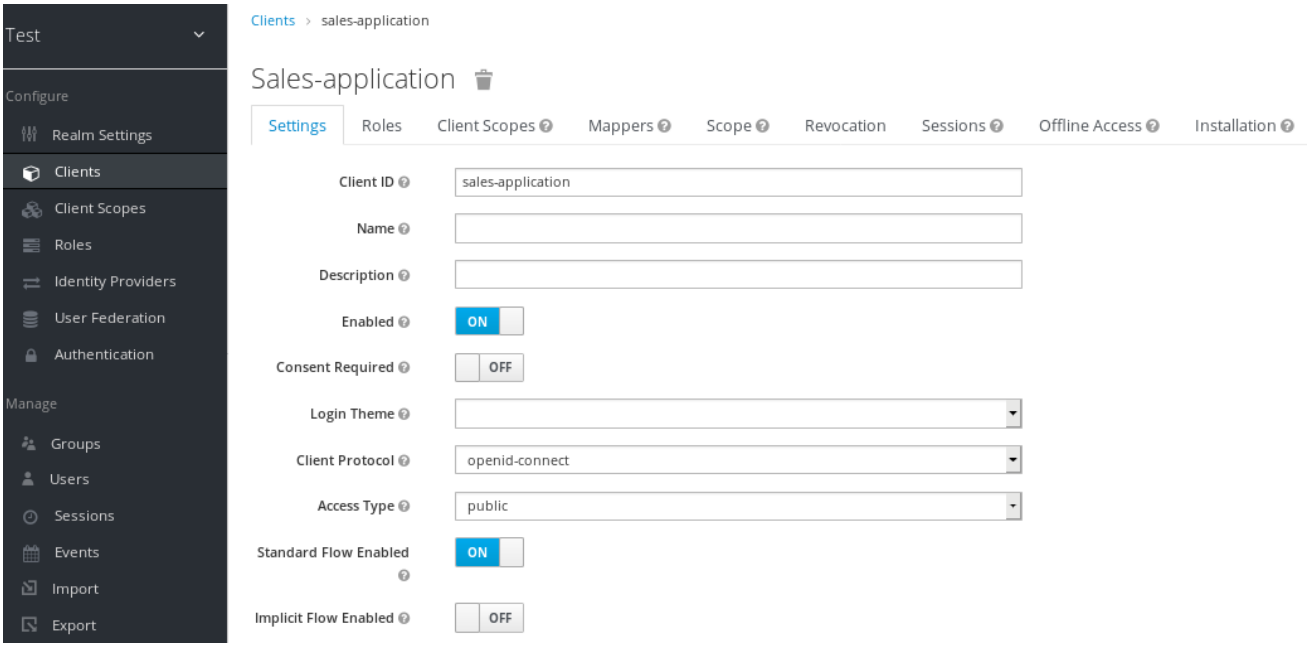
重要

您不能进行跨域细致权限。master 域中的管理员仅限于前面章节中定义的预定义 admin 角色。

11.3.1.1. 权限设置

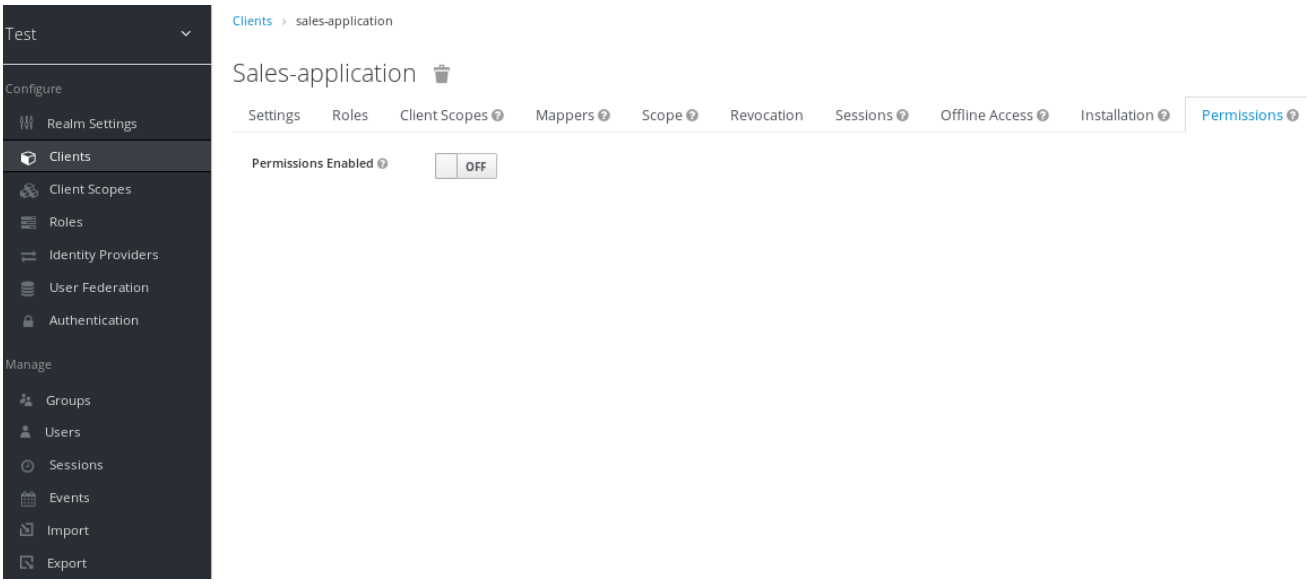
我们首先必须登录 Admin Console，以便我们能够为该客户端设置权限。我们导航到客户端的管理部分，为.

客户端管理

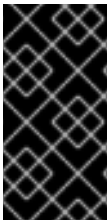


您应看到一个名为 Permissions 的选项卡菜单项。点击该选项卡。

客户端权限选项卡



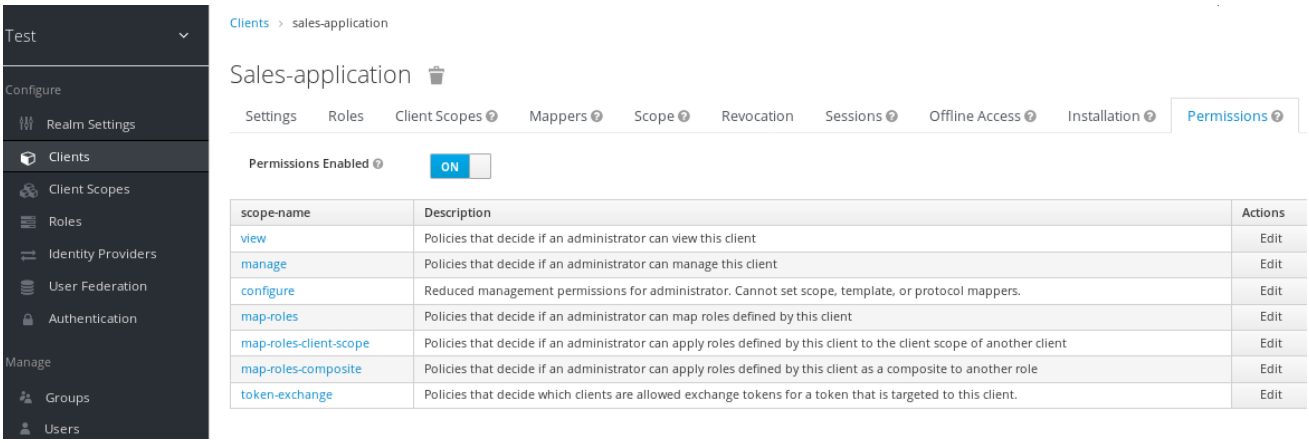
默认情况下，每个客户端都未启用才能进行细化权限。因此，将 Permissions Enabled 切换到 on 以初始化权限。



重要

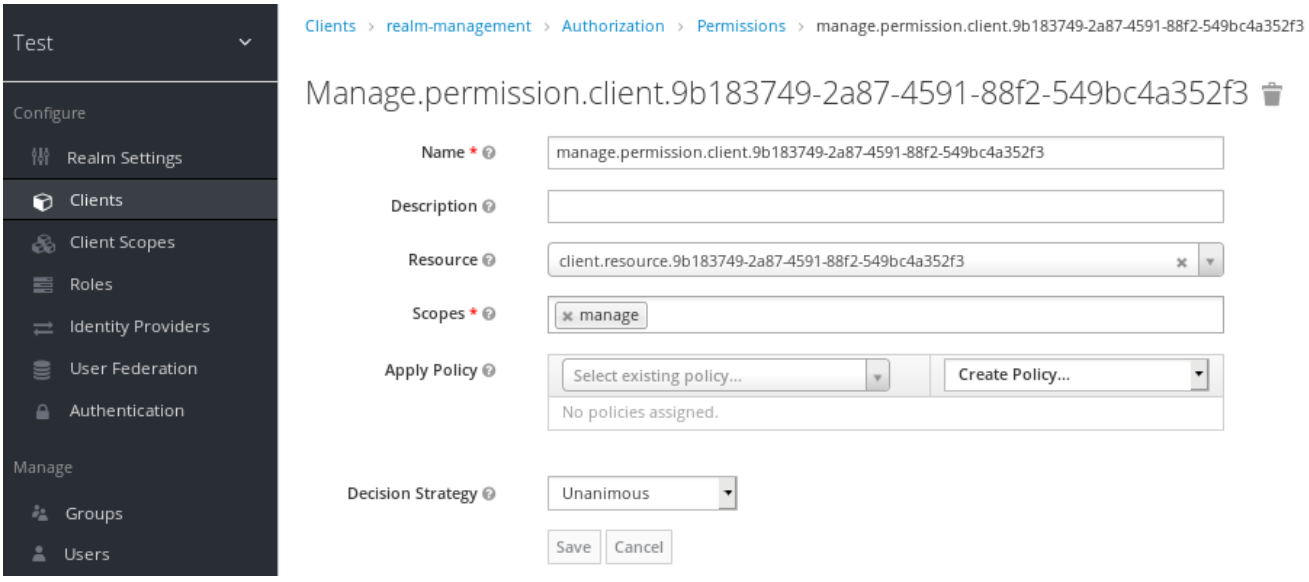
如果将 Permissions Enabled 切换为 off，它将删除任何权限以及您为这个客户端定义的所有权限。

客户端权限选项卡



当您将 Permissions Enabled 切换到 on 时，它会使用授权服务 初始化幕后的各种权限对象。https://access.redhat.com/documentation/zh-cn/red_hat_single_sign-on/7.6/html-single/authorization_services_guide/在本示例中，我们对客户端 的管理权限 感兴趣。单击该按钮，将您重定向到处理客户端 管理权限 的权限。所有授权对象都包含在 realm-management 客户端的 Authorization 选项卡中。

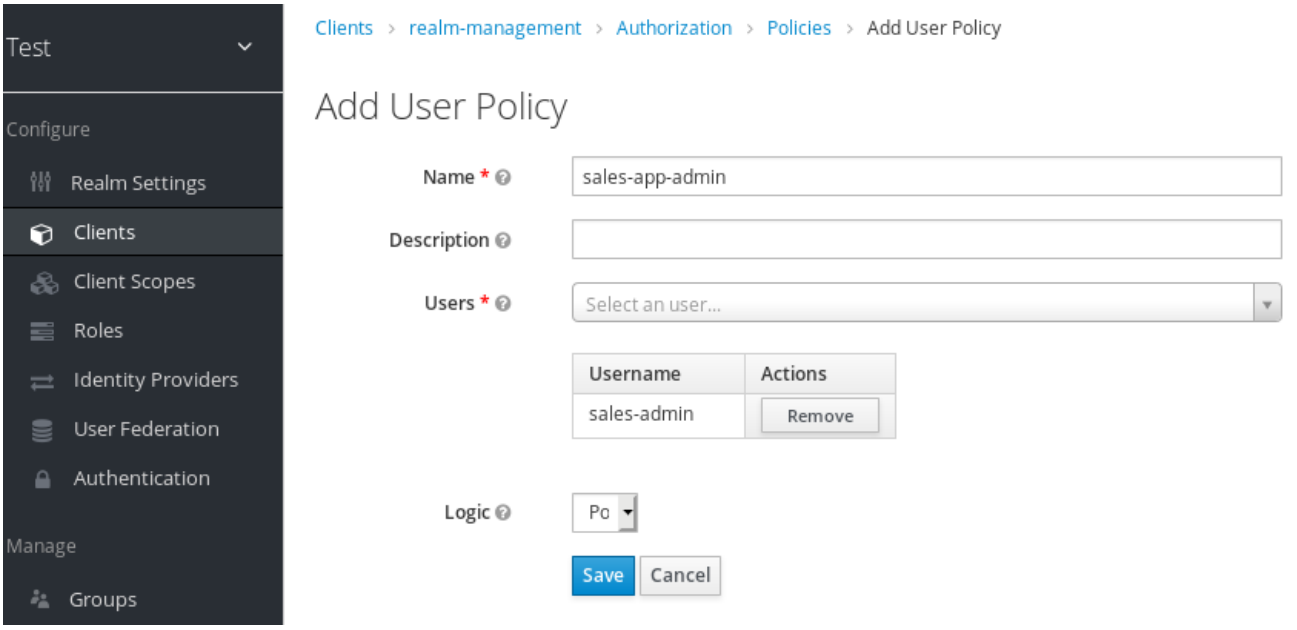
客户端管理权限



当第一次初始化时，管理权限没有关联任何策略。您将需要创建一个策略选项卡。若要快速获取，可单击以上镜像中显示的 授权 链接。然后点击 policy 选项卡。

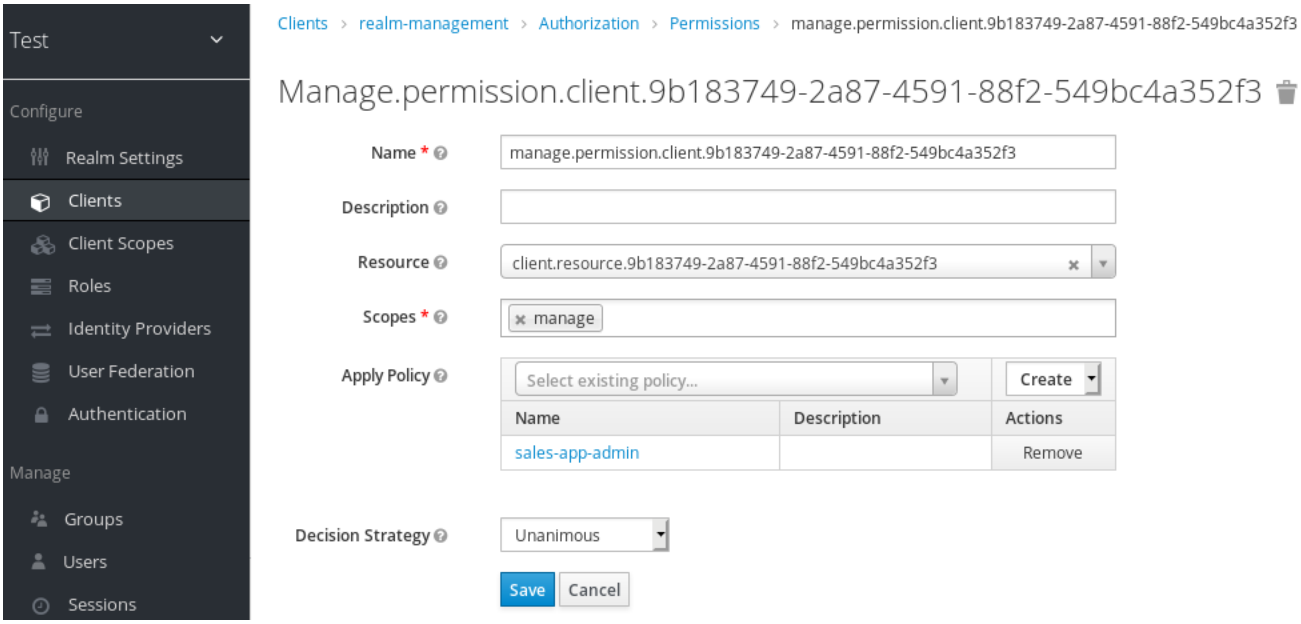
本页中有一个名为 Create policy 的下拉菜单。您可以定义多个策略。您可以定义与角色或组关联的策略，或者在 JavaScript 中定义规则。对于这个简单示例，我们将 创建用户策略。

用户策略



此策略将与用户数据库中硬编码的用户匹配。在本例中，它是 sales-admin 用户。然后，我们必须返回 sales-application 客户端的管理权限页面，并将策略分配给权限对象。

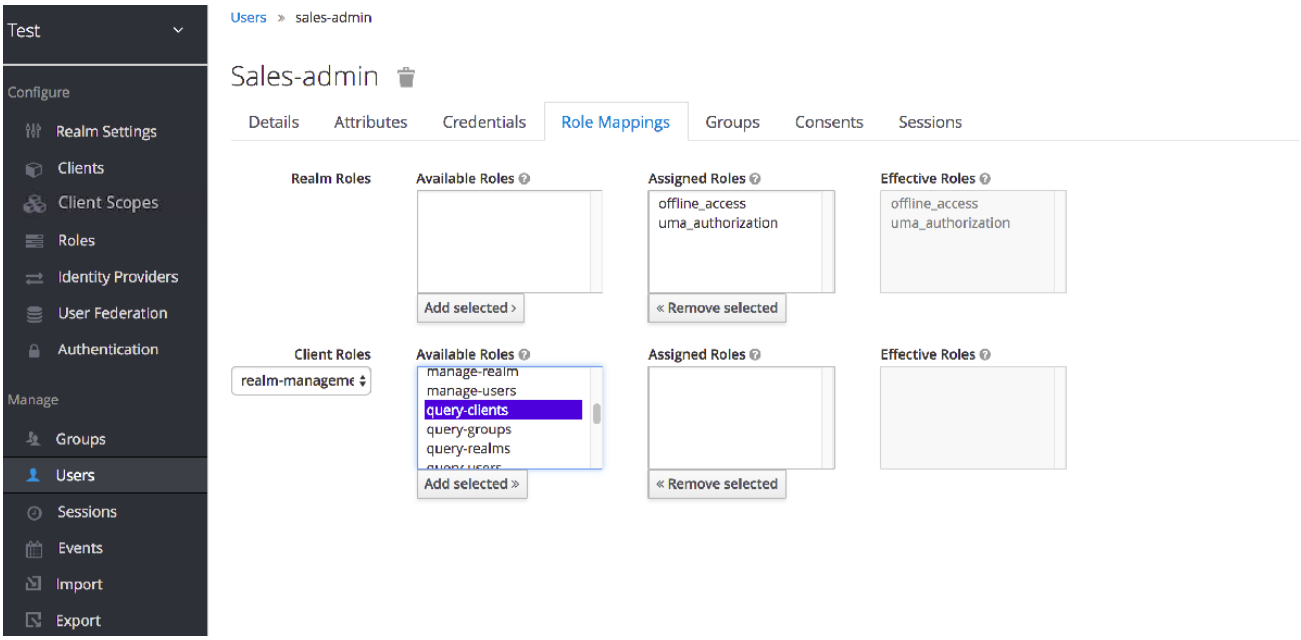
分配用户策略



sales-admin 用户现在可以有管理 sales-application 客户端的权限。

我们还需要做得再做。前往 Role Mappings 选项卡，并将 query-clients 角色分配给 sales-admin。

分配查询客户端



您为何要执行此操作？此角色告知 Admin Console 哪个菜单项目在 sales-admin 访问管理控制台时要呈现出来。query-clients 角色告知 Admin Console，它应该为 sales-admin 用户呈现客户端菜单。

重要信息如果没有设置 query-clients 角色，则 sales-admin 等受限管理员不会在用户登录管理控制台时看到任何菜单选项

11.3.1.2. 测试它

接下来，我们从 master 域注销，并使用 Sales - admin 作为用户名重新登录到 测试 域的专用 管理控制台。它位于 /auth/admin/test/console 下。

销售管理员登录

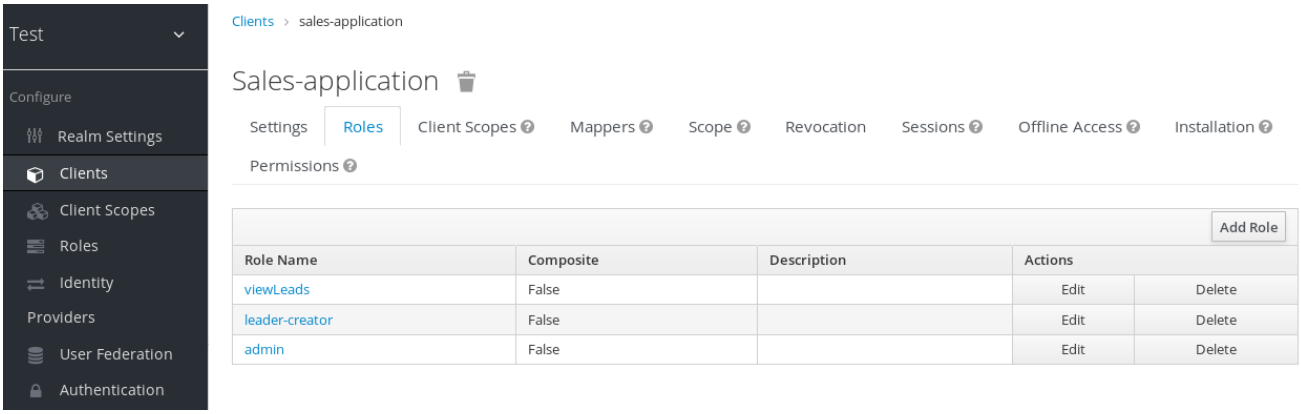
此管理员现在可以管理这个一个客户端。

11.3.2. 限制用户角色映射

您想要做的另一个操作是限制管理员被允许分配给用户的一组角色。继续我们的最后一个示例，让我们扩展"销售管理员"用户的权限集，以便他还能够控制允许哪些用户访问此应用程序。通过精细的权限，我们可以启用它，以便 sales-admin 只能分配授予对 销售应用程序 特定访问权限的角色。我们还可以限制它，使得管理员只能映射角色，而不执行任何其他用户管理类型。

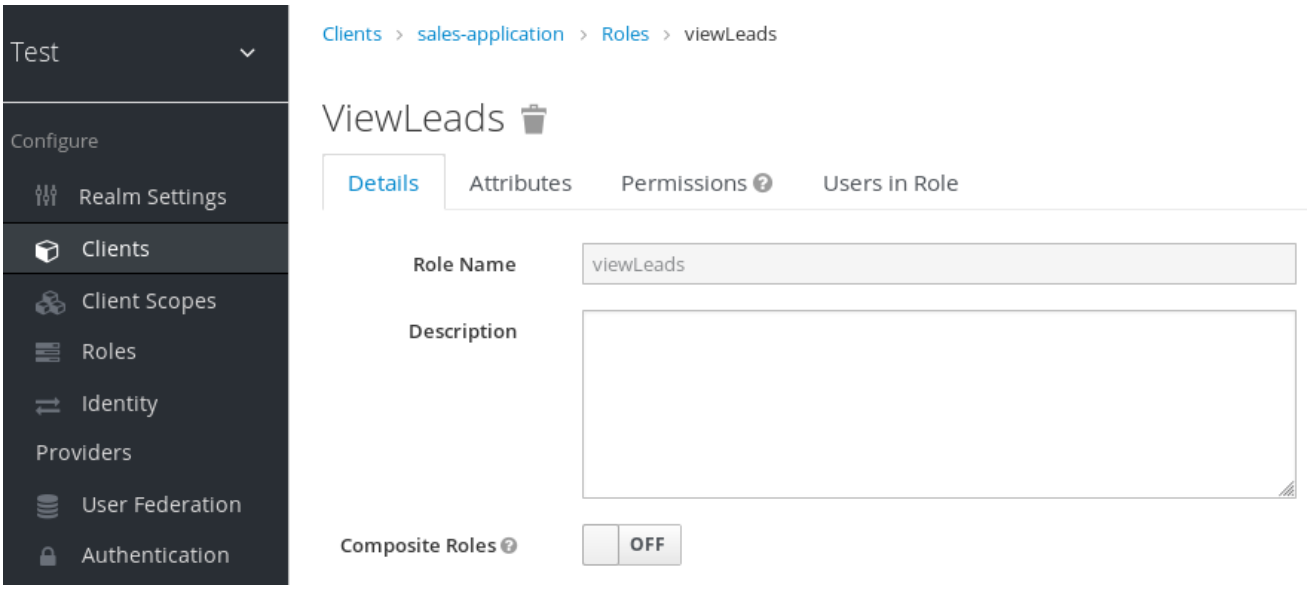
sales-application 定义了三个不同的客户端角色。

销售应用程序角色



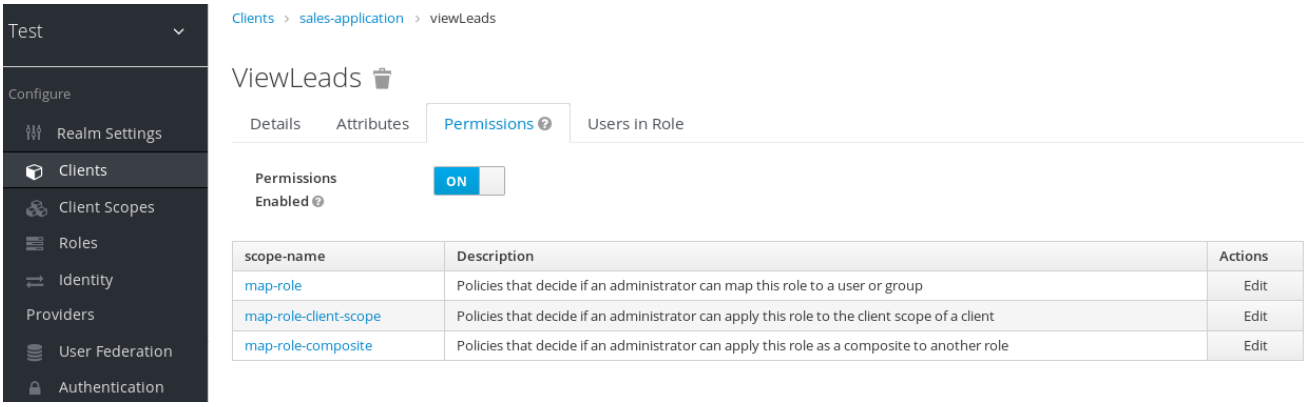
我们希望 sales-admin 用户能够将这些角色映射到系统中的任何用户。执行此操作的第一步是允许 admin 映射角色。如果点击 viewLeads 角色，您会看到此角色的 Permissions 选项卡。

查看领导角色权限选项卡



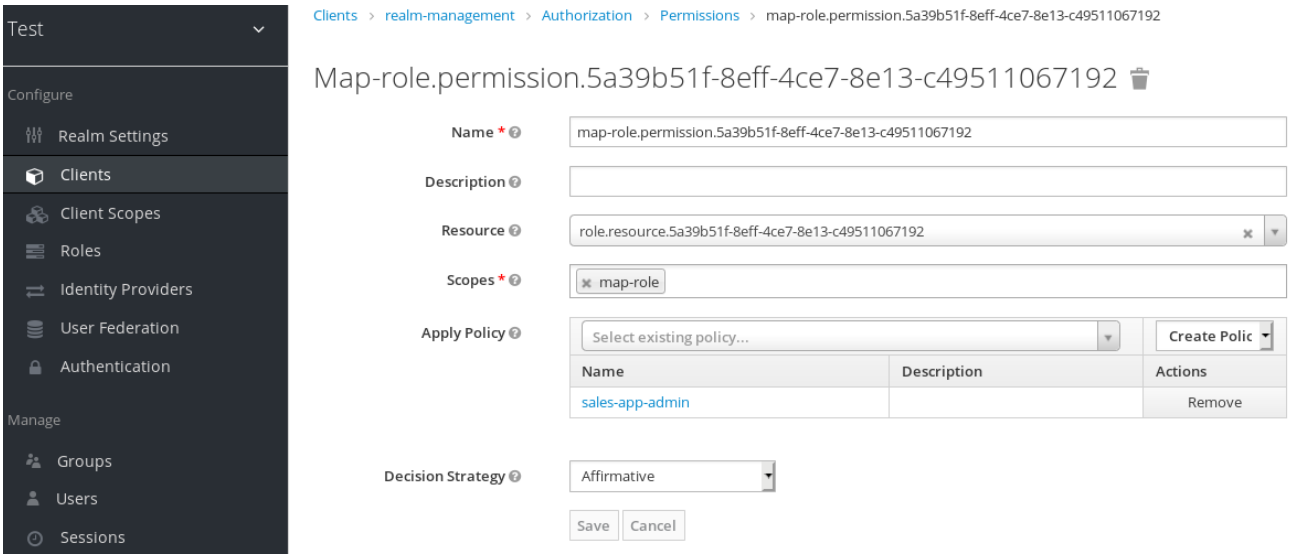
如果我们单击该选项卡并开启了"权限"启用，您会看到我们可以将策略应用于多个操作。

查看领导权限



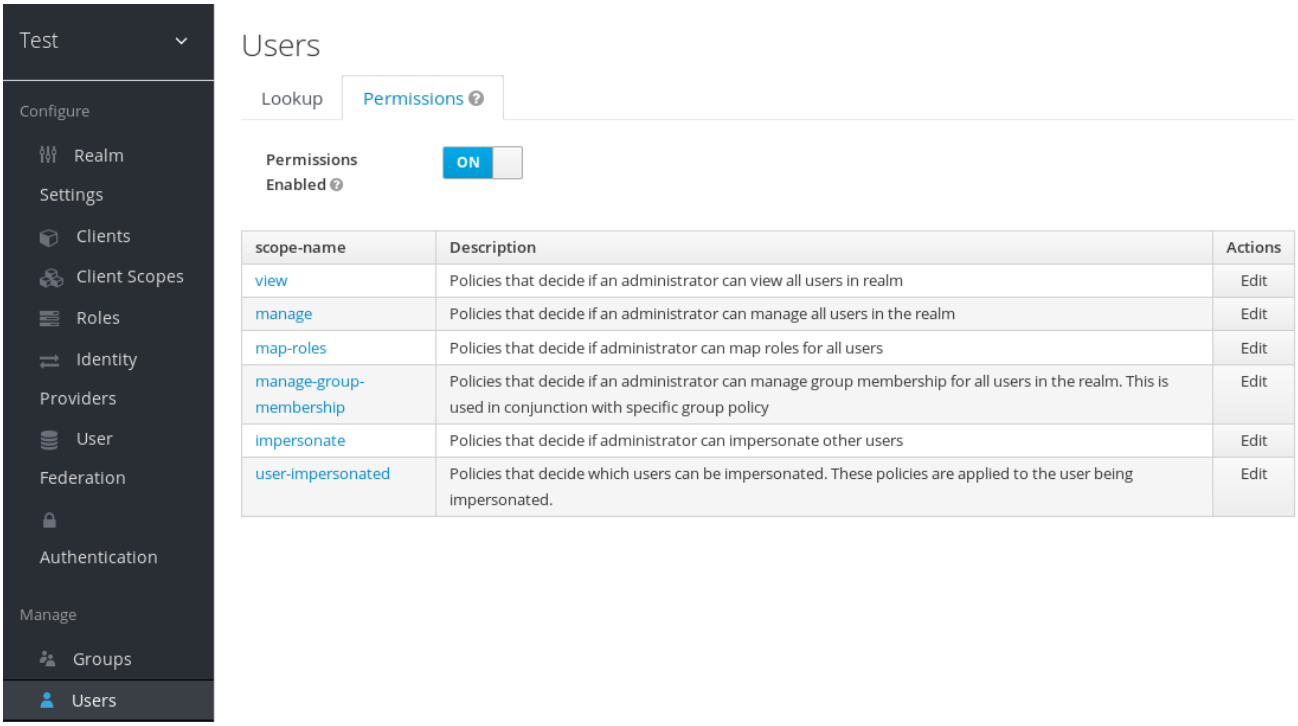
我们感兴趣的其中一项是 映射角色。单击此权限，再添加在先前示例中创建的同一用户策略。

map-roles 权限



我们所执行的操作是，sales-admin 可以映射 viewLeads 角色。我们尚未执行的操作是什么，指定管理员也可以映射此角色的用户。为此，我们必须进入此域的管理控制台的 Users 部分。单击 Users left 菜单项可进入域的用户界面。您应看到一个 权限选项卡。点击这个窗口并启用它。

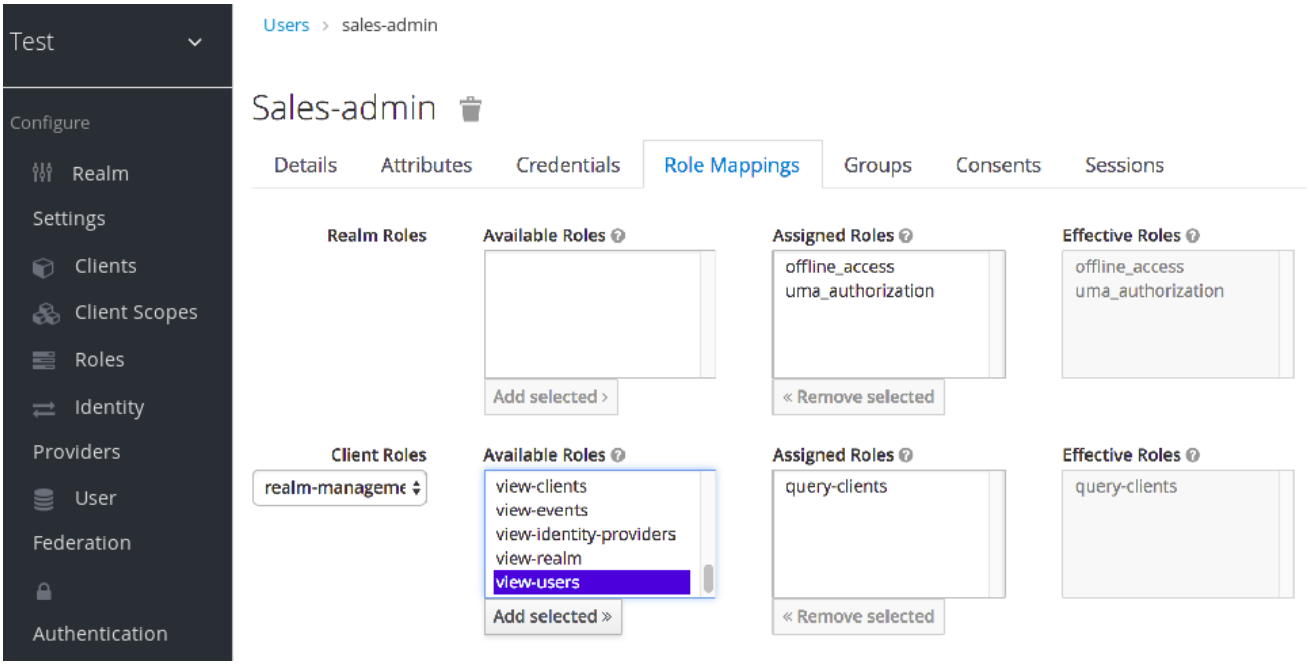
用户权限



我们感兴趣的权限是 map-roles。这是限制性策略，其仅允许管理员将角色映射到用户。如果我们单击 map-roles 权限并再次添加我们为此创建的 User Policy，则 sales-admin 将可以将角色映射到任何用户。

我们要做的最后一件事是将 view-users 角色添加到 sales-admin。这样，管理员可以查看 realm 中的用户，希望将 sales-application 角色添加到其中。

添加 view-users

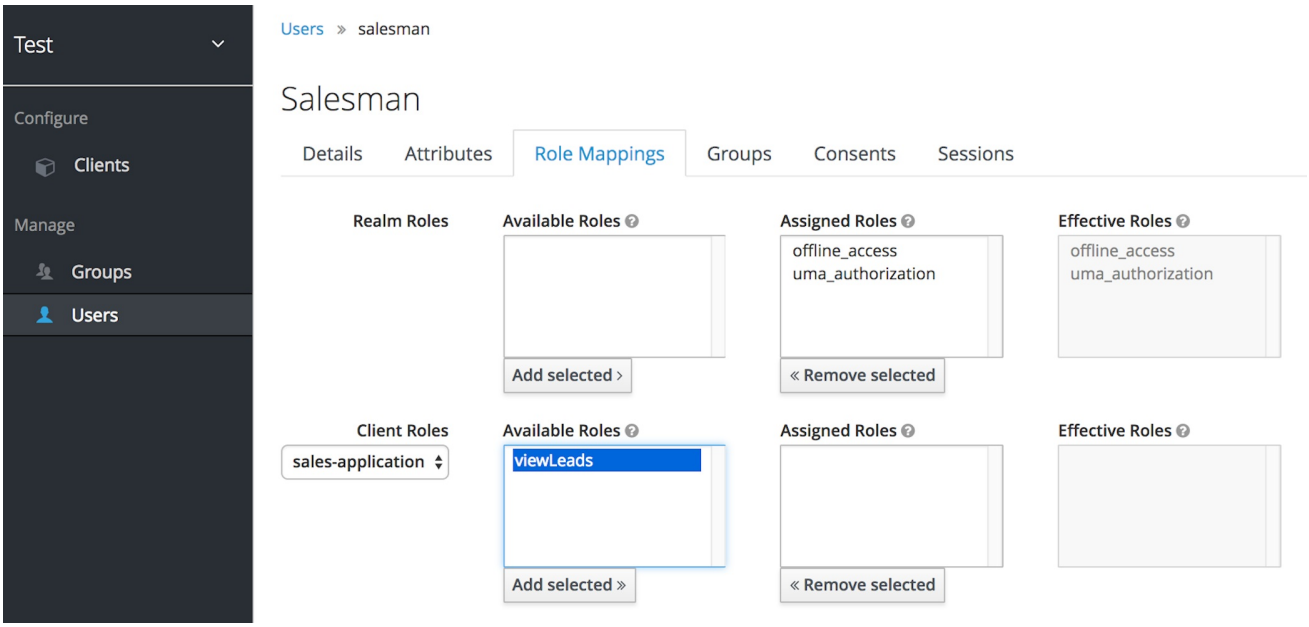


11.3.2.1. 测试它

接下来，我们从 master 域注销，并使用 Sales - admin 作为用户名重新登录到 测试 域的专用 管理控制台。它位于 /auth/admin/test/console 下。

您将看到 sales-admin 可以查看系统中的用户。如果您选择了其中一个用户，您将看到每个用户详情页面都是只读的，但 Role Mappings 选项卡除外。转至此选项卡，您会发现 admin 没有可用的 角色来映射到用户，除非浏览 sales-application 角色时除外。

添加 viewleads

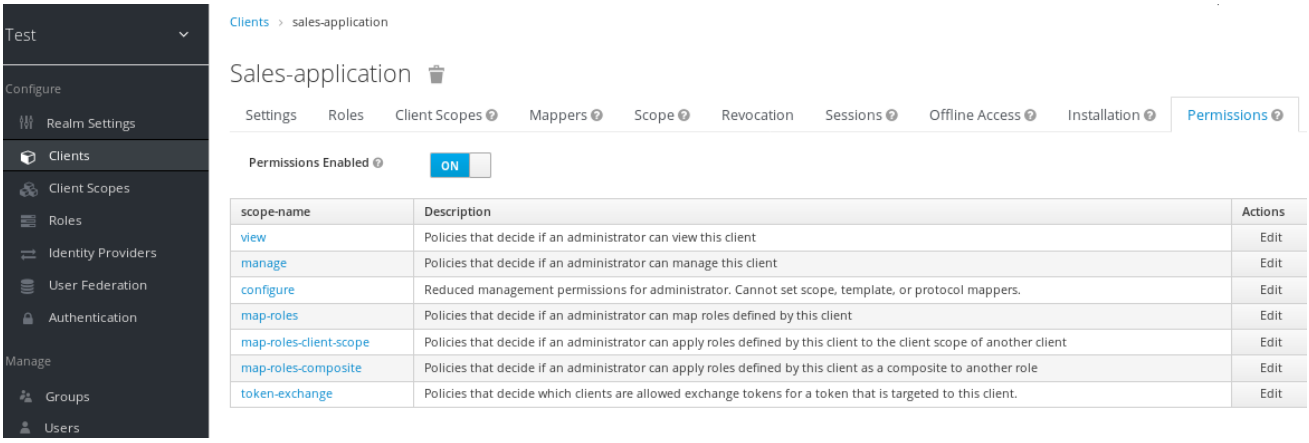


我们仅指定 sales-admin 可以映射 viewLeads 角色。

11.3.2.2. 每个客户端映射 -roles 快捷方式

如果我们需要对发布5发布的每个客户端角色执行此操作，这非常繁琐。为了方便事情，指定管理员可以映射客户端所定义的任何角色。如果我们重新登录到我们的主域管理员管理员并返回 sales-application 权限页，您将看到 map-roles 权限。

client map-roles 权限



如果授予管理员对此特定权限的访问权限，则 admin 能够映射客户端定义的任何角色。

11.3.3. 权限完整列表

除了管理特定客户端或客户端的特定角色外，您还可进行更细化的权限。本章定义域可以描述的完整权限类型列表。

11.3.3.1. 角色

当进入特定角色的 Permissions 选项卡时，您将看到列出的这些权限类型。

map-role

决定管理员是否可以将此角色映射到用户的策略。这些策略仅指定角色可以映射到用户，而 admin 则被允许执行用户角色映射任务。管理员还必须具有管理或角色映射权限。如需更多信息，请参阅 用户权限。

map-role-composite

决定管理员是否可以将此角色映射为复合到另一个角色的策略。如果他必须管理该客户端的权限，则管理员可以定义角色，但是他将无法将复合添加到这些角色，除非他拥有要添加为复合的角色的 map-role-composite 特权。

map-role-client-scope

决定管理员是否可以将此角色应用到客户端范围的策略。即使管理员能够管理客户端，他将有权为该客户端创建包含此角色的令牌，除非授予了此特权。

11.3.3.2. 客户端

当进入特定客户端的 Permissions 选项卡时，您将看到列出的这些权限类型。

view

决定管理员能否查看客户端的配置的策略。

管理

决定管理员是否可以查看和管理客户端的配置的策略。在这种情形中，这可能会意外泄漏一些问题。例如，管理员可以定义一个协议映射程序来硬编码角色，即使管理员没有将角色映射到客户端的范围的权限。目前，协议映射程序的限制，因为它们没有向他们分配独立权限的方法，如角色的作用。

配置

减少管理客户端的特权集合。它与 管理 范围类似，但管理员不允许管理员定义协议映射程序、更改客户端模板或客户端的范围。

map-roles

决定管理员是否可以将客户端定义的任何角色映射到用户的策略。这是一个快捷方式、易于使用的功能，以避免为客户端定义每一个和每一个角色定义策略。

map-roles-composite

决定管理员是否可以将客户端定义为复合到另一个角色的策略。这是一个快捷方式、易于使用的功能，以避免为客户端定义每一个和每一个角色定义策略。

map-roles-client-scope

决定管理员是否可将客户端定义的任何角色映射到另一客户端的范围。这是一个快捷方式、易于使用的功能，以避免为客户端定义每一个和每一个角色定义策略。

11.3.3.3. 用户

进入所有用户的 权限选项卡 时，您将看到列出的这些权限类型。

view

决定管理员能否查看域中的所有用户的策略。

管理

决定管理员是否可以管理域中的所有用户的策略。此权限向 admin 授予执行用户角色映射的特权，但它不指定 admin 允许映射的角色。您需要为希望管理员进行映射的每个角色定义特权。

map-roles

这是 管理 范围授予的特权的子集。在这种情况下，管理员只能映射角色。admin 不允许执行任何其他用户管理操作。另外，如 管理，必须为每个角色指定 admin 应用的角色，或者在处理客户端角色时为每个角色指定每个角色。

manage-group-membership

与 map-roles 类似，但它与组成员资格相关：用户可以从中删除用户的组。这些策略仅授予管理组成员资格的 admin 权限，而不向管理员管理成员资格的组。您必须为每个组的 manage-members 权限指定策略。

模拟

决定管理员是否允许其他用户模拟策略。这些策略应用到管理员的属性和角色映射。

user-impersonated

决定哪些用户可以模拟的策略。这些策略将应用到被模拟的用户。例如，您可能希望定义一个策略来禁止任何人模拟具有管理员特权的用户。

11.3.3.4. 组

当进入特定组的 Permissions 选项卡时，您将看到列出的这些权限类型。

view

决定管理员能否查看该组的相关信息的策略。

管理

决定管理员是否可以管理该组的配置的策略。

view-members

决定管理员能否查看组成员的用户详情的策略。

manage-members

决定管理员是否可以管理属于此组的用户的策略。

manage-membership

决定管理员是否可以更改组成员资格的策略。添加或删除组中的成员。

第 12 章 管理 OPENID CONNECT 和 SAML 客户端

客户端是可以请求用户进行身份验证的实体。客户端以两种形式提供。第一种类型的客户端是希望参与单点登录的应用程序。这些客户端只希望 Red Hat Single Sign-On 为它们提供安全。另一种类型的客户端是请求访问令牌，以便它能够代表经过身份验证的用户调用其他服务。本节讨论配置客户端以及执行它的各种方法。

12.1. OIDC 客户端

OpenID Connect 是用来保护应用程序的推荐协议。它从基础上设计为非常友好，它最适合使用 HTML5/JavaScript 应用程序。

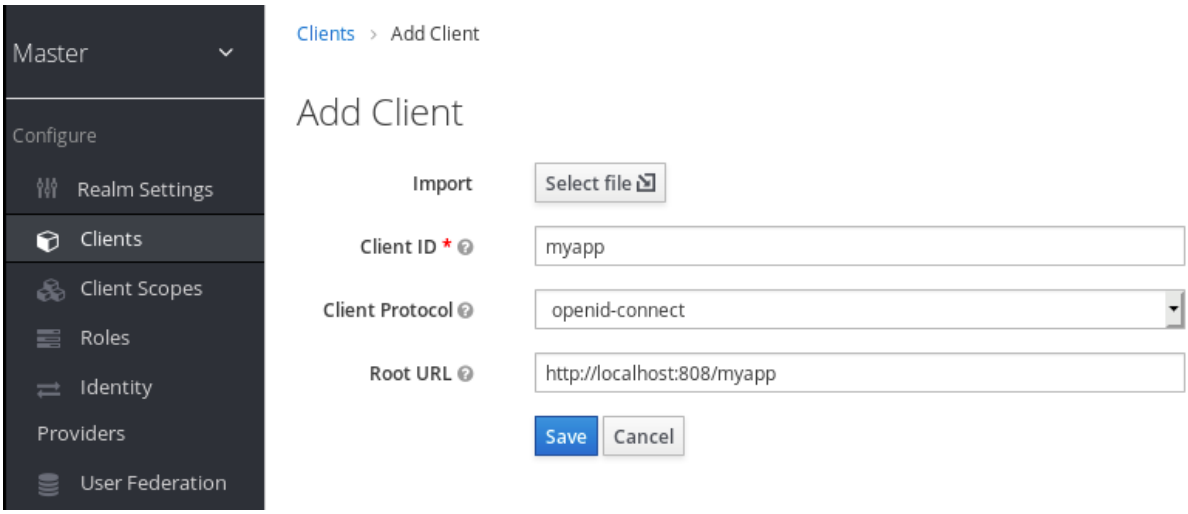
12.1.1. 创建 OpenID Connect 客户端

为了保护使用 OpenID 连接协议的应用程序，您可以创建一个客户端。

流程

1. 点菜单中的 **Clients**。
2. 点 **Create** 进入 **Add Client** 页面。

添加客户端



3. 为客户端 ID 输入任何名称。

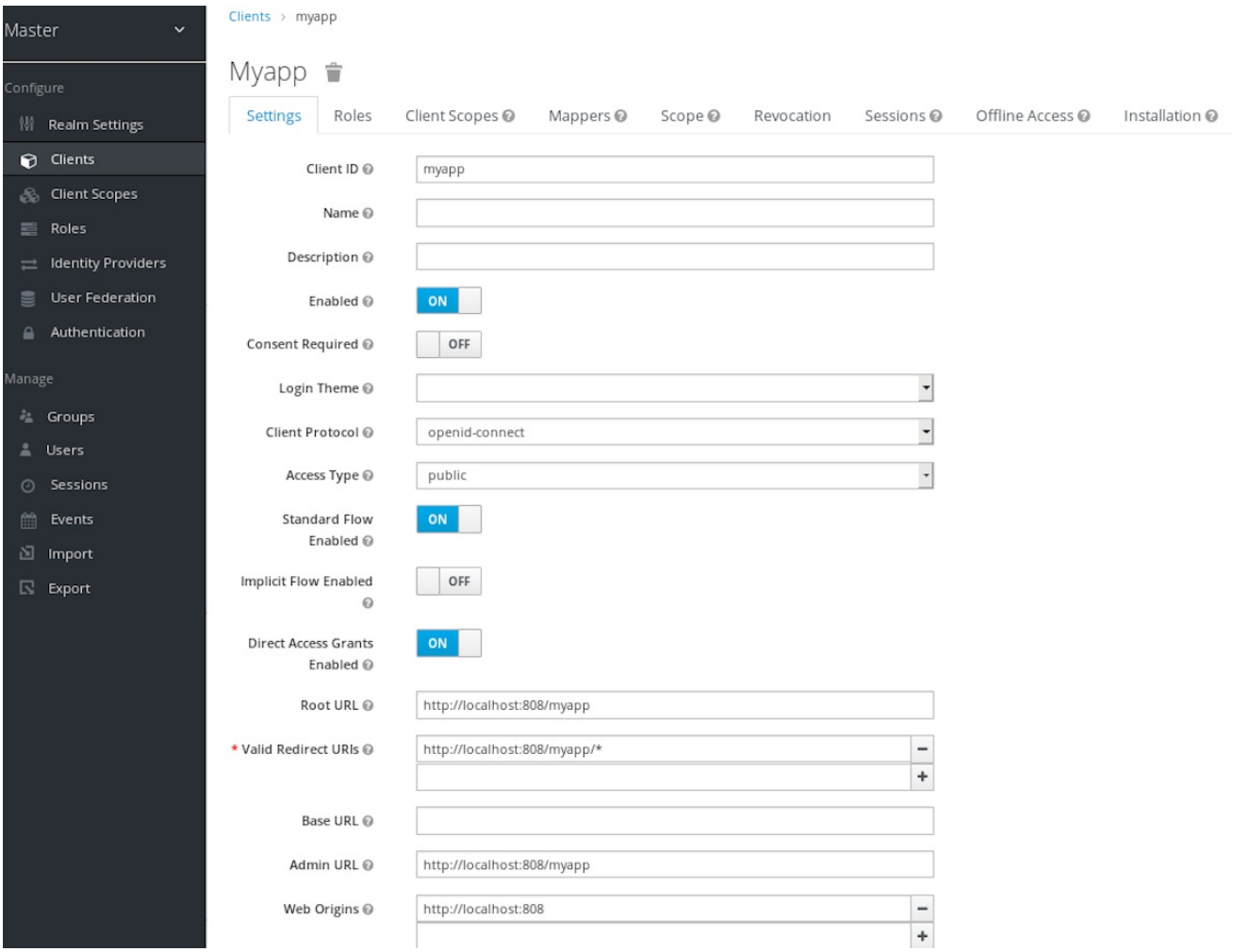
4. 在客户端 协议 下拉菜单中选择 **openid-connect**。

5. 在 **Root URL** 字段中输入应用程序的基本 URL。

6. 点 **Save**。

此操作会创建客户端，并将您带到 **Settings** 选项卡。

客户端设置



其他资源

- 有关 OIDC 协议的更多信息，请参阅 OpenID Connect。

12.1.2. 基本设置

当您创建 OIDC 客户端时，您可以在 Settings 标签页中看到以下字段。

客户端 ID

OIDC 请求和 Red Hat Single Sign-On 数据库中使用的 alpha-numeric ID 字符串来识别客户端。

Name

Red Hat Single Sign-On UI 屏幕中的客户端名称。要本地化名称，请设置替换字符串值。例如，字符串值，如 ${myapp}。如需更多信息，请参阅 服务器开发人员指南。

描述

客户端的描述。此设置也可以本地化。

Enabled

关闭后，客户端无法请求身份验证。

需要同意

在打开后，用户会看到一个授权页面，他们可用于授予该应用程序的访问权限。它还会显示元数据，以便用户知道用户可以访问的确切信息。

访问类型

OIDC 客户端的类型。

机密

对于执行浏览器登录并要求客户端 secret 在发出访问令牌请求时的服务器端客户端。此设置应该用于服务器端应用程序。

公开

对于执行浏览器登录的客户端。由于无法确保 secret 可以与客户端客户端保持安全，因此务必要通过配置正确的重定向 URI 来限制访问。

bearer-only

应用只允许 bearer 令牌请求。开启后，此应用无法参与浏览器登录。

启用标准流

启用后，客户端可以使用 OIDC 授权代码流。

隐式流已启用

启用后，客户端可以使用 OIDC Implicit Flow。

启用直接访问

启用后，客户端可以使用 OIDC Direct Access Grants。

启用 OAuth 2.0 设备授权

如果这样做，则允许客户端使用 OIDC 设备授权授权。

OpenID Connect Client Initiated Backchannel Authentication Enabled

如果出现这种情况，则允许客户端使用 OIDC 客户端初始频道身份验证Grant。

Root URL

如果 Red Hat Single Sign-On 使用任何配置的相对 URL，则会在其前面加上这个值。

有效的 Redirect URI

必填字段.输入 URL 模式，再单击 + 来添加和 - 来删除现有 URL，然后单击保存。确切的（区分大小写）字符串匹配用于比较有效的重定向 URI。

您可以在 URL 模式末尾使用通配符。例如 http://host.com/path/*。为避免安全问题，如果传递的重定向 URI 包含 userinfo 部分或其路径 来管理对 父目录的访问(/../)，则不会执行通配符比较，但标准和安全匹配的字符串匹配。

完整的通配符 * 有效重定向 URI 也可以配置为允许任何 http 或 https 重定向 URI。请不要在生产环境中使用它。

专用重定向 URI 模式通常更安全。如需更多信息，请参阅 未知的重定向 URI。

基本 URL

当 Red Hat Single Sign-On 需要链接到客户端时，会使用这个 URL。

管理员 URL

客户端的回调端点。服务器使用此 URL 进行回调，如推送撤销策略、执行回频道注销和其他管理操作。对于 Red Hat Single Sign-On servlet 适配器，这个 URL 可以是 servlet 应用的根 URL。如需更多信息，请参阅 保护应用程序和服务指南。

徽标 URL

引用客户端应用程序徽标的 URL。

策略 URL

Relying 客户端向 End-User 提供相关 URL 来阅读如何使用配置集数据。

服务 URL 条款

重复客户端为最终用户提供的 URL，可阅读有关相应服务条款的 URL。

Web Origins

输入 URL 模式并点击 + 来添加和 - 删除现有的 URL。点 Save。

此选项处理 跨资源共享(CORS)。如果浏览器 JavaScript 尝试 AJAX HTTP 请求到域不同于 JavaScript 代码来自的服务器，请求必须使用 CORS。服务器必须处理 CORS 请求，否则浏览器不会显示或允许处理请求。这个协议可防止 XSS、CSRF 和其他基于 JavaScript 的攻击。

这里列出的域 URL 嵌入到发送到客户端应用程序的访问令牌中。客户端应用使用此信息来确定是否允许在其上调用 CORS 请求。只有 Red Hat Single Sign-On 客户端适配器只支持此功能。如需更多信息，请参阅保护应用程序和服务指南。

前端频道注销

如果启用了 Front Channel Logout，则应用程序应该可以根据 OpenID Connect Front-Channel Logout 规格通过前端频道注销用户。如果启用，您还应提供 Front-Channel Logout URL。

front-Channel Logout URL

Red Hat Single Sign-On 将通过前端频道向客户端发送注销请求的 URL。

Backchannel Logout URL

当注销请求发送到这个域（通过 end_session_endpoint）时，会导致客户端自己注销的 URL。如果省略，则不会向客户端发送注销请求。

12.1.3. 高级设置

当您点 *Advanced Settings* 时，会显示其他字段。

OAuth 2.0 Mutual TLS Certificate Bound Access Tokens Enabled

注意

要在 Red Hat Single Sign-On 中启用 mutual TLS，请参阅在 WildFly 中启用 mutual SSL。

Mutual TLS 将访问令牌和刷新令牌与客户端证书绑定，后者在 TLS 握手期间交换。这个绑定可防止攻击者使用 stolen 令牌。

这种类型的令牌是持有密钥令牌的拥有者。与 bearer 令牌不同的是，持有者令牌的接收者是否可以验证令牌的发件人是否合法。

如果这个设置位于，则工作流为：

1. 令牌请求在授权代码流或混合流中发送到令牌端点。
2. 红帽单点登录请求客户端证书。
3. Red Hat Single Sign-On 接收客户端证书。

4. 红帽单点登录成功验证客户端证书。

如果验证失败，红帽单点登录会拒绝令牌。

在以下情形中，Red Hat Single Sign-On 将验证客户端发送访问令牌或刷新令牌：

- 令牌刷新请求发送到令牌端点，并带有拥有者(key)刷新令牌。
- UserInfo 请求会发送到 userInfo 端点，其具有拥有者(key)访问令牌。
- 注销请求发送到 Logout 端点，并带有拥有者(key)刷新令牌。

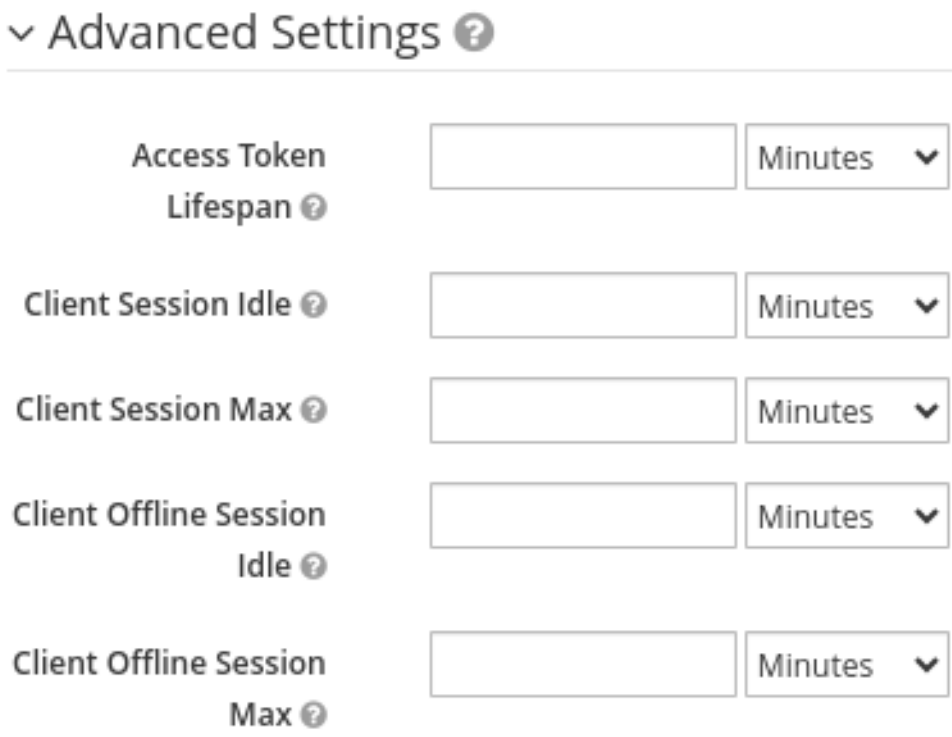
如需了解更多详细信息，请参阅 OAuth 2.0 Mutual TLS 客户端身份验证和证书 Bound Access Tokens 中的 Mutual TLS Client Certificate Bound Access Tokens。

注意

目前，Red Hat Single Sign-On 客户端适配器不支持拥有者密钥令牌验证。红帽单点登录适配器将访问和刷新令牌视为 bearer 令牌。

OIDC 的高级配置

OpenID Connect 的 Advanced Settings 允许您在客户端级别上配置 会话和令牌超时的覆盖。



Configuration	描述
访问令牌 Lifespan	该值会覆盖名称相同的 realm 选项。
客户端会话识别	该值会覆盖名称相同的 realm 选项。该值应小于全局 **SSO 会话空闲**。
客户端会话最大	该值会覆盖名称相同的 realm 选项。该值应小于全局 **SSO Session Max**。
客户端离线会话 Idle	此设置允许您为客户端配置较短的离线会话闲置超时。超时是 Red Hat Single Sign-On 撤销其离线令牌前会话闲置的时间长度。如果没有设置，则使用 realm Offline Session Idle。
客户端离线会话最大	此设置允许您为客户端配置较短的离线会话最大生命周期。生命周期是红帽单点登录撤销对应的离线令牌前的最长时间。这个选项需要在域中全局启用 Session Max Limited，默认为 Offline Session Max。

代码交换代码挑战方法验证密钥

如果攻击者破坏合法客户端的授权代码，则代码交换的概念验证密钥(PKCE)可防止攻击者获得应用到代码的令牌。

管理员可以选择以下选项之一：

（空白）

除非客户端向红帽单点登录授权端点发送适当的 PKCE 参数，否则 Red Hat Single Sign-Ons 不适用于 PKCE。

S256

Red Hat Single Sign-On 适用于客户端 PKCE，其代码质询方法是 S256。

plain

Red Hat Single Sign-On 适用于客户端 PKCE，其代码质询方法是纯的。

如需了解更多详细信息，请参阅 OAuth 公共客户端代码交换的 RFC 7636 证明密钥。

已签名和加密 ID 令牌支持

红帽单点登录可以根据 Json Web 加密(JWE)规范加密 ID 令牌。管理员决定是否为每个客户端配置了 ID 令牌。

用于加密 ID 令牌的密钥是内容加密密钥(CEK)。Red Hat Single Sign-On 和客户端必须协商使用 CEK 以及它的交付方式。用于确定 CEK 的方法是密钥管理模式。Red Hat Single Sign-On 支持的密钥管理模式是密钥加密。

在密钥加密中：

1. 客户端生成非对称加密密钥对。
2. 公钥用于加密 CEK。
3. Red Hat Single Sign-On 会为每个 ID 令牌生成 CEK

4. Red Hat Single Sign-On 使用这个生成的 CEK 加密 ID 令牌

5. Red Hat Single Sign-On 使用客户端的公钥加密 CEK。

6. 客户端使用其私钥解密这个加密的 CEK

7. 客户端使用解密的 CEK 来解密 ID 令牌。

除客户端外，没有方可以解密 ID 令牌。

客户端必须将其公钥用于加密 CEK 到 Red Hat Single Sign-On。Red Hat Single Sign-On 支持从客户端提供的 URL 下载公钥。客户端必须根据 Json Web Keys (JWK) 规范提供公钥。

此过程是：

1. 打开客户端的 Keys 选项卡。

2. 将 JWKS URL 切换到 ON。

3. 在 JWKS URL textbox 中输入客户端的公钥 URL。

密钥加密算法在 Json Web Algorithm (JWA) 规范中定义。Red Hat Single Sign-On 支持：

- RSAES-PKCS1-v1_5(RSA1_5)

- RSAES OAEP 使用默认参数(RSA-OAEP)

- RSAES OAEP 256 使用 SHA-256 和 MFG1 (RSA-OAEP-256)

选择算法的步骤是：

1. 打开客户端的 Settings 选项卡。
2. Open Fine Grain OpenID Connect 配置.
3. 从 ID Token Encryption Content Encryption Algorithm 下拉菜单中选择算法。

ACR 到身份验证级别(LoA)映射

在客户端的高级设置中，您可以定义哪个 身份验证上下文类参考(ACR) 值映射到哪些 级别的身份验证 (LoA)。此映射也可以在 ACR 到 LoA Mapping 所述的域中指定。最佳实践是在 realm 级别配置此映射，它允许在多个客户端间共享相同的设置。

Default ACR Values 可用于指定在从此客户端发送到 Red Hat Single Sign-On 时的默认值，而无需 acr_values 参数，且没有附加了 acr claim 的声明参数。请参阅 不使用 OIDC 动态客户端注册规格。

警告

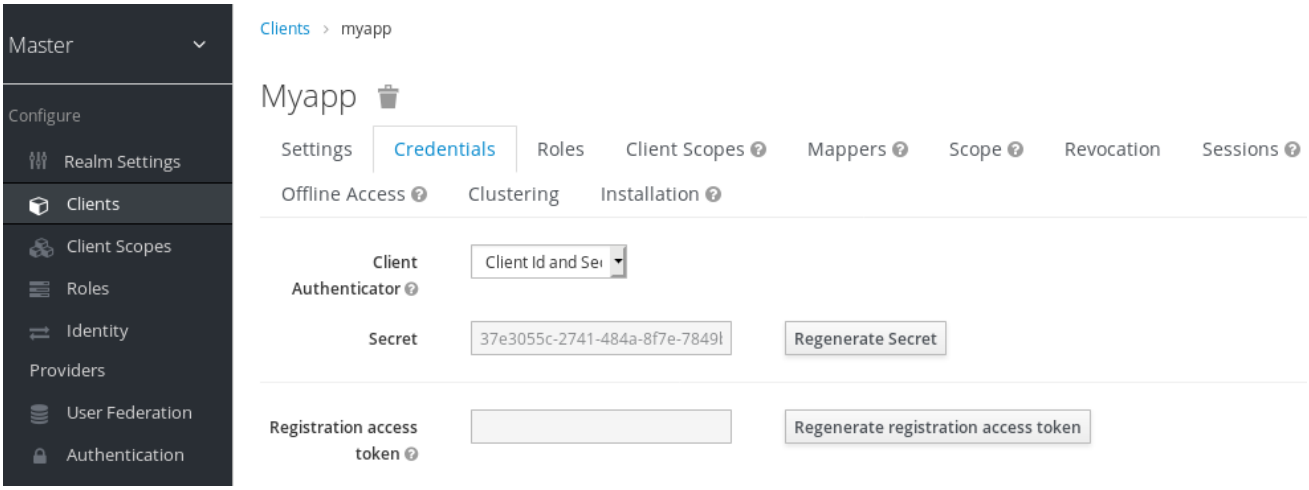
请注意，默认的 ACR 值用作默认级别，但无法可靠使用特定级别强制登录。例如，假设您将 Default ACR 值 配置为级别 2。默认情况下，用户需要使用级别 2 进行验证。但是，当用户将参数显式附加到登录请求时，如 acr_values=1 时，将使用级别 1。因此，如果客户端真正需要级别 2，则鼓励客户端检查 ID Token 内部是否存在 acr 声明，并再次检查它包含请求的级别 2。

详情请查看 Step-up Authentication 和 the offical OIDC specification。

12.1.4. 机密客户端凭证

如果客户端的 访问类型设为 机密，必须在 Credentials 选项卡下配置客户端的凭据。

凭证标签页

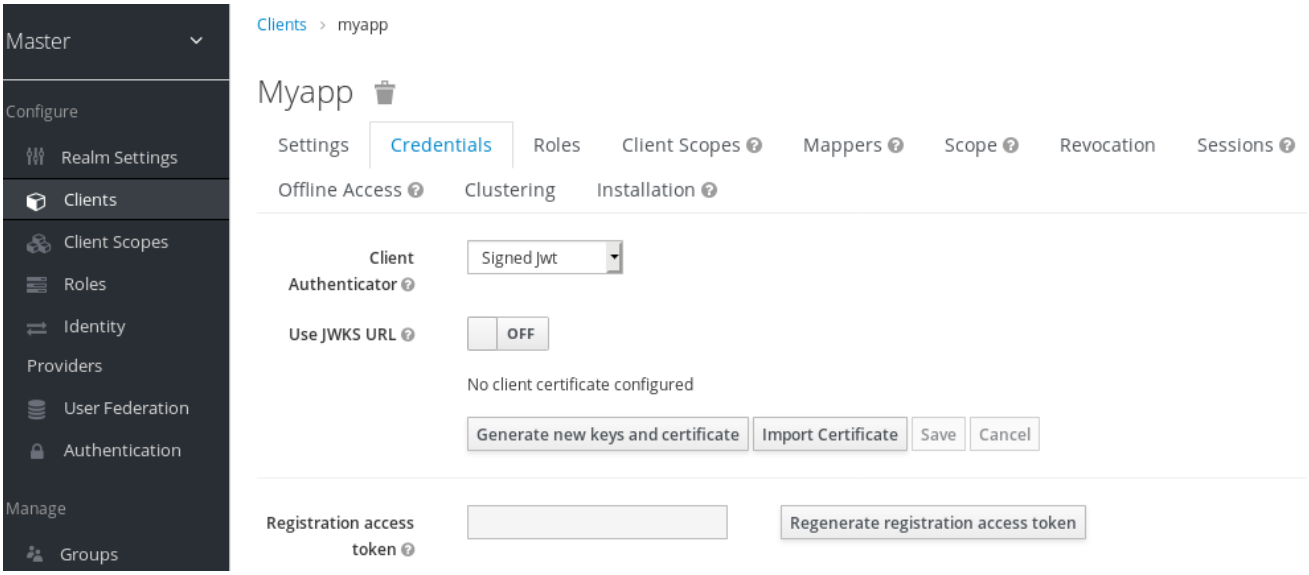


Client Authenticator 下拉列表指定要用于客户端的凭证类型。

客户端 ID 和机密

这个选择是默认设置。为您自动生成 secret，点 Regenerate Secret 会根据需要重新创建 secret。

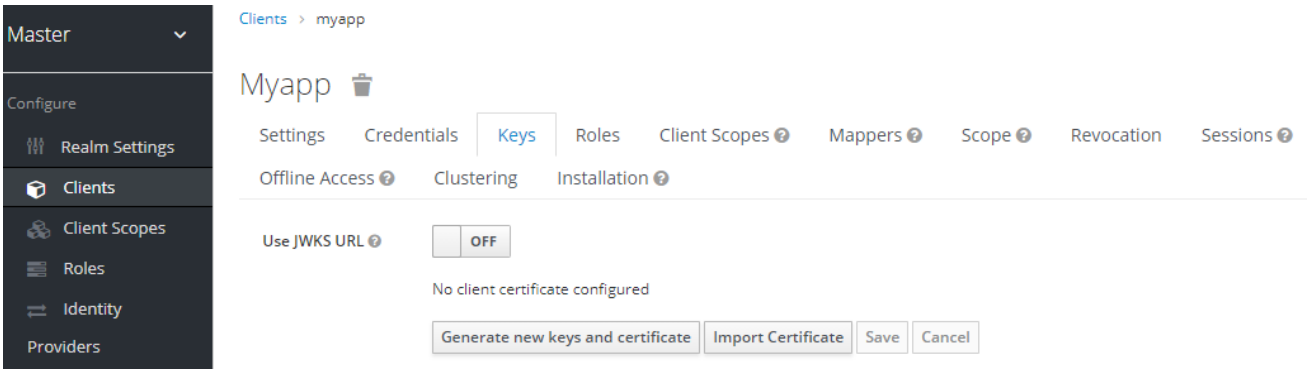
签名的 JWT



签名的 JWT "Signed Json Web 令牌"。

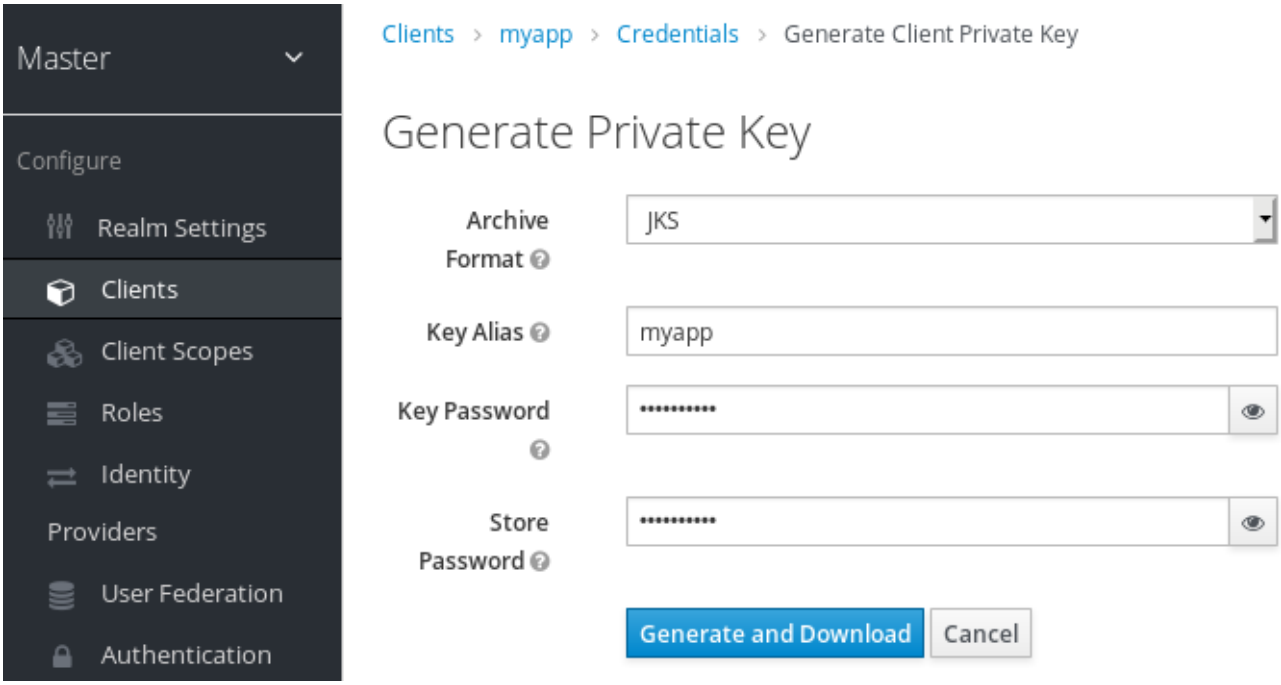
在选择此凭证类型时，您还需要在标签 键中为客户端生成私钥和证书 。私钥将用于为 JWT 签名，而证书则供服务器用于验证签名。

keys 标签页



点击 Generate new key and certificate 按钮启动此过程。

生成密钥



1. 选择您要使用的归档格式。

2. 输入 密钥密码。

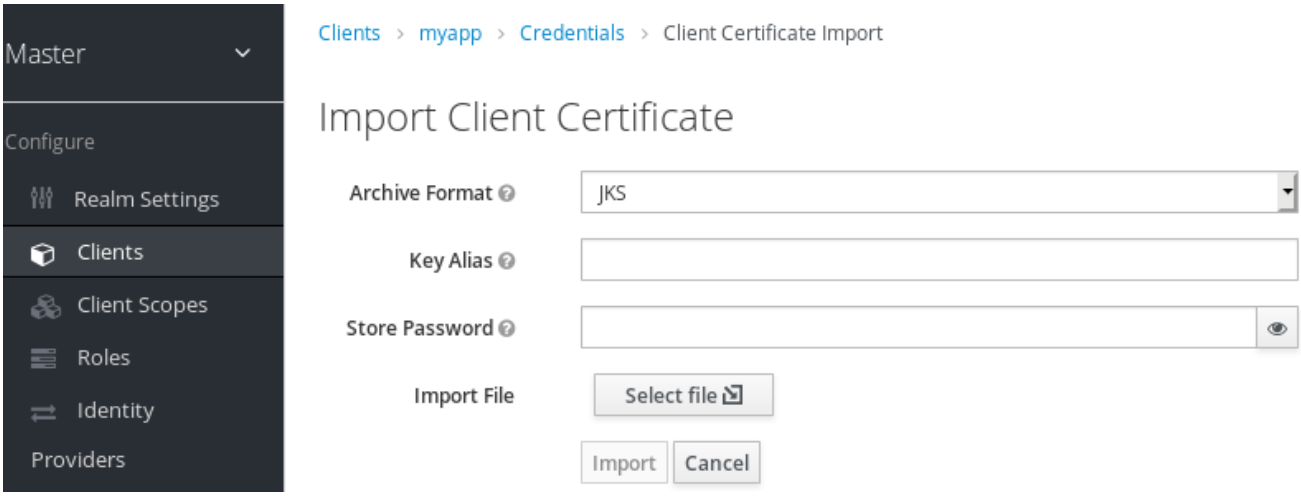
3. 输入 存储密码。

4. 点 Generate and Download。

生成密钥时，Red Hat Single Sign-On 将存储证书并下载您的客户端的私钥和证书。

您还可以使用外部工具生成密钥，然后点 Import Certificate 导入客户端证书。

导入证书



1. 选择证书的归档格式。

2. 输入存储密码。

3. 单击 **Import File**，选择证书文件。

4. 点 **Import**。

如果点击 **Use JWKS URL**，则不需要导入证书。在这种情况下，您可以提供以 JWK 格式发布公钥的 URL。使用此选项时，如果密钥被改变，Red Hat Single Sign-On 会重新导入密钥。

如果您使用由 Red Hat Single Sign-On 适配器保护的客户端，您可以使用以下格式配置 JWKS URL，假设 https://myhost.com/myapp 是客户端应用程序的根 URL：

```
https://myhost.com/myapp/k_jwks
```

如需了解更多详细信息，请参阅 服务器开发人员指南。

警告

Red Hat Single Sign-On 会缓存 OIDC 客户端的公钥。如果您的客户端的私钥被破坏，请更新您的密钥并清除密钥缓存。如需了解更多详细信息，请参阅清除 cache 部分。

使用客户端 Secret 签名 JWT

如果选择此选项，您可以使用由客户端 secret 签名的 JWT，而不使用私钥。

客户端机密将用于由客户端为 JWT 签名。

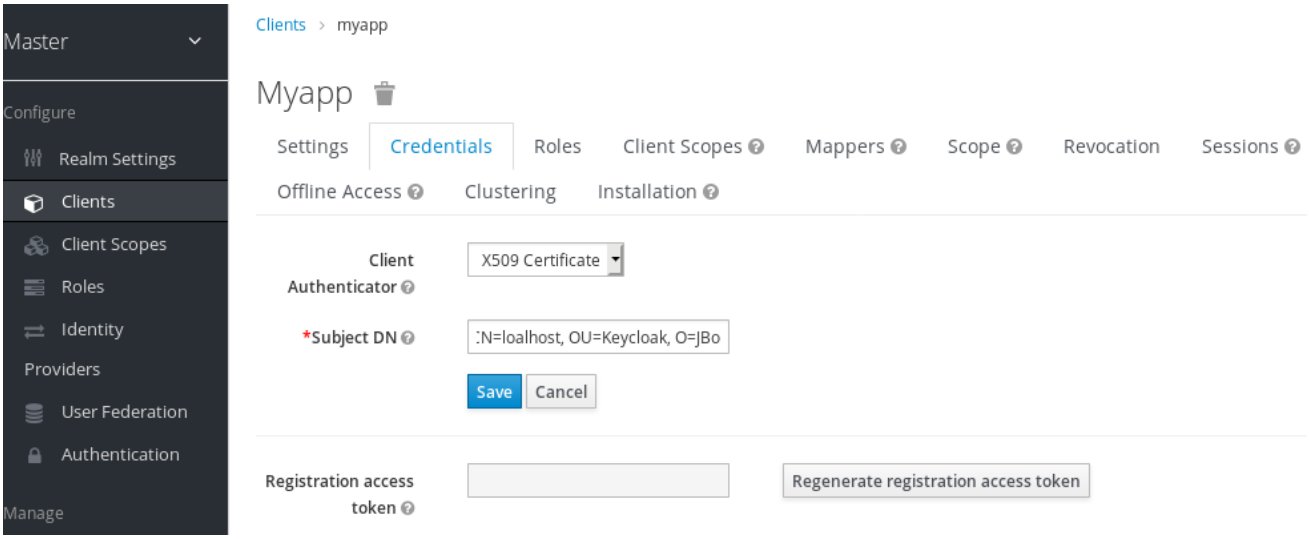
X509 证书

Red Hat Single Sign-On 在 TLS Handshake 中验证客户端是否使用正确的 X509 证书。

注意

这个选项需要在 Red Hat Single Sign-On 中需要 mutual TLS。请参阅 WildFly 中启用 mutual SSL。

X509 证书

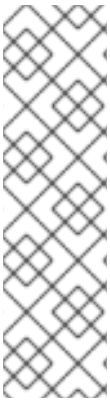


验证器还使用配置的 **regexp** 验证表达式检查证书的 **Subject DN** 字段。对于某些用例，就可以接受所有证书。在这种情况下，您可以使用 **(.*?)(?:$)** 表达式。

Red Hat Single Sign-On 有两种方法以从请求获取客户端 **ID**：

- 查询中的 **client_id** 参数（在 **OAuth 2.0** 规格 的第 **2.2** 节中规定）。
- 提供 **client_id** 作为表单参数。

12.1.5. 客户端机密轮转



注意

客户端 Secret 轮转 是技术预览，不被完全支持。此功能默认为禁用。

要使用 **-Dkeycloak.profile=preview** 或 **-Dkeycloak.profile.feature.client_secret_rotation=enabled** 来启用服务器。如需了解更多详细信息，请参阅 配置文件。

对于具有 机密 访问类型的客户端，红帽单点登录支持通过客户端 策略 轮转客户端 **secret** 的功能。???

客户端 secret 轮转策略提供了更大的安全性，以缓解 secret 泄漏等问题。启用后，Red Hat Single Sign-On 支持每个客户端最多两个并发活跃 secret。策略会根据以下设置管理轮转：

- Secret expiration: [seconds] - 当 secret 被轮转时，这是新 secret 的过期时间。添加到 secret 创建日期 *中的 量（以秒为单位）*。在策略执行时间计算.
- 轮转 secret 过期： [秒] - 当 secret 被轮转时，这个值是旧 secret 的剩余过期时间。这个值应该始终小于 Secret 过期。当值为 0 时，在客户端轮转过程中立即删除旧 secret。添加到 secret 轮转日期 *中的 量（以秒为单位）*。在策略执行时间计算.
- 剩余的更新期间轮转的过期时间： [秒] - 当更新到动态客户端时，应该执行客户端 secret 轮转。在策略执行时间计算.

当出现客户端 secret 轮转时，会生成新的主 secret，而旧客户端主 secret 就会使用新的过期日期成为二级 secret。

12.1.5.1. 客户端 secret 轮转规则

轮转不会自动或通过后台进程发生。要执行轮转，客户端需要更新操作，可以通过 Red Hat Single Sign-On Admin Console 的功能（在客户端的凭证标签页或 Admin REST API 中通过 Regenerate Secret 的功能）。在调用客户端更新操作时，secret 轮转会根据规则进行：

- 当 Secret 过期 值小于当前日期时。
- 在动态客户端注册客户端更新请求时，如果 更新期间 Remaining expiration 时间的值与当前日期与 Secret 到期 间的周期相匹配，客户端 secret 会自动轮转。

另外，管理员 REST API 可以随时强制进行客户端 secret 轮转。

注意

在创建新客户端时，如果客户端 secret 轮转策略处于活跃状态，则会自动应用此行为。

警告

要将 secret 轮转行为应用到现有客户端，请在定义策略后更新该客户端，以便应用此行为。

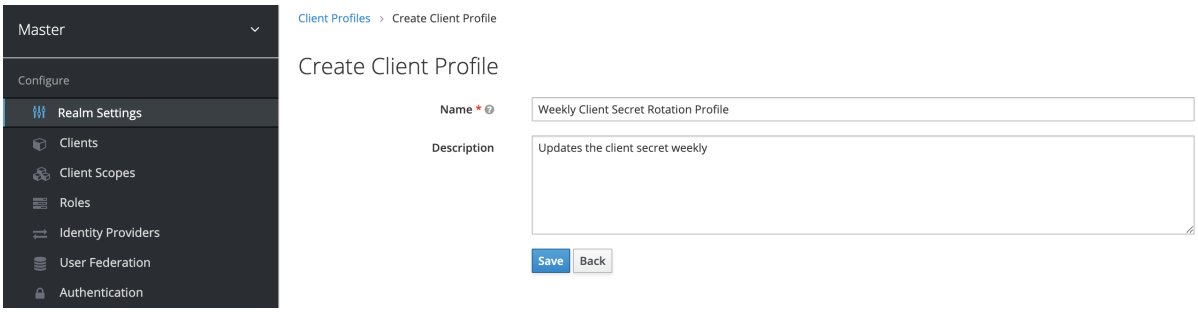
12.1.6. 创建 OIDC 客户端 Secret Rotation 策略

以下是定义 secret 轮转策略的示例：

流程

1. 点左侧菜单中的 **Realm Settings**。
2. 点 **Client Policies** 标签页。
3. 在 **Profiles Page** 上，点 **Create**

创建配置集



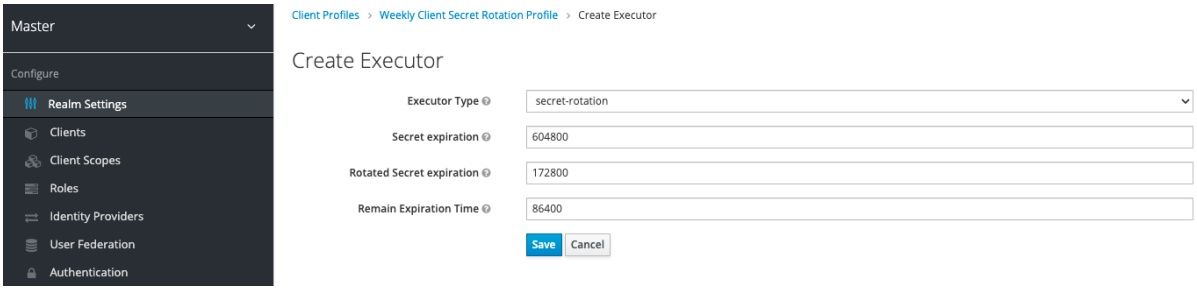
4. 为 **名称** 输入任意名称。
5. 输入描述，以帮助您确定 描述 的配置集用途。

6. **点 Save。**

   此操作会创建配置集，并可让您配置 executors。

7. **点 Create 为这个配置集配置 executor。**

创建配置集 executors



8. 为 **Executor Type** 选择 ***secret-rotation***。

9. 为 **Secret Expiration 输入每个 secret 的最长持续时间时间（以秒为单位）。**

10. 为 **Rotated Secret Expiration 输入每个轮转 secret 的最长持续时间时间（以秒为单位）。**

> **警告**
>
> **请记住，轮转 Secret 过期 值必须始终小于 Secret Expiration。**

11. **输入时间（以秒为单位）后，任何更新操作都会更新 Remain Expiration Time 的客户端。**

12. **点 Save。**

注意

在上例中：

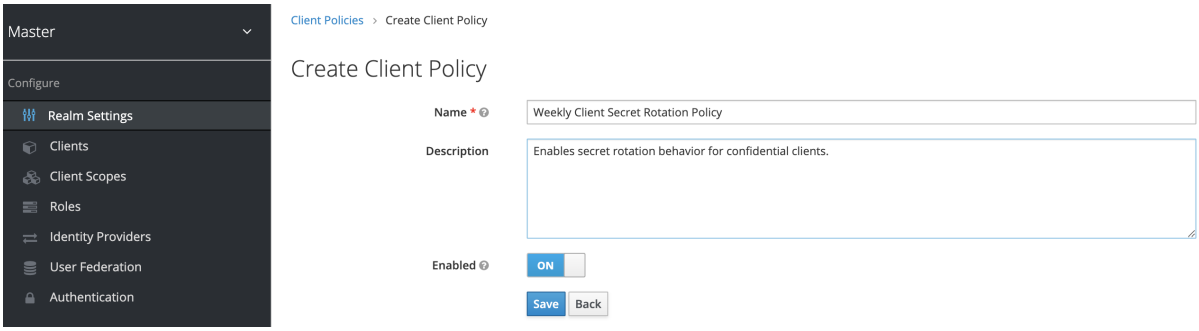
- 每个 secret 只在一周内有效。
- 轮转 secret 在两天后过期。
- 更新动态客户端的窗口将在机密到期前一天启动。

13. 返回到 Client Policies 选项卡。

14. 点 Policies。

15. 点 Create。

创建 Client Secret Rotation 策略



16. 为 名称 输入任意名称。

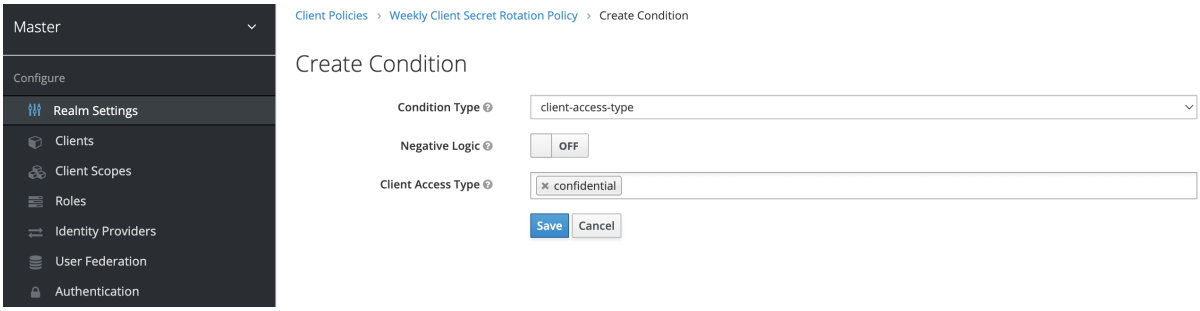
17. 输入描述，帮助您确定 描述 的策略的用途。

18. **点 Save。**

此操作会创建策略，并允许您将策略与配置集关联。它还允许您配置策略执行的条件。

19. **在 Conditions 下，点 Create。**

创建 Client Secret Rotation Policy Condition



20. **要将行为应用到所有机密客户端，请在 Condition Type 字段中选择 *client-access-type***

注意

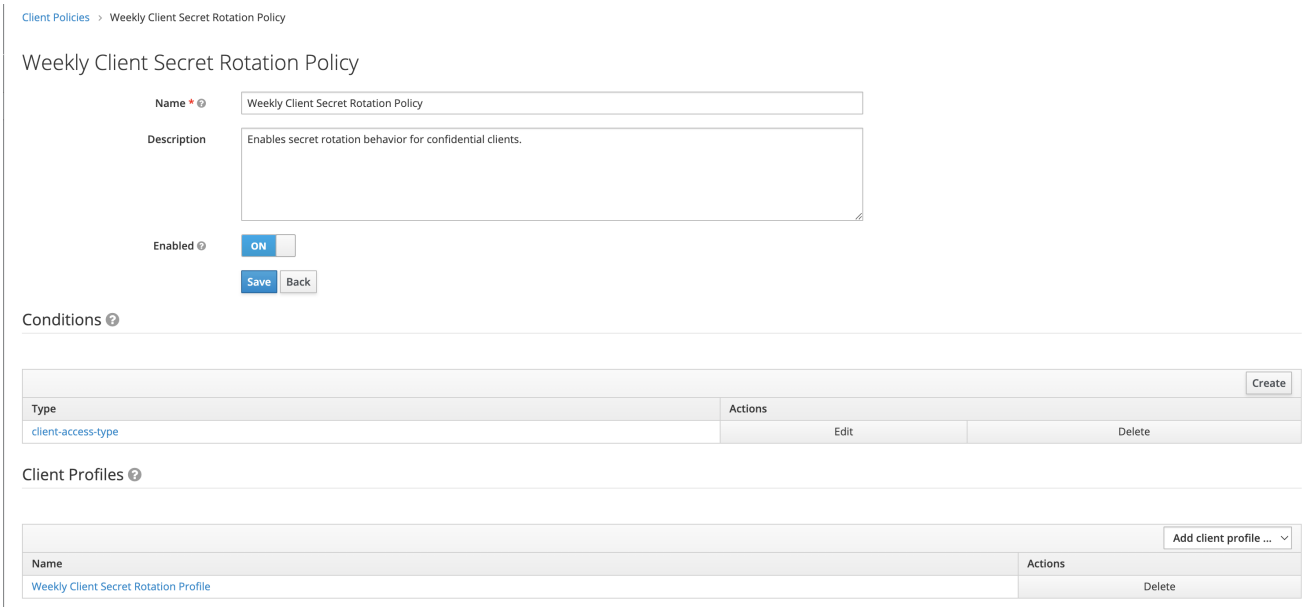
要应用到特定的一组客户端，另一种方法是选择 Condition Type 字段中的 *client-roles* 类型。这样，您可以创建特定的角色，并为每个角色分配自定义轮转配置。

21. **将机密 *添加到* 字段 Client Access Type。**

22. **点 Save。**

23. **返回到在 Add client 配置集选择菜单中的 *Client Profiles* 下的策略设置，选择之前创建的配置集 Weekly Client Secret Rotation Profile。**

Client Secret Rotation 策略



注意

要将 secret 轮转行为应用到现有客户端，请按照以下步骤执行：

使用管理控制台

1. **转至一些客户端。**
2. **转至" *凭证* "选项卡。**
3. **点击 *Re-generate secret*。**

使用客户端 REST 服务可以通过两种方式执行：

- **通过客户端上的更新操作**
- **通过重新生成客户端 secret 端点**

12.1.7. 使用服务帐户

每个 OIDC 客户端都有一个内置 *服务帐户*。使用 *此服务帐户* 获取访问令牌。

流程

1. 点菜单中的 Clients。
2. 选择您的客户端。
3. 点 Settings 选项卡。
4. 将客户端的 Access Type 设置为 机密。
5. 将 服务帐户已启用 切换到 ON。
6. 点 Save。
7. 配置 您的客户端凭据。
8. 点 Scope 选项卡。
9. 验证您有角色或切换 完整范围允许 ON。
10. 点 Service Account Roles 选项卡
11. 配置可用于此客户端的角色。

访问令牌的角色是以下的交集：

- 客户端的角色范围映射和从链接的客户端范围映射中继承的角色范围映射。

- 服务帐户角色.

要调用的 REST URL 是 /auth/realms/{realm-name}/protocol/openid-connect/token。这个 URL 必须以 POST 请求形式调用，并要求您使用请求发布客户端凭证。

默认情况下，客户端凭证由 Authorization: Basic 标头中的客户端的 clientId 和 clientSecret 表示，但您也可以使用签名的 JWT 断言或客户端身份验证的其他自定义机制验证客户端。

您还需要根据 OAuth2 规格将 grant_type 参数设置为 "client_credentials"。

例如，POST 调用以检索服务帐户，如下所示：

```
POST /auth/realms/demo/protocol/openid-connect/token
Authorization: Basic cHJvZHVjdC1zYS1jbGllbnQ6cGFzc3dvcmQ=
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials
```

从 OAuth 2.0 规范中，响应与此 Access Token 响应 类似。

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
    "access_token":"2YotnFZFEjr1zCsicMWpAA",
    "token_type":"bearer",
    "expires_in":60
}
```

默认情况下，仅返回访问令牌。默认情况下，不会返回刷新令牌，在 Red Hat Single Sign-On side 上不创建用户会话。由于缺少刷新令牌，在访问令牌过期时需要重新身份验证。但是，这种情形并不意味着红帽单点登录服务器的额外开销，因为默认不会创建会话。

在这种情况下，注销是不必要的。但是，通过将请求发送到 OAuth2 Revocation Endpoints 部分所述，可以撤销颁发的访问令牌。

其他资源

如需了解更多详细信息，请参阅 客户端凭证授予。

12.1.8. 受众支持

通常，部署红帽单点登录的环境由一组使用 Red Hat Single Sign-On 进行身份验证的 *保密* 或 *公共客户端* 应用程序组成。

服务 (OAuth 2 规范中的*资源服务器*)还可以为来自客户端应用程序的请求提供服务，并为这些应用程序提供资源。这些服务需要向它们发送访问令牌(Bearer 令牌)来验证请求。*此令牌在登录 Red Hat Single Sign-On 后由 frontend 应用获取。*

在服务间信任较低的环境中，您可能会遇到这种情况：

1. *前端客户端应用程序需要针对 Red Hat Single Sign-On 进行身份验证。*
2. *红帽单点登录验证用户。*
3. *红帽单点登录向应用程序签发令牌。*
4. *该应用使用令牌来调用不受信任的服务。*
5. *不受信任的服务会返回对应用程序的响应。但是，它会保留应用程序令牌。*
6. *然后，不受信任的服务使用应用令牌调用可信服务。这会导致安全性损坏，因为不受信任的服务滥用令牌以代表客户端应用程序访问其他服务。*

这种情境在服务间有高信任但并不是信任较低的环境的环境中不太可能出现。在某些环境中，这个工作流可能正确，因为不受信任的服务可能需要从可信服务检索数据，以将数据返回到原始客户端应用。

在服务之间存在高度信任时，一个无限制的受众很有用。否则，受众应限制。您可以限制使用者和同时，允许不受信任的服务从可信服务检索数据。在这种情况下，请确保将不受信任的服务和可信服务添加为令牌的受众。

为防止访问令牌的任何滥用，请限制令牌上的使用者，并配置您的服务来验证令牌上的使用者。流会按如下方式改变：

1. frontend 应用通过 Red Hat Single Sign-On 进行身份验证。
2. 红帽单点登录验证用户。
3. 红帽单点登录向应用程序签发令牌。应用程序知道需要调用不受信任的服务，以便它将 scope=<untrusted service> 放置到发送到 Red Hat Single Sign-On 的身份验证请求中（请参阅 Client Scopes 部分 以了解更多有关 scope 参数的详细信息）。

   给应用程序发布的令牌包含对其受众提供的不受信任的服务的引用("audience": ["<untrusted service>"])，该令牌声明了客户端使用此访问令牌调用不受信任的服务。
4. 不可信服务使用令牌调用可信服务。调用不成功，因为受信任的服务会检查令牌上的使用者，并且发现其受众只针对不受信任的服务。这个行为是正常的，安全没有被破坏。

如果客户端想要稍后调用可信服务，它必须通过使用 范围 =<受信任的服务>重新运行 SSO 登录来获取另一个令牌。然后，返回的令牌会将可信服务作为受众包含在内：

```
"audience": [ "<trusted service>" ]
```

使用这个值调用 < trusted service>。

12.1.8.1. 设置

设置受众检查时：

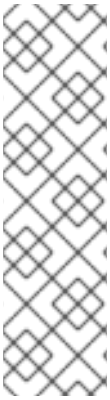
- 通过添加 标志在适配器配置中添加 verify-token-audience，确保服务被配置为检查发送的访问令牌的使用者。详情请参阅 Adapter 配置。
- 确保红帽单点登录发布的令牌包含所有必要的受众。可以使用客户端角色添加使用者，如下一部分所述或硬编码。???请参阅 硬编码的使用者。

12.1.8.2. 自动添加受众

Audience Resolve protocol mapper 在默认的客户端范围 角色 中定义。映射器会检查至少一个可用于当前令牌的客户端角色。然后，每个客户端的客户端 ID 会被添加为受众，如果您的服务（通常为 bearer-only）客户端依赖于客户端角色，这很有用。

例如，对于仅限 bearer 的客户端和机密客户端，您可以使用为机密客户端发布的访问令牌来调用仅 bearer 的客户端 REST 服务。如果满足，则仅限 bearer 的客户端作为使用者自动添加到为机密客户端发布的访问令牌中。

- bearer-only 客户端本身定义了任何客户端角色。
- 目标用户至少分配了其中一个客户端角色。
- 机密客户端具有所分配角色的角色范围映射。



注意

如果要确保受众没有自动添加，请不要在机密客户端上配置角色范围映射。反之，您可以创建一个专用的客户端范围，其中包含专用客户端范围范围的角色范围映射。

假设客户端范围作为可选客户端范围添加到机密客户端，客户端角色和受众将会在令牌中明确请求，则将添加到令牌中。

注意

前端客户端本身不会自动添加到访问令牌受众，从而允许轻松地区分访问令牌和 ID 令牌，因为访问令牌将不包含作为受众发布令牌的客户端。

如果您需要客户端本身作为使用者，请参阅 硬编码的 audience 选项。但是，不建议使用与 frontend 和 REST 服务相同的客户端。

12.1.8.3. 硬编码的受众

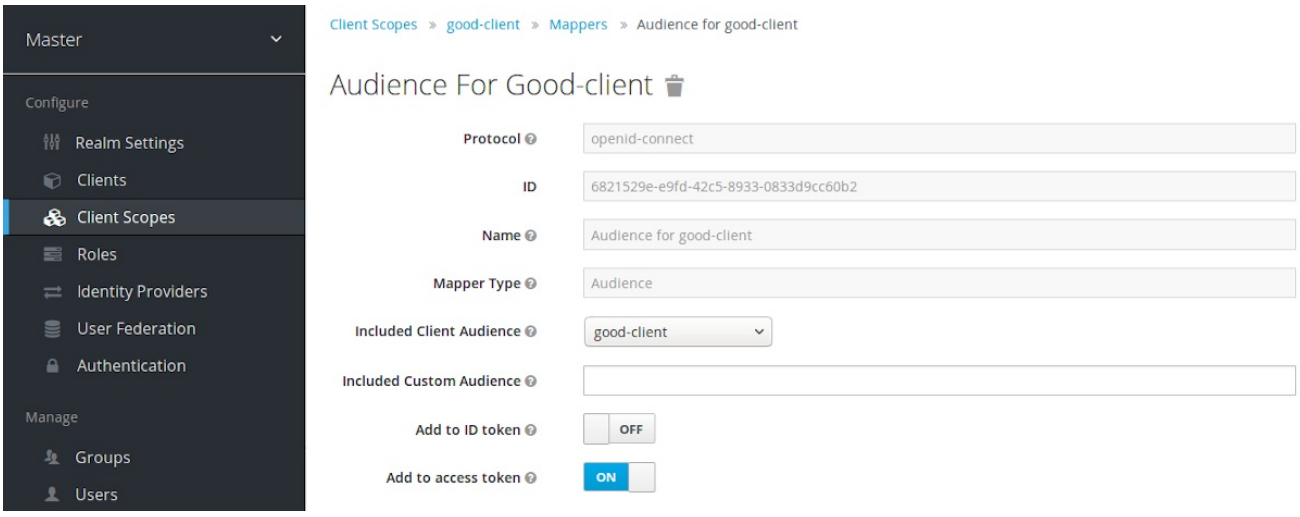
如果您的服务依赖于 realm 角色，或者不依赖于令牌中的角色，那么使用硬编码的受众会很有用。硬编码的使用者是一个协议映射程序，它将将指定服务客户端的客户端 ID 添加为令牌的使用者。如果要使

用与客户端 ID 不同的受众，您可以使用任何自定义值，如 URL。

您可以将协议映射程序直接添加到 frontend 客户端。如果直接添加协议映射程序，那么还将始终添加听众。

如需更多对 protocol mapper 的控制，您可以在专用客户端范围内创建协议映射程序，该范围将调用，如 good服务。

受众协议映射程序



- 在 良好 服务的 Installation 选项卡中，您可以生成适配器配置，您可以确认 verify-token-audience 选项将设置为 true。这在您使用此配置时，会强制适配器验证受众。

- 您需要确保机密客户端能够在其令牌中以使用者请求 良好的服务。

  在机密客户端上：

  1. 点 Client Scopes 选项卡。

  2. 将 good服务 分配为可选（或默认）客户端范围。

     如需了解更多详细信息，请参阅客户端范围链接部分。

- 您可以选择 评估客户端范围 并生成示例访问令牌。如果将 good-service 包含在 scope 参数中，如果将 good服务作为可选客户端范围，则服务 将添加到生成的访问令牌的受众中。

- 在您的机密客户端应用程序中，确保使用 scope 参数。当您要发布令牌以访问良好服务时，必须包括值 good-service 。

  请参阅：

  - 应用程序使用 servlet 适配器时 的参数转发部分。

  - 如果您的应用程序使用 javascript 适配器，则 JavaScript adapter 部分。

注意

默认情况下，A udience 和 Audience Resolve 协议映射器会将受众仅添加到访问令牌中。ID 令牌通常仅包含单个受众，以及发出令牌的客户端 ID，这是 OpenID Connect 规格的要求。但是，访问令牌不一定具有客户端 ID，而这是发出的令牌，除非使用者映射程序被添加。

12.2. 创建 SAML 客户端

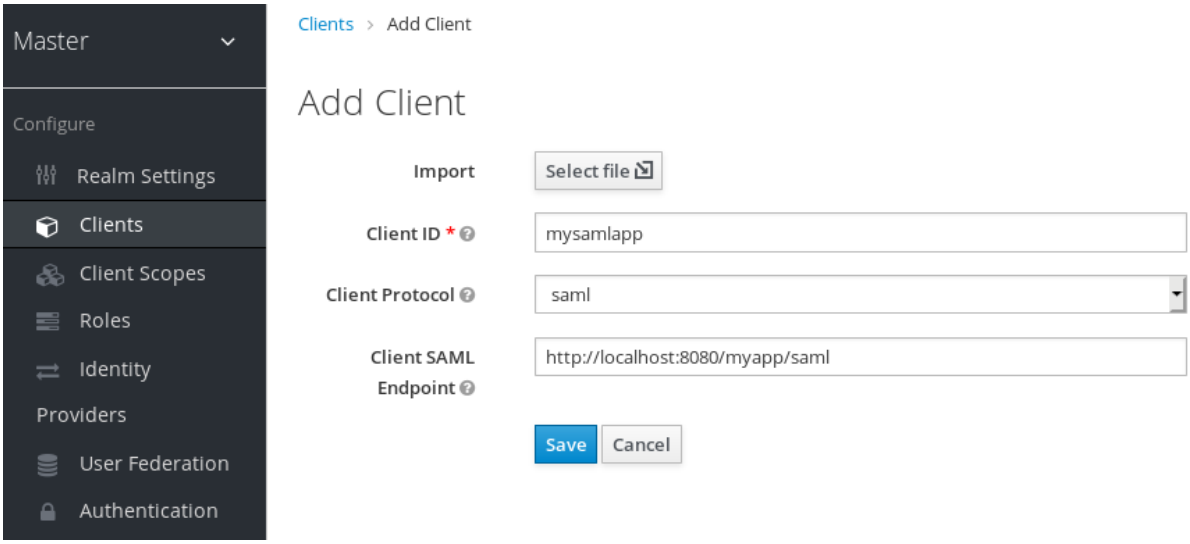
Red Hat Single Sign-On 支持 SAML 2.0 进行注册的应用程序。支持 POST 和 Redirect 绑定。您可以选择需要客户端签名验证。您还可以具有服务器符号和/或加密响应。

流程

1. 点菜单中的 Clients。

2. 点 Create 进入 Add Client 页面。

添加客户端



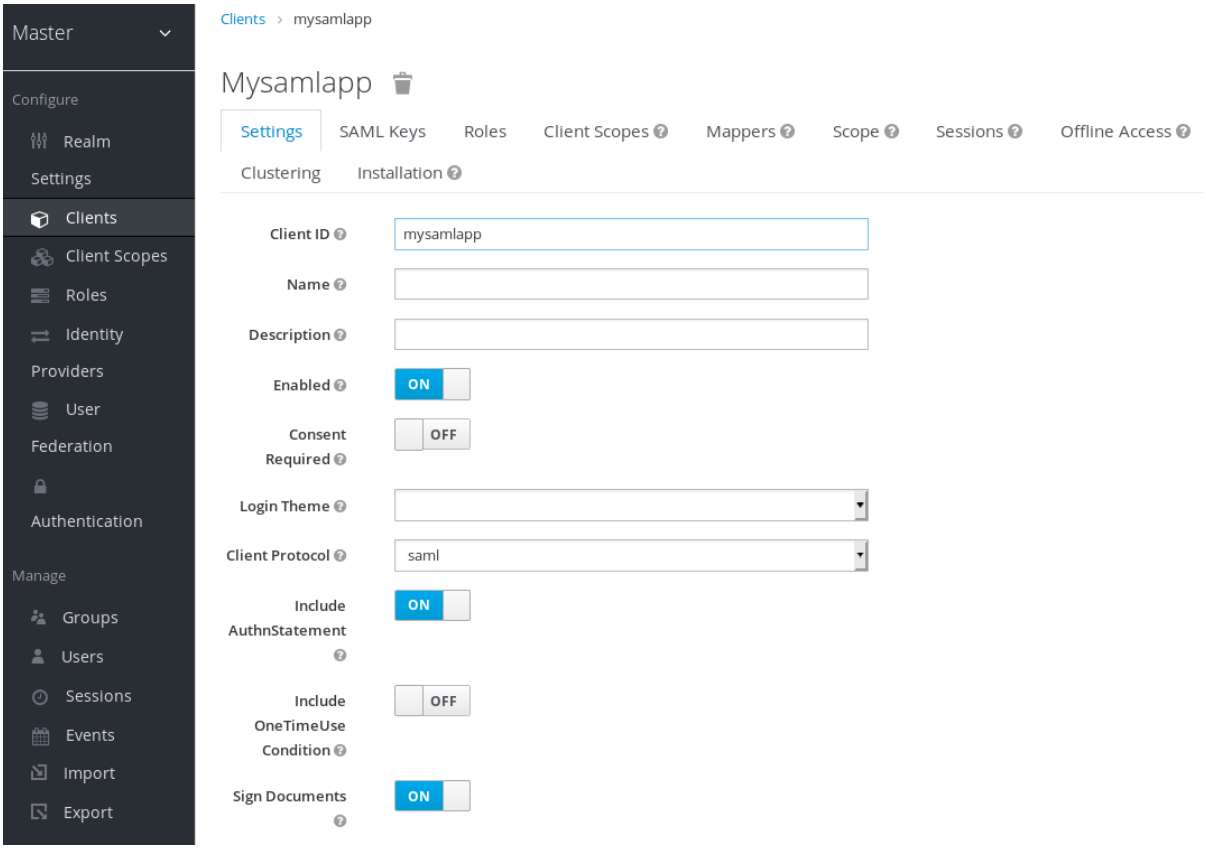
3. *输入客户端的客户端 ID。这通常是 URL，是应用程序发送的 SAML 请求中预期的 签发者值。*

4. *在客户端 协议下拉列表中， 选择 saml。*

5. *输入 Client SAML Endpoint URL。这个 URL 是您希望 Red Hat Single Sign-On 服务器发送 SAML 请求和响应的地方。通常，应用程序有一个处理 SAML 请求的 URL。可在客户端的 Settings 选项卡中设置多个 URL。*

6. *点 Save。此操作会创建客户端，并将您带到 Settings 选项卡。*

客户端设置



以下列表描述了每个设置：

客户端 ID

OIDC 请求和 Red Hat Single Sign-On 数据库中使用的 alpha-numeric ID 字符串来识别客户端。这个值必须与 AuthNRequests 发送的签发值匹配。Red Hat Single Sign-On 从 Authn SAML 请求拉取签发者，并将其与客户端匹配。

Name

Red Hat Single Sign-On UI 屏幕中的客户端名称。要本地化名称，请设置替换字符串值。例如，字符串值，如 ${myapp}。如需更多信息，请参阅 服务器开发人员指南。

描述

客户端的描述。此设置也可以本地化。

Enabled

当设置为 OFF 时，客户端无法请求身份验证。

需要同意

当设置为 ON 时，用户会看到一个授权对该应用的访问权限的同意页面。该页面也显示

客户端可以访问的信息的元数据。如果您曾对 Facebook 进行了一个社交登录，您通常会看到一个类似的页面。红帽单点登录提供相同的功能。

包括 AuthnStatement

SAML 登录响应可能会指定所用的身份验证方法，如密码，以及登录和会话过期的时间戳。包括 Authn Statement 会被默认启用，以便在登录响应中包含 AuthnStatement 元素。将其设置为 OFF 可防止客户端确定最大会话长度，它们可以创建不过期的客户端会话。

签署文档

当设置为 ON 时，Red Hat Single Sign-On 使用域私钥为文档签名。

优化 REDIRECT 签名密钥查找

当设置为 ON 时，SAML 协议信息包括 Red Hat Single Sign-On 原生扩展。这个扩展包含一个带有签名密钥 ID 的提示。SP 使用扩展进行签名验证，而不是尝试使用密钥验证签名。

这个选项适用于在查询参数中传输签名的 REDIRECT 绑定，且此信息不会在签名信息中找到。这只适用于 POST 绑定消息，其中密钥 ID 始终包含在文档签名中。

当 Red Hat Single Sign-On 服务器和适配器提供 IDP 和 SP 时，使用这个选项。只有在将 Sign Documents 设为 ON 时，这个选项才有意义。

Signsertions

断言是签名并嵌入在 SAML XML Auth 响应中。

签名算法

签名 SAML 文档中使用的算法。

SAML 签名密钥名称

使用 POST 绑定发送签名的 SAML 文档包含 KeyName 元素中的签名密钥识别。此操作可由 SAML Signature Key Name 选项控制。此选项控制关键字 的内容。

- KEY_id The KeyName 包含密钥 ID。此选项是默认选项。
- CERT_SUBJECT 的 KeyName 包含与域密钥对应的证书的主题。Microsoft Active Directory Federation 服务预期这个选项。

- NONE The KeyName hint 在 SAML 消息中完全省略。

规范方法

XML 签名的规范方法。

encrypt Assertions

使用 realms 私钥加密 SAML 文档中的断言。AES 算法使用密钥大小为 128 位。

需要客户端签名

如果需要启用客户端签名，则应该签署来自客户端的文档。Red Hat Single Sign-On 将使用在 Keys 选项卡中设置的客户端公钥或证书验证此签名。

强制 POST 绑定

默认情况下，Red Hat Single Sign-On 会使用原始请求的初始 SAML 绑定进行响应。通过启用 Force POST Binding，Red Hat Single Sign-On 会使用 SAML POST 绑定响应，即使原始请求使用了重定向绑定。

前端频道注销

如果启用了 Front Channel Logout，应用程序需要浏览器重定向才能执行注销。例如，应用可能需要对 Cookie 进行重置，而只能通过重定向进行。如果禁用 Front Channel Logout，Red Hat Single Sign-On 会调用后台 SAML 请求退出应用程序。

强制名称 ID 格式

如果请求具有名称 ID 策略，忽略它并使用 名称 ID 格式 下管理控制台中配置的值。

允许 ECP 流

如果为 true，则允许此应用程序使用 SAML ECP 配置集进行身份验证。

名称 ID 格式

主题的 Name ID 格式。如果没有在请求中指定名称 ID 策略，则使用此格式，或者 Force Name ID Format 属性设置为 ON。

Root URL

当 Red Hat Single Sign-On 使用配置的相对 URL 时，这个值会与 URL 的前面。

有效的 Redirect URI

输入 URL 模式，然后单击要添加的 + 符号。点击要删除的 - 符号。点 Save 保存更改。

通配符值只能在 URL 的末尾进行。例如： http://host.com/*$$。当不正确的 SAML 端点没有注册时，会使用此字段，Red Hat Single Sign-On 会拉取 Assertion Consumer URL。

基本 URL

如果 Red Hat Single Sign-On 需要链接到客户端，则会使用这个 URL。

徽标 URL

引用客户端应用程序徽标的 URL。

策略 URL

Relying 客户端向 End-User 提供相关 URL 来阅读如何使用配置集数据。

服务 URL 条款

重复客户端为最终用户提供的 URL，可阅读有关相应服务条款的 URL。

Master SAML 处理 URL

这个 URL 用于所有 SAML 请求，响应会被定向到 SP。它被用作 Assertion Consumer Service URL 和 Single Logout Service URL。

如果登录请求包含 Assertion Consumer Service URL，则这些登录请求将优先使用。这个 URL 必须通过注册的 Valid Redirect URI 模式进行验证。

Assertion Consumer Service POST Binding URL

Assertion Consumer Service 的 POST Binding URL。

Assertion Consumer Service Redirect Binding URL

重定向 Assertion Consumer Service 的绑定 URL。

注销 Service POST Binding URL

Logout Service 的 POST Binding URL。

Logout Service Redirect Binding URL

重定向 Logout 服务的绑定 URL。

logout Service Artifact Binding URL

Logout Service 的构件绑定 URL。当与 Force Artifact Binding 选项结合使用时，在登录和注销流中都强制使用 Artifact 绑定。除非设置了此属性，否则不会用于注销工件 绑定。

工件绑定 URL

将 HTTP 构件消息发送到的 URL。

工件解析服务

客户端 SOAP 端点的 URL，用于将 ArtifactResolve 消息发送到.

12.2.1. IDP Initiated 登录

IDP Initiated Login 是一个功能，允许您在 Red Hat Single Sign-On 服务器上设置端点，供您登录到特定应用程序/客户端。在客户端的 Settings 选项卡中，您需要指定 IDP Initiated SSO URL Name。这是一个简单的字符串，其中没有空格。在此之后，您可以在以下 URL 引用您的客户端：root/auth/realms/{realm}/protocol/saml/clients/{url-name}

IDP 启动登录实施首选通过 REDIRECT 绑定 POST （检查 saml 绑定 以了解更多信息）。因此，最终的绑定和 SP URL 会被选择，其方式如下：

1. *如果定义了特定的 Assertion Consumer Service POST Binding URL （在 Fine Grain SAML Endpoint Configuration 部分） POST 绑定通过该 URL 使用。*
2. *如果指定了常规 Master SAML 处理 URL，则再次通过这个通用 URL 使用 POST 绑定。*
3. *最后的手段是，如果配置了 Assertion Consumer Service Redirect Binding URL （在 Fine Grain SAML Endpoint Configuration中） REDIRECT 绑定与该 URL 一起使用。*

如果您的客户端需要特殊的中继状态，也可以在 IDP Initiated SSO Relay State 字段中的 Settings 选项卡中配置它。或者，浏览器可以在 RelayState 查询参数中指定转发状态，即

root/auth/realms/{realm}/protocol/saml/clients/{url-name}?RelayState=thestate。

在使用 身份代理 时，可以从外部 IDP 为客户端设置 IDP 启动登录。根据上述所述，实际客户端在代理 IDP 中为 IDP 设置 IDP。外部 IDP 必须为应用程序 IDP 发起的登录设置客户端，该登录将指向指向代理的特殊 URL，并在代理 IDP 中代表所选客户端 IDP 的 IDP 发起的登录端点。这意味着，在外部 IDP 的客户端设置中：

- IDP Initiated SSO URL Name 设置为将作为 IDP Initiated Login 初始点发布的名称。
- Fine Grain SAML Endpoint Configuration 部分中的 assertion Consumer Service POST Binding URL 必须设置为以下 URL: broker-root/auth/realms/{broker-realm}/broker/{idp-name}/endpoint/clients/{client-id}，其中：
  - broker-root 是基本代理 URL
  - broker-realm 是声明外部 IDP 的代理的域名称
  - IdP-name 是代理的外部 IDP 的名称
  - client-id 是代理上定义的 SAML 客户端的 IDP Initiated SSO URL Name 属性的值。此客户端将提供给来自外部 IDP 的 IDP 发起的登录。

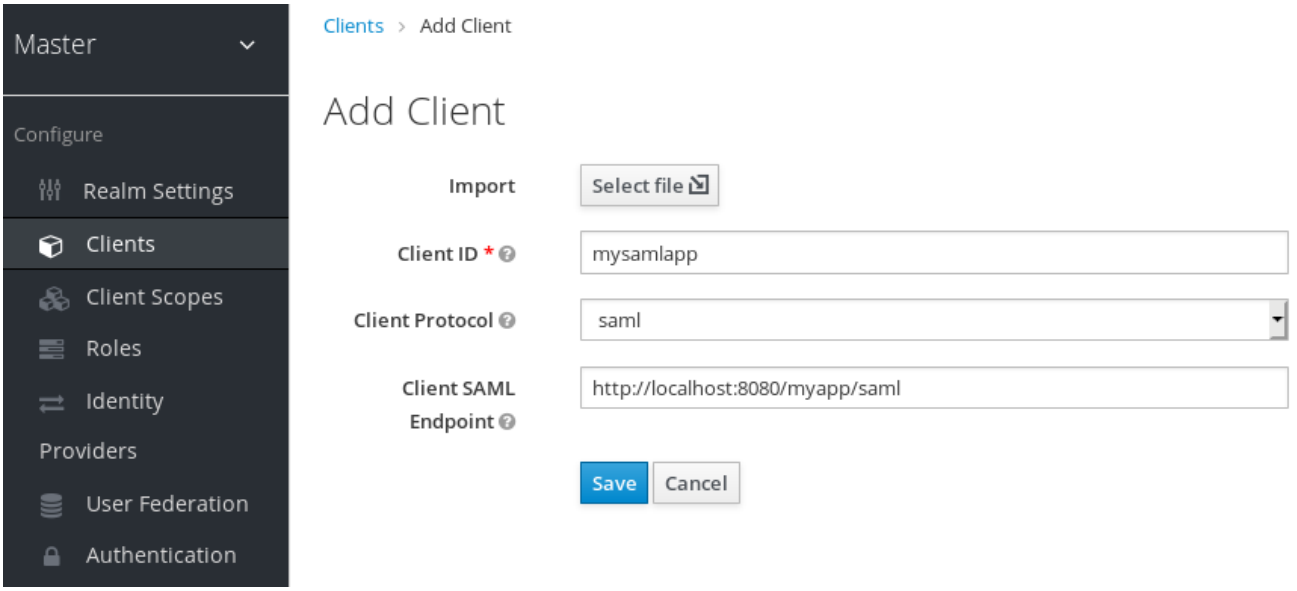
请注意，您可以将基本的客户端设置从代理 IDP 导入到外部 IDP 的客户端设置中 - 仅使用代理 IDP 中的身份提供程序设置中的 SP 描述符，并将 客户端/客户端ID 添加到端点 URL 中。

12.2.2. 使用实体描述符创建客户端

您可以使用标准 SAML 实体描述符 XML 文件导入客户端，而不必手动注册 SAML 2.0 客户端。

Add Client 页面包含一个 Import 选项。

添加客户端



流程

1. *点 Select File。*
2. *加载包含 XML 实体描述符信息的文件。*
3. *检查信息以确保一切设置正确无误。*

有些 SAML 客户端适配器（如 mod-auth-mellon ）需要 IDP 的 XML 实体描述符。您可以通过进入这个 URL 来查找这个描述符：

```
root/auth/realms/{realm}/protocol/saml/descriptor
```

其中 realm 是您的客户端的域。

12.3. 客户端链接

要从一个客户端链接到另一个客户端，Red Hat Single Sign-On 提供了一个重定向端点：/realms/realm_name/clients/{client-id}/redirect。

如果客户端使用 HTTP GET 请求访问此端点，Red Hat Single Sign-On 会以 HTTP 307 (Temporary Redirect)的形式返回提供的客户端和 Realm 形式。因此，客户端只需要知道 Realm 名称和客户端 ID 来

链接到它们。这种间接处理可避免硬编码客户端基本 URL。

例如，为 realm master 和 client-id 帐户 指定：

> http://host:port/auth/realms/master/clients/account/redirect

这个 URL 会临时重定向到： http://host:port/auth/realms/master/account

12.4. OIDC 令牌和 SAML 断言映射

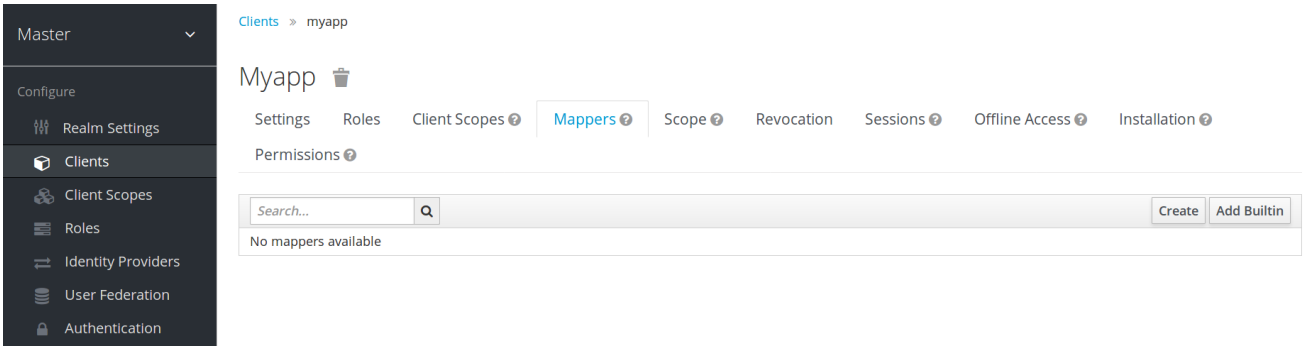
接收 ID 令牌、访问令牌或 SAML 断言的应用程序可能需要不同的角色和用户元数据。

您可以使用 Red Hat Single Sign-On 来：

- Hardcode 角色、声明和自定义属性。
- 将用户元数据拉取到令牌或断言。
- 重命名角色.

您可以在管理控制台中的" 映射程序 "选项卡中执行这些操作。

映射程序标签页

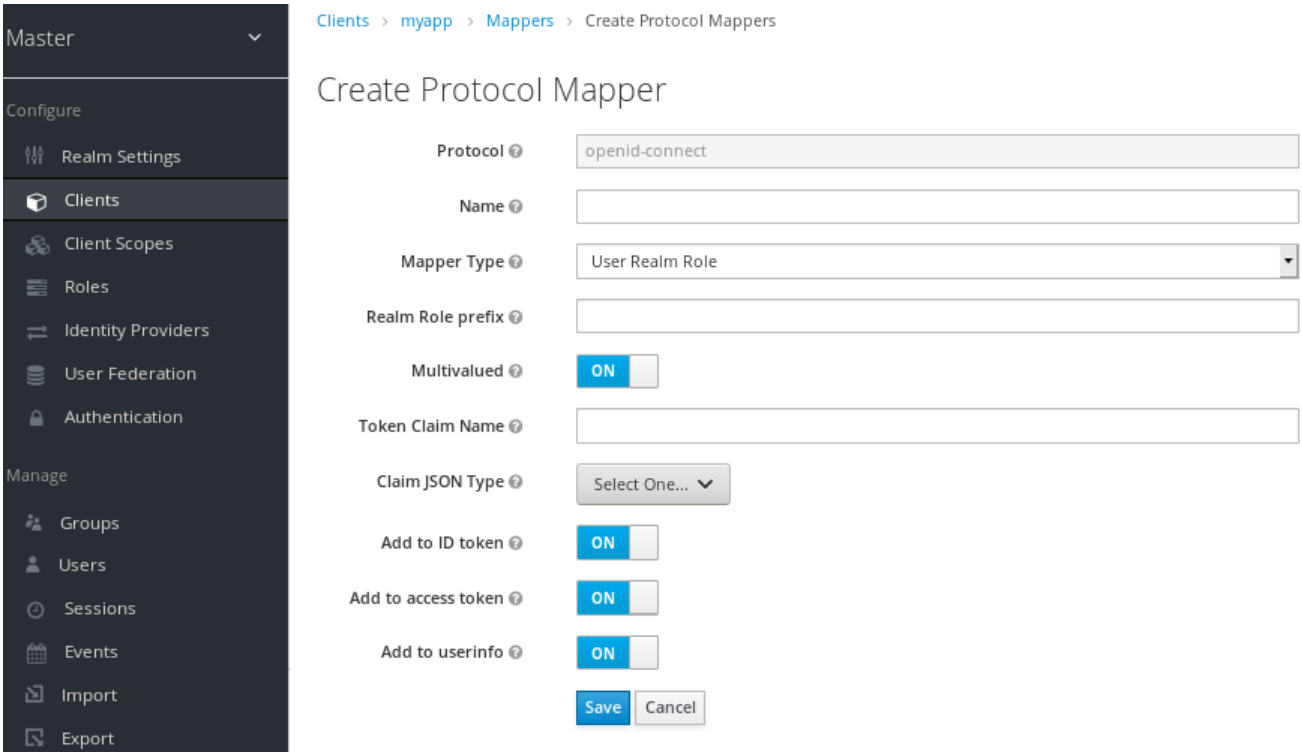


新客户端没有内置映射程序，但它们可以从客户端范围内继承一些映射程序。如需了解更多详细信息，请参阅 客户端范围部分。

协议映射项目（如电子邮件地址）到身份和访问令牌中的特定声明。mapper 的功能应该从其名称中自我说明。您可以通过单击 **Add Builtin** 来添加预先配置的映射程序。

每个映射程序都有一组通用的设置。根据映射程序类型，提供了其他设置。点 mapper 旁边的 **Edit** 来访问配置屏幕，以调整这些设置。

映射器配置



您可以通过将鼠标悬停在工具提示来查看每个选项的详情。

您可以使用大多数 OIDC 映射来控制声明的位置。您可以通过调整 **Add to ID 令牌**和 **Add to access token** 切换来从 **id** 和 **token** 中排除声明。

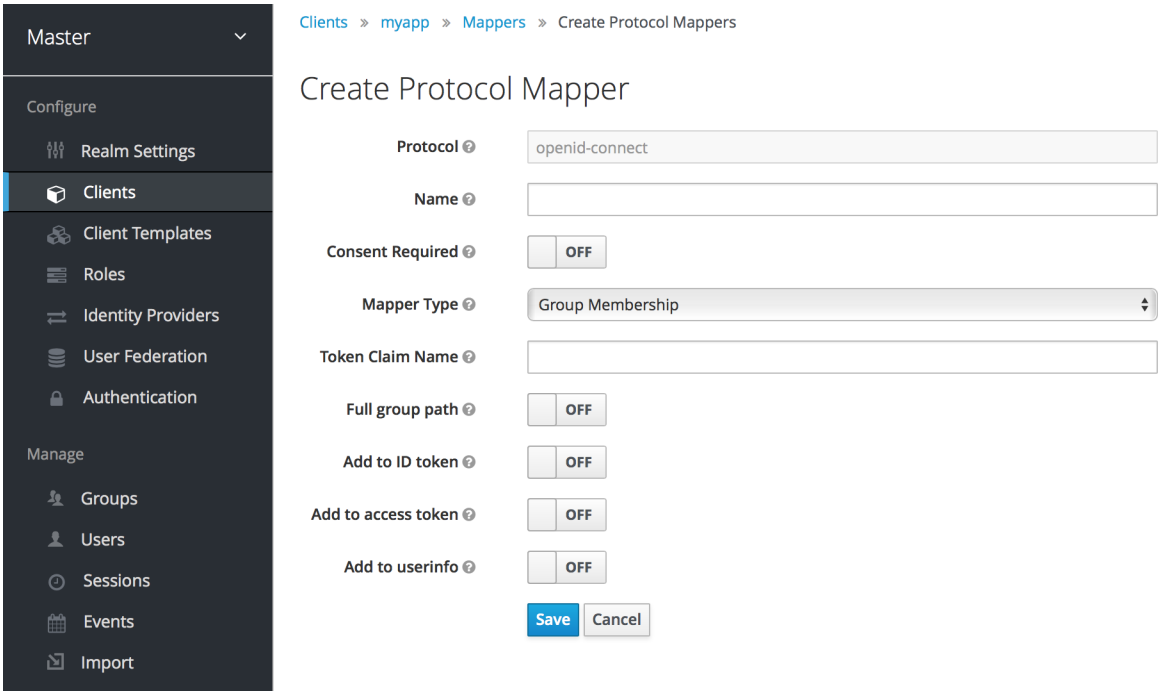
您可以添加映射程序类型，如下所示：

流程

1. *转至" 映射程序 "选项卡。*

2. *点 Create。*

添加映射程序



3. *从列表中选择一个 映射程序类型。*

12.4.1. 优先级顺序

映射程序实施的 优先级顺序.优先级顺序 不是 mapper 的配置属性。它是映射程序实施的特性。

映射程序按映射程序列表按顺序排序。令牌或断言的变化会首先应用最低应用。因此，依赖其他实施的实施会按照必要的顺序处理。

例如，计算将要包含在令牌的角色：

1. 基于这些角色解决受众。

2. 处理使用令牌中已经提供的角色和受众的 JavaScript 脚本。

12.4.2. OIDC 用户会话备注映射程序

用户会话详情使用 mapper 定义，当您使用或启用客户端的功能时会自动包含用户会话详情。点 Add builtin 包含会话详情。

模拟的用户会话提供以下详情：

- IMPERSONATOR_ID ：模拟用户的 ID。
- IMPERSONATOR_USERNAME ：模拟用户的用户名。

服务帐户会话提供以下详情：

- clientID ：服务帐户的客户端 ID。
- clientAddress ：服务帐户验证设备的远程主机 IP。
- clientHost ：服务帐户验证设备的远程主机名。

12.4.3. script mapper

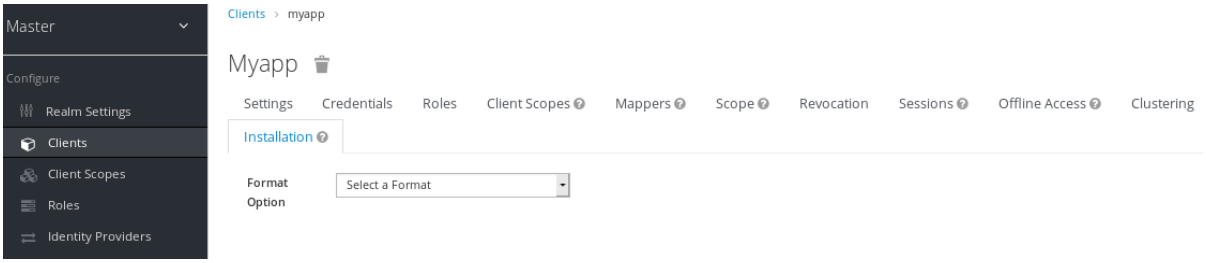
使用 Scriptmapper，通过运行用户定义的 JavaScript 代码将声明映射到令牌。有关将脚本部署到服务器的详情，请参阅 JavaScript Providers。

部署脚本时，您应该能够从可用的映射程序列表中选择部署的脚本。

12.5. 生成客户端适配器配置

Red Hat Single Sign-On 可以生成配置文件，可用于在应用程序部署环境中安装客户端适配器。OIDC 和 SAML 支持很多适配器类型。

1. *转至您要生成配置的客户端 的安装 选项卡。*



2. *选择您要为其 生成的格式化选项。*

支持 OIDC 和 SAML 的所有 Red Hat Single Sign-On 客户端适配器。支持 SAML 的 mod-auth-mellon Apache HTTPD 适配器以及标准 SAML 实体描述符文件。

12.6. 客户端范围

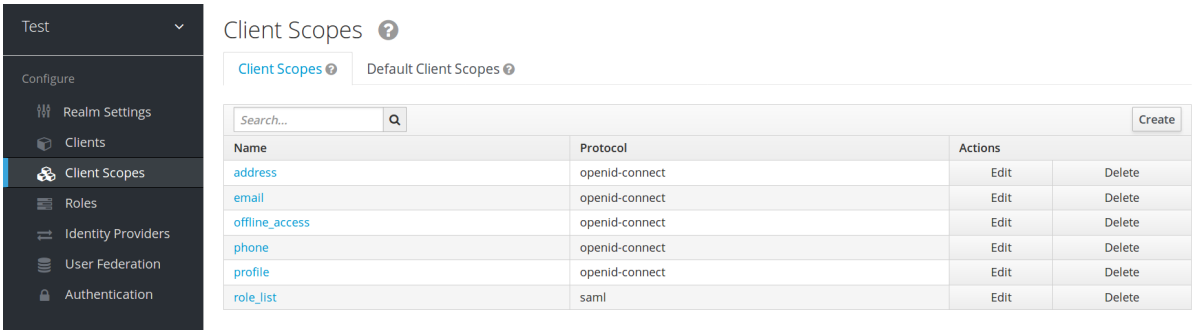
使用红帽单点登录在名为 客户端范围 的实体中定义共享客户端配置。客户端范围为 多个客户端配置 协议映射器 和角色范围映射。

客户端范围也支持 OAuth 2 范围 参数。客户端应用使用此参数来请求访问令牌中的声明或角色，具体取决于应用的要求。

要创建客户端范围，请按照以下步骤执行：

1. *点击菜单中的 Client Scopes。*

客户端范围列表



2. 点 Create。

3. 为您的客户端范围命名。

4. 点 Save。

客户端范围 与常规客户端有类似的选项卡。您可以定义 协议映射器 和角色范围映射。这些映射可以被其他客户端继承，并配置为从这个客户端范围内继承这些映射。

12.6.1. 协议

创建客户端范围时，选择 协议。同一范围中的链接的客户端必须具有相同的协议。

每个 realm 在菜单中具有一组预定义的内置客户端范围。

- SAML 协议： role_list.此范围包含一个协议映射程序，用于 SAML 断言的角色列表。
- OpenID Connect 协议： Several 客户端范围如下：
  - roles

    这个范围没有在 OpenID Connect 规格中定义，它们不会自动添加到访问令牌中的 范围 声明中。此范围具有映射器，用于将用户的角色添加到访问令牌，并为至少具有一个客户端角色的客户端添加受众。这些映射器在 Audience 部分中 详细介绍。

- *web-origins*

  这个范围还没有在 OpenID Connect 规格中定义，且不会添加到声明访问令牌的范围中。此范围用于将允许的 Web 源添加到访问令牌 允许的声明中。

- *microprofile-jwt*

  此范围处理 MicroProfile/JWT Auth 规范 中定义的声明。此范围为 upn 声明定义了用户属性映射程序，以及 组 声明的域角色映射程序。这些映射程序可以更改，以便使用不同的属性来创建 MicroProfile/JWT 特定的声明。

- *offline_access*

  当客户端需要获取离线令牌时，会使用这个范围。有关离线令牌的更多详细信息，请参见 Offline Access 部分和 OpenID Connect 规格。

- *配置集*

- *email*

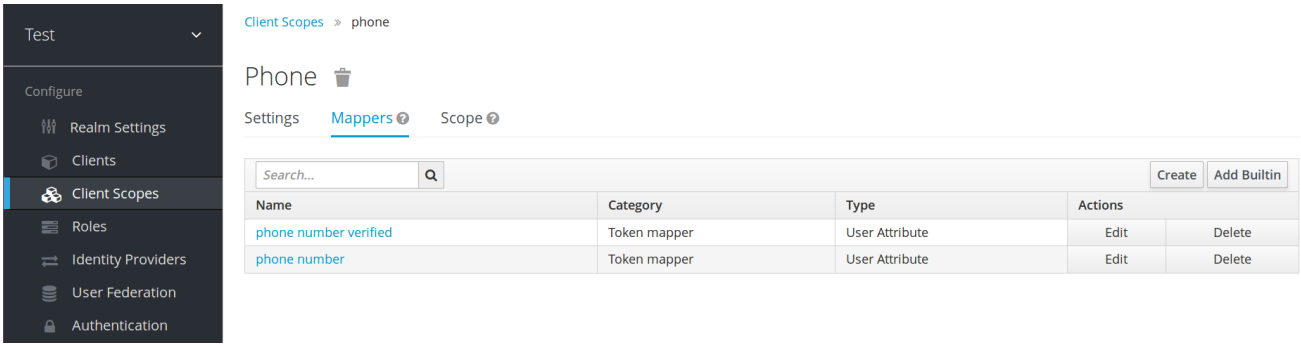
- *address*

- *电话*

客户端范围、电子邮件、地址 和电话 在 OpenID Connect 规范 中定义。这些范围没有定义任何角色范围映射，但它们定义了协议映射。这些映射程序与 OpenID Connect 规范中定义的声明对应。

例如，当您打开 电话 客户端范围并打开 映射程序 标签时，您会看到与范围 电话 规范中定义的声明对应的协议映射程序。

客户端范围映射程序



当 电话 客户端范围连接到客户端时，客户端会自动继承 电话 客户端范围中定义的所有协议映射程序。为这个客户端发布的访问令牌包含用户的电话号码信息，假设该用户有定义的电话号码。

内置客户端范围包含规格中定义的协议映射程序。您可以自由编辑客户端范围，并创建、更新或删除任何协议映射程序或角色范围映射。

12.6.2. 同意相关设置

客户端范围包含与同意屏幕相关的选项。如果客户端上启用了相关的客户端，则 这些选项很有用。

显示 On Consent Screen

如果启用了 Display On Consent Screen，且范围被添加到需要同意的客户端中，则 Consent Screen Text 中指定的文本将显示在同意屏幕上。当用户通过身份验证并且用户从 Red Hat Single Sign-On 重定向到客户端之前，会显示此文本。如果禁用了 Display On Consent Screen，则此客户端范围不会显示在同意屏幕上。

同意屏幕文本

当此客户端范围添加到客户端范围时，在同意过程中显示的文本会默认为客户端范围的名称。此文本的值可以通过使用 ${var-name} 字符串指定替换变量来定制。customised 值在您的主题的属性文件中配置。有关自定义的更多信息，请参阅 服务器开发人员指南。

12.6.3. 使用客户端链接客户端范围

客户端范围和客户端之间的链接在客户端的客户端 范围 选项卡中配置。在客户端范围和客户端之间链接的两种方法如下：

默认客户端范围

此设置适用于 OpenID Connect 和 SAML 客户端。在为客户端发布 OpenID Connect 令牌或 SAML 断言时，会应用默认的客户端范围。客户端将继承客户端范围中定义的协议映射和角色范围映

射。对于 OpenID Connect 协议，始终应用映射程序和角色范围映射，不论 OpenID Connect 授权请求中的范围参数使用的值是什么。

可选的客户端范围

此设置仅适用于 OpenID Connect 客户端。为这个客户端发出令牌时应用可选的客户端范围，但仅在 OpenID Connect 授权请求中的 scope 参数请求时才应用。

12.6.3.1. Example

在本例中，假设客户端有 配置集 和电子邮件 链接为默认客户端范围，以及连接为可选客户端范围的地址。 在向 OpenID Connect 授权端点发送请求时，客户端使用 scope 参数的值。

```
scope=openid phone
```

scope 参数包含字符串，范围值按空格分开。value openid 是用于所有 OpenID Connect 请求的 meta-value。令牌将包含来自默认客户端范围 配置集 和电子邮件 以及 电话 的 mappers 和角色范围映射，以及 scope 参数请求的可选客户端范围。

12.6.4. 评估客户端范围

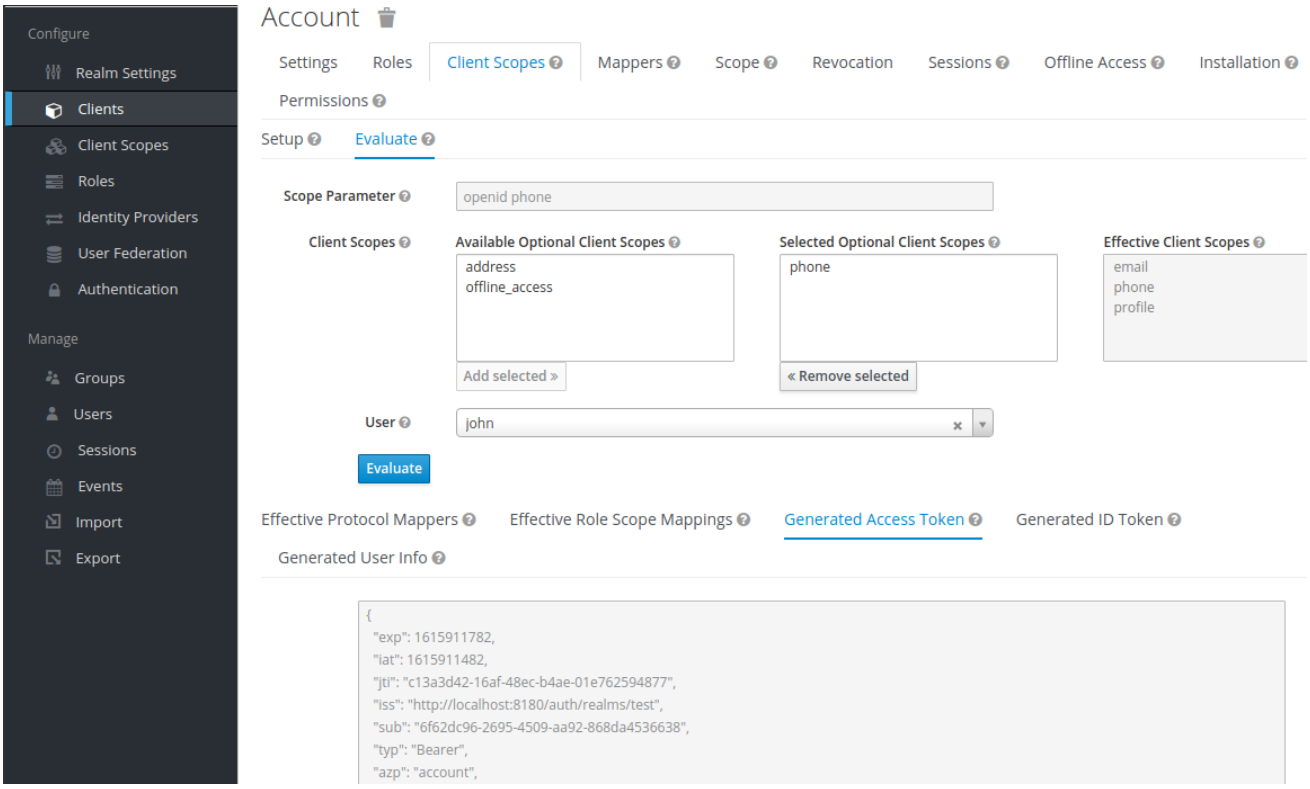
mapper s 选项卡包含 protocol mappers，Scope 选项卡包含为此客户端声明的角色范围映射。它们不包含从客户端范围继承的 mappers 和范围映射。可以查看有效协议映射程序（即客户端本身中定义的协议映射程序以及从链接的客户端范围继承）以及为客户端生成令牌时使用的有效角色范围映射。

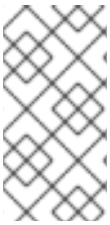
流程

1. 点 客户端的客户端范围 选项卡。
2. 打开"子选项卡 评估 "
3. 选择您要应用的可选客户端范围。

这也会显示 scope 参数的值。此参数需要从应用程序发送到 Red Hat Single Sign-On OpenID Connect 授权端点。

评估客户端范围





注意

要为应用程序发送 范围 参数的自定义值，请参阅 参数转发部分，用于 servlet 适配器 或 javascript adapter 部分，以用于 javascript 适配器。

所有示例是为特定用户生成的，并为特定客户端发布，其值用于指定的 scope 参数。这个示例包括使用的所有声明和角色映射。

12.6.5. 客户端范围权限

向用户发出令牌时，仅当用户被允许使用令牌时，客户端范围才会生效。

当客户端范围没有定义任何角色范围映射时，允许每个用户使用此客户端范围。但是，当客户端范围定义了角色范围映射时，用户必须至少是其中一个角色的成员。用户角色和客户端范围的角色之间必须有一个交集。复合角色工厂用于评估此交集。

如果用户不允许使用客户端范围，则在生成令牌时不会使用协议映射程序或角色范围映射。客户端范围不会出现在令牌中的 范围 值中。

12.6.6. realm 默认客户端范围

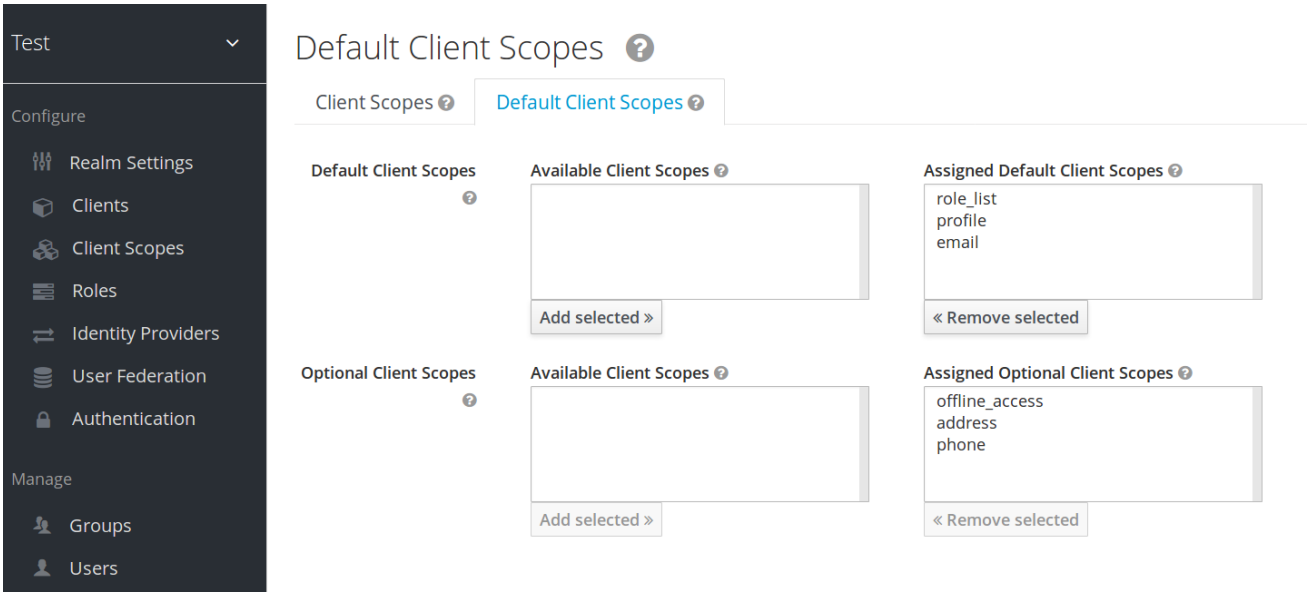
使用 Realm Default Client Scopes 定义自动链接到新创建的客户端的客户端范围集。

流程

1. 点 客户端的客户端范围 选项卡。
2. 点 Default Client Scopes。

从这里，选择您要作为默认客户端范围添加到新创建的 客户端范围 和可选客户端 范围 的客户端范围。

默认客户端范围



创建客户端时，如果需要，可以取消链接默认的客户端范围。这与删除 默认角色类似。

12.6.7. 范围解释

客户端范围

客户端范围是 Red Hat Single Sign-On 中的实体，这些实体在 realm 级别配置，并可链接到客户端。当请求发送到 Red Hat Single Sign-On 授权端点时，客户端范围会引用，其值为 scope 参

数。如需了解更多详细信息，请参阅 客户端范围链接 部分。

角色范围映射

这可以在客户端或客户端范围的 Scope 选项卡中提供。使用角色范围映射 来限制可在访问令牌中使用的角色。如需了解更多详细信息，请参阅角色范围映射部分。

12.7. 客户端策略

为了便于保护客户端应用程序，以统一的方式实现以下点：

- 根据配置客户端可以有什么策略
- 客户端配置验证
- 遵守所需的安全标准和配置文件，如财务级 API (FAPI)

为了实现这些点，引进了 Client Policies 概念。

12.7.1. 使用案例

客户端策略实现以下要点，如下所示：

根据配置客户端可以有什么策略

客户端策略可以在客户端创建/更新期间强制实施客户端的配置设置，但也在与特定客户端相关的 Red Hat Single Sign-On 服务器进行 OpenID Connect 请求期间。Red Hat Single Sign-On 支持类似的事情，也可以通过 保护应用程序和服务指南中的客户端注册 政策支持。但是，客户端注册策略只能涵盖 OIDC Dynamic Client Registration。客户端策略不仅包括客户端注册策略可以做什么，但其他客户端注册和配置方法。当前计划被客户端注册策略替代。

客户端配置验证

Red Hat Single Sign-On 支持验证客户端是否遵循代码交换的概念验证、请求对象签名算法、Holder-of-Key Token 等一些端点，等等。它们可由每个设置项（在管理控制台中、交换机、下拉菜单等）指定。要使客户端应用程序安全，管理员需要以适当方式设置许多设置，因此管理员很难保护客户端应用程序。客户端策略可以对上述客户端配置进行这些验证，也可用于自动配置一些客户端配置开关来满足高级安全要求。以后，单个客户端配置设置可能会被客户端策略直接执行所需的验证替代。

遵守所需的安全标准和配置集，如 FAPI

Global 客户端配置集 是 Red Hat Single Sign-On 中预先配置的客户端配置集。它们被预先配置为符合 FAPI 等标准安全配置集，使得管理员可以轻松地确保其客户端应用程序符合特定的安全配置集。目前，Red Hat Single Sign-On 具有支持 FAPI 1 规格的全局配置集。管理员只需要配置客户端策略，以指定哪些客户端应与 FAPI 兼容。管理员可以配置客户端配置文件和客户端策略，以便可轻松地遵循 Red Hat Single Sign-On 客户端，如 SPA、Native App、Open Banking 等。

12.7.2. 协议

客户端策略概念独立于任何特定的协议。但是，Red Hat Single Sign-On 只支持 OpenID Connect (OIDC)协议。

12.7.3. 架构

客户端策略由四个构建块组成：Condition、Executor、Profile 和 Policy。

12.7.3.1. 状况

条件决定了策略在采用以及何时采用策略的客户端。在客户端请求过程中(OIDC 授权请求、令牌端点请求等)检查了客户端创建/更新的一些条件。该条件检查是否满足指定的条件。例如，一些条件会检查客户端的访问类型是否机密。

该条件不能单独使用。它可以在稍后描述 的策略 中使用。

条件可以与其他可配置提供程序相同。可以配置的内容取决于每个条件的性质。

提供了以下条件：

创建/发布客户端的方法

- 动态客户端注册（匿名或带有初始访问令牌的验证或注册访问令牌）
- admin REST API (Admin 控制台等)

因此，在创建客户端时，当这个客户端由 OIDC Dynamic Client Registration 创建而不初始访问令

牌(Anonymous Dynamic Client Registration)创建此客户端时，可将条件配置为评估为 true。因此，可以使用这个条件来确保通过 OIDC Dynamic Client Registration 注册的所有客户端都兼容 FAPI。

客户端的作者（根据特定角色或组存在）

在 OpenID Connect 动态客户端注册中，客户端的作者是验证获得访问令牌的最终用户，用来生成新客户端，而不是实际通过访问令牌访问注册端点的现有客户端的服务帐户。在由 Admin REST API 注册时，客户端作者是作为红帽单点登录管理员的最终用户。

客户端访问类型（机密、公共、仅 bearer-only）

例如，当客户端发送授权请求时，如果这个客户端保密，则采用策略。

客户端范围

如果客户端有特定的客户端范围（默认或者作为当前请求中使用的可选范围），评估为 true。这可用于确保使用范围为 fapi-example-scope 的 OIDC 授权请求需要符合 FAPI。

客户端角色

适用于具有指定名称的客户端角色

客户端域名、主机或 IP 地址

用于客户端的特定域名。或者对于管理员从特定主机或 IP 地址注册/更新客户端时的情况。

任何客户端

此条件始终评估为 true。它可用于确保特定域中的所有客户端都兼容 FAPI。

12.7.3.2. executor

executor 指定在采用策略的客户端上执行什么操作。executor 执行一个或多个指定的操作。例如，一些 executor 检查授权请求中的参数 redirect_uri 的值是否与授权端点上的预先注册的重定向 URI 完全匹配，并在非授权端点上拒绝此请求。

executor 无法单独单独使用。它可以在稍后 描述的配置 集中使用。

executor 可以与其他可配置供应商相同。可以配置的内容取决于每个 executor 的性质。

executor 对各种事件执行操作。executor 实现可以忽略特定类型的事件（例如，用于检查 OIDC 请求 对象的 executor 则只是 OIDC 授权请求）。事件是：

- 创建客户端（包括通过动态客户端注册创建）
- 更新客户端
- 发送授权请求
- 发送令牌请求
- 发送令牌刷新请求
- 发送令牌撤销请求
- 发送令牌内省请求
- 发送 userinfo 请求
- 使用刷新令牌发送注销请求

在每个事件中，Executor 可以在多个阶段工作。例如，在创建/备份客户端时，Executor 可以通过自动配置特定的客户端设置来修改客户端配置。之后，Executor 在验证阶段验证此配置。

此执行程序有几个目的之一是实现客户端一致性配置集（如 FAPI）的安全要求。要做到这一点，需要以下 executors：

- 对 客户端使用 强制安全客户端身份验证方法
- 使用强制密钥令牌
- 使用 代码交换(PKCE)的 强制密钥

- 使用 签名 JWT 客户端身份验证(private-key-jwt) 的强制签名算法
- 强制 HTTPS 重定向 URI，并确保配置的重定向 URI 不包含通配符
- 强制 OIDC 请求 对象满足高安全性等级
- 强制 OIDC 混合流的 response Type，包括用作 分离签名 的 ID 令牌，如 FAPI 1 规格中所述，这意味着从授权响应返回的 ID 令牌不包含用户配置集数据
- 为防止 CSRF 强制实施更安全的 状态 和非ce 参数处理
- 在客户端注册时强制执行更安全的签名算法
- enforce binding_message 参数用于 CIBA 请求
- 强制 客户端 Secret 轮转

12.7.3.3. profile

一个配置集由多个 executors 组成，它可以实现一个安全配置集，如 FAPI。配置集可以通过 Admin REST API (Admin Console)及其 executors 配置。有三个 全局配置集，它们默认在 Red Hat Single Sign-On 中配置，它与 FAPI Baseline、FAPI Advanced 和 FAPI CIBA 规格兼容。《安全应用程序和服务 指南》 中的 FAPI 部分提供了更多详细信息。

12.7.3.4. 策略

策略由多个条件和配置文件组成。该策略可用于满足此策略的所有条件的客户端。策略会引用多个配置集，这些配置集的所有 executors 都会针对针对此策略所采用的客户端执行其任务。

12.7.4. Configuration

策略、配置文件、条件、执行程序可通过 Admin REST API 配置，这意味着管理控制台。要做到这一点，有一个 Realm → Realm Settings → Client Policies 的标签页，这意味着管理员可以为每个域具有

客户端策略。

Global Client Profiles 会在每个域中自动可用。但是，默认情况下没有配置任何客户端策略。这意味着，如果管理员希望示例其域的客户端符合 FAPI，则始终需要创建任何客户端策略。无法更新全局配置文件，但管理员可以轻松将其用作模板，如果他们想要在全局配置集配置中进行一些小更改，则创建自己的配置集。Admin Console 中提供 JSON Editor，它简化了基于某些全局配置集的新配置集的创建。

12.7.5. 向后兼容性

客户端策略可以替换安全应用程序和服务 指南中的客户端注册策略。但是，客户端注册策略仍存在共存。这意味着，在 Dynamic Client Registration 请求时，要创建/更新客户端，同时应用客户端策略和客户端注册策略。

当前计划适用于要删除的客户端注册策略功能，并且现有的客户端注册策略将自动迁移到新的客户端策略。

12.7.6. 客户端 Secret 轮转示例

请参阅 客户端 secret 轮转的示例配置。

第 13 章 使用 VAULT 获取 SECRET

要从 vault 获取 secret，而不是直接输入它，在适当的字段中输入以下特殊字符串：

```
**${vault.**_key_**}**
```

其中，密钥是 密码库识别的机密的名称。

为防止 secret 在域间泄漏，Red Hat Single Sign-On 会将域名与从 vault 表达式获取 的密钥 合并。此方法意味着 密钥 不会直接映射到密码库中的条目，而是根据用于组合键和域名的算法创建最终条目名称。

您可以在以下字段从 vault 获取 secret：

SMTP 密码

在 realm SMTP 设置中

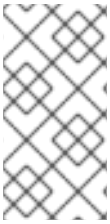
LDAP 绑定凭证

在基于 LDAP 的用户联合的 LDAP 设置中。

OIDC 身份提供程序 secret

在身份提供程序 OpenID Connect Config 中的客户端 Secret 中

要使用密码库，请在 Red Hat Single Sign-On 中注册 vault 提供程序。您可以使用 此处描述的 供应商或实施您的供应商。如需更多信息，请参阅 服务器开发人员指南。



注意

Red Hat Single Sign-On 一次允许每个红帽单点登录实例最多一个活跃的 vault 供应商。在集群中的每个实例中一致地配置 vault 提供程序。

13.1. KUBERNETES/ OPENSHIFT 文件纯文本 VAULT 供应商

红帽单点登录支持 Kubernetes secret 的 vault 实现。您可以将 Kubernetes secret 挂载为数据卷，它们显示为具有无格式文件结构的目录。Red Hat Single Sign-On 代表每个 secret 作为文件，文件名称

为 secret 名称，文件的内容作为 secret 值。

您必须将此目录中的文件命名为以 realm 名称和下划线前缀的 secret 名称。在 secret 名称或域名中的域名中加倍所有下划线。例如，对于名为 sso_realm 的域中的字段，名称 secret-name 的 secret 引用将写为 ${vault.secret-name}，其文件名查找为 sso_realm_secret-name。请注意域名中的下划线。

要使用这类 secret 存储，您必须在 standalone.xml 文件中声明 files-plaintext vault provider，并为包含挂载卷的目录设置其参数。本例演示了 files-plaintext provider with directory，其中 vault 文件被设置为 standalone/configuration/vault，相对于 Red Hat Single Sign-On 基础目录：

```
<spi name="vault">
    <default-provider>files-plaintext</default-provider>
    <provider name="files-plaintext" enabled="true">
        <properties>
            <property name="dir" value="${jboss.home.dir}/standalone/configuration/vault/" />
        </properties>
    </provider>
</spi>
```

以下是使用 CLI 命令的对等配置：

```
/subsystem=keycloak-server/spi=vault/:add
/subsystem=keycloak-server/spi=vault/provider=files-plaintext/:add(enabled=true,properties={dir => "${jboss.home.dir}/standalone/configuration/vault"})
/subsystem=keycloak-server/spi=vault:write-attribute(name=default-provider,value=files-plaintext)
```

13.2. ELYTRON 凭证存储 VAULT 供应商

Red Hat Single Sign-On 还提供对存储在 Elytron 凭证存储中的机密的支持。elytron-cs-keystore vault 提供程序可以从凭据存储密钥存储的实施中检索机密，这也是实施 Elytron 提供的默认实施。

密钥存储来备份此凭据存储。JCEKS 是默认格式，但您可以使用其他格式，如 PKCS12。用户可以使用 WildFly/JBoss EAP 中的 elytron 子系统或 elytron-tool.sh 脚本来创建和管理存储内容。

要使用此提供程序，您必须声明 keycloak-server 子系统中的 elytron-cs-keystore，并设置 Elytron 创建的密钥存储的位置和主机密。供应商配置的最小配置示例：

```
<spi name="vault">
    <default-provider>elytron-cs-keystore</default-provider>
    <provider name="elytron-cs-keystore" enabled="true">
```

```
        <properties>
            <property name="location" value="${jboss.home.dir}/standalone/configuration/vault/credential-store.jceks" />
            <property name="secret" value="secretpw1!"/>
        </properties>
    </provider>
</spi>
```

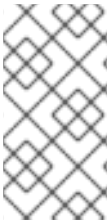
如果底层密钥存储的格式与 JCEKS 不同，则必须使用 keyStoreType 指定以下格式：

```
<spi name="vault">
    <default-provider>elytron-cs-keystore</default-provider>
    <provider name="elytron-cs-keystore" enabled="true">
        <properties>
            <property name="location" value="${jboss.home.dir}/standalone/configuration/vault/credential-store.p12" />
            <property name="secret" value="secretpw1!"/>
            <property name="keyStoreType" value="PKCS12"/>
        </properties>
    </provider>
</spi>
```

对于 secret， elytron-cs-keystore 提供程序使用 elytron-tool.sh 脚本支持明文值和屏蔽值：

```
<spi name="vault">
  ...
            <property name="secret" value="MASK-3u2HNQaMogJJ8VP7J6gRll;12345678;321"/>
  ...
</spi>
```

有关创建和管理电子凭据存储和屏蔽密钥存储 secret 的更多信息，请参阅 Elytron 文档。



注意

红帽单点登录将 elytron-cs-keystore vault 提供程序实施为 WildFly 扩展，并且红帽单点登录服务器仅在 WildFly/JBoss EAP 上运行。

13.3. 密钥解析器

所有内置供应商都支持密钥解析器的配置。键解析器实施算法或策略，用于将域名和从 ${vault.key} 表达式获取的密钥合并到用于从 vault 检索 secret 的最终条目名称中。Red Hat Single Sign-On 使用 keyResolvers 属性来配置提供程序使用的解析器。该值是一个以逗号分隔的解析器名称列表。以下是 files-plaintext 供应商配置示例：

```
<spi name="vault">
    <default-provider>files-plaintext</default-provider>
    <provider name="files-plaintext" enabled="true">
        <properties>
            <property name="dir" value="${jboss.home.dir}/standalone/configuration/vault/" />
            <property name="keyResolvers" value="REALM_UNDERSCORE_KEY, KEY_ONLY"/>
        </properties>
    </provider>
</spi>
```

解析器按照您在配置中声明的顺序相同的顺序运行。对于每个解析器，Red Hat Single Sign-On 使用解析器生成的最后一个条目名，它将 realm 与 vault 密钥合并，以搜索 vault 的机密。如果 Red Hat Single Sign-On 发现了 secret，它将返回 secret。如果没有，Red Hat Single Sign-On 将使用下一个解析器。这个搜索将继续，直到 Red Hat Single Sign-On 找到一个非空 secret 或运行解析器为止。如果 Red Hat Single Sign-On 没有找到 secret，Red Hat Single Sign-On 会返回一个空 secret。

在上例中，Red Hat Single Sign-On 会首先使用 REALM_UNDERSCORE_KEY 解析器。如果 Red Hat Single Sign-On 在使用该解析器的 vault 中找到条目，Red Hat Single Sign-On 会返回该条目。如果没有，Red Hat Single Sign-On 会使用 KEY_ONLY 解析器再次搜索。如果 Red Hat Single Sign-On 使用 KEY_ONLY 解析器发现条目，Red Hat Single Sign-On 会返回该条目。如果 Red Hat Single Sign-On 使用所有解析器，Red Hat Single Sign-On 会返回一个空 secret。

当前可用解析器的列表如下：

Name	描述
KEY_ONLY	Red Hat Single Sign-On 忽略域名并使用 vault 表达式中的密钥。
REALM_UNDERSCORE_KEY	红帽单点登录结合了域和密钥，使用下划线字符。Red Hat Single Sign-On 在 realm 或 key 中又带有另一个下划线的下划线进行转义。例如，如果 realm 名为 **master_realm**，并且密钥是 **smtp_key**，则组合键是 **master__realm_smtp_key**。
REALM_FILESEPARATOR_KEY	红帽单点登录结合了域和密钥，方法是使用平台文件分隔符字符。

如果您还没有为内置提供程序配置解析器，Red Hat Single Sign-On 选择 REALM_UNDERSCORE_KEY。

13.4. 配置示例

以下是配置密码库和凭据存储的示例：该流程涉及两个部分：

- 创建凭据存储和密码库，凭据存储和 vault 密码采用纯文本。
- 更新凭据存储和 vault，使密码使用 elytron-tool.sh 提供的掩码。

在本例中，使用的测试目标是使用 BIND DN 凭证的 LDAP 实例：secret12。目标使用域 ldaptest 中的用户联合来映射。

13.4.1. 在没有掩码的情况下配置凭证存储和库

您可以创建凭证存储和 vault 密码以纯文本形式的密码库。

先决条件

- 正在运行的 LDAP 实例具有 BIND DN 凭证：secret12。
- 使用默认密钥解析器时，别名使用 <realm-name>_< key-value> 格式。在本例中，实例在 realm ldaptest 中运行，而 ldaptest_ ldap_secret 是与该域中的值对应的别名。

注意

解析器用 realm 和密钥名称中的双下划线字符替换下划线字符。例如，对于键 ldaptest_ldap_secret，最终密钥是 ldaptest_ldap_secret。

流程

1. 创建 Elytron 凭据存储。

```
[standalone@localhost:9990 /] /subsystem=elytron/credential-store=test-store:add(create=true, location=/home/test/test-store.p12, credential-reference={clear-text=testpwd1!},implementation-properties={keyStoreType=PKCS12})
```

2. 在凭据存储中添加一个别名。

```
/subsystem=elytron/credential-store=test-store:add-
alias(alias=ldaptest_ldap__secret,secret-value=secret12)
```

请注意，解析器如何使键 ldaptest_ldap__secret 使用双下划线。

3. 列出凭据存储中的别名，以检查 Elytron 生成的密钥存储的内容。

```
keytool -list -keystore /home/test/test-store.p12 -storetype PKCS12 -storepass
testpwd1!
Keystore type: PKCS12
Keystore provider: SUN

Your keystore contains 1 entries

ldaptest_ldap__secret/passwordcredential/clear/, Oct 12, 2020, SecretKeyEntry,
```

4. 在 Red Hat Single Sign-On 中配置 vault SPI。

```
/subsystem=keycloak-server/spi=vault:add(default-provider=elytron-cs-keystore)

/subsystem=keycloak-server/spi=vault/provider=elytron-cs-
keystore:add(enabled=true, properties={location=>/home/test/test-store.p12,
secret=>testpwd1!, keyStoreType=>PKCS12})
```

此时，密码库和凭据存储密码不会被屏蔽。

```
    <spi name="vault">
        <default-provider>elytron-cs-keystore</default-provider>
        <provider name="elytron-cs-keystore" enabled="true">
          <properties>
            <property name="location" value="/home/test/test-store.p12"/>
            <property name="secret" value="testpwd1!"/>
            <property name="keyStoreType" value="PKCS12"/>
          </properties>
        </provider>
      </spi>

     <credential-stores>
        <credential-store name="test-store" location="/home/test/test-store.p12"
create="true">
          <implementation-properties>
            <property name="keyStoreType" value="PKCS12"/>
          </implementation-properties>
          <credential-reference clear-text="testpwd1!"/>
        </credential-store>
     </credential-stores>
```

5. *在 LDAP 供应商中，将 binDN 凭据 替换为 ${vault.ldap_secret}。*

6. *测试您的 LDAP 连接。*

LDAP Vault

13.4.2. 屏蔽凭据存储和 vault 中的密码

现在，您可以更新凭据存储和 vault，使其使用密码使用 elytron-tool.sh 提供的掩码。

1. *使用 salt 和 迭代 参数的值创建屏蔽密码：*

```
$ EAP_HOME/bin/elytron-tool.sh mask --salt SALT --iteration ITERATION_COUNT --secret PASSWORD
```

例如：

```
elytron-tool.sh mask --salt 12345678 --iteration 123 --secret testpwd1!
MASK-3BUbFEyWu0lRAu8.fCqyUk;12345678;123
```

2. *更新 Elytron 凭据存储配置以使用屏蔽的密码。*

```
/subsystem=elytron/credential-store=cs-store:write-attribute(name=credential-reference.clear-text,value="MASK-3BUbFEyWu0lRAu8.fCqyUk;12345678;123")
```

3. *更新 Red Hat Single Sign-On vault 配置以使用屏蔽的密码。*

```
/subsystem=keycloak-server/spi=vault/provider=elytron-cs-keystore:remove()
/subsystem=keycloak-server/spi=vault/provider=elytron-cs-keystore:add(enabled=true, properties={location=>/home/test/test-store.p12, secret=>"MASK-3BUbFEyWu0lRAu8.fCqyUk;12345678;123", keyStoreType=>PKCS12})
```

现在，库和凭证存储被屏蔽：

```
<spi name="vault">
      <default-provider>elytron-cs-keystore</default-provider>
      <provider name="elytron-cs-keystore" enabled="true">
        <properties>
          <property name="location" value="/home/test/test-store.p12"/>
          <property name="secret" value="MASK-
3BUbFEyWu0lRAu8.fCqyUk;12345678;123"/>
          <property name="keyStoreType" value="PKCS12"/>
        </properties>
      </provider>
    </spi>
    ....
    .....
    <credential-stores>
      <credential-store name="test-store" location="/home/test/test-store.p12"
create="true">
        <implementation-properties>
          <property name="keyStoreType" value="PKCS12"/>
        </implementation-properties>
        <credential-reference clear-text="MASK-
3BUbFEyWu0lRAu8.fCqyUk;12345678;123"/>
      </credential-store>
    </credential-stores>
```

4. 现在，您可以使用 ${vault.ldap_secret} 来测试与 LDAP 的连接。

其他资源

有关 Elytron 工具的更多信息，请参阅 使用带有 Elytron 客户端的凭据 存储。

第 14 章 配置审核以跟踪事件

Red Hat Single Sign-On 包括一组审计功能。您可以记录每个登录和管理员操作，并在管理控制台中查看这些操作。Red Hat Single Sign-On 还包括侦听事件的 Listener SPI，它可以触发操作。内置监听程序示例包括日志文件，并在事件发生时发送电子邮件。

14.1. 登录事件

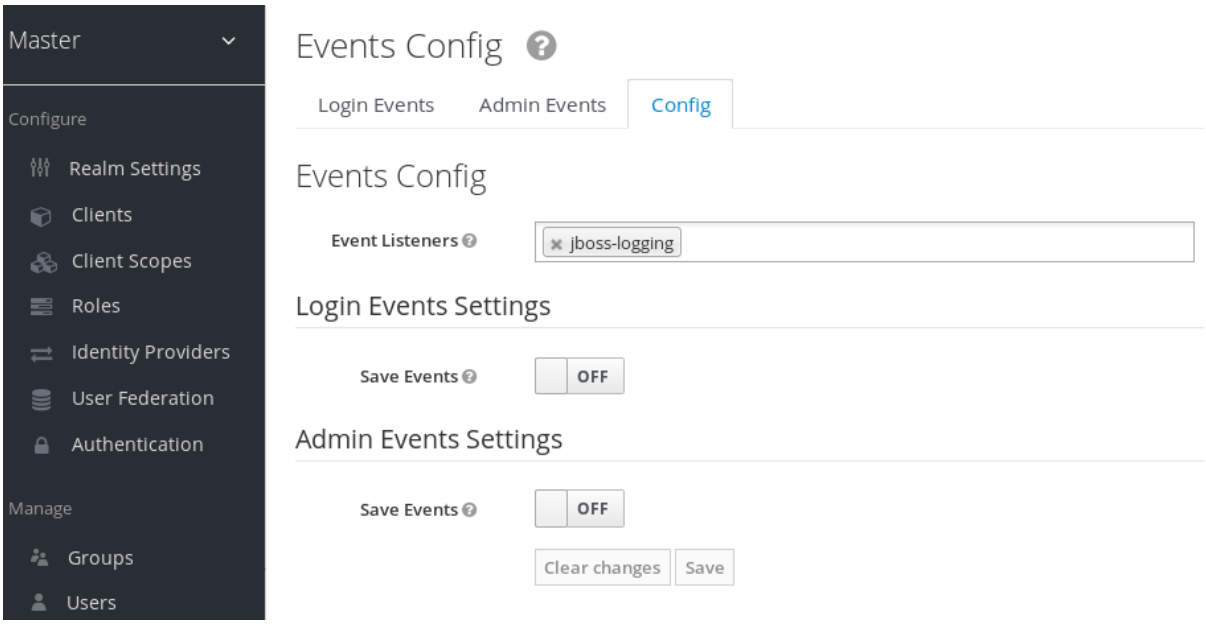
您可以记录并查看影响用户的每个事件。Red Hat Single Sign-On 会触发登录事件，例如成功用户登录、用户输入错误的密码或用户帐户更新等操作。默认情况下，Red Hat Single Sign-On 不会将事件存储在管理控制台中。只有错误事件才会记录到 Admin 控制台和服务器的日志文件。

要开始保存事件，请启用存储。

流程

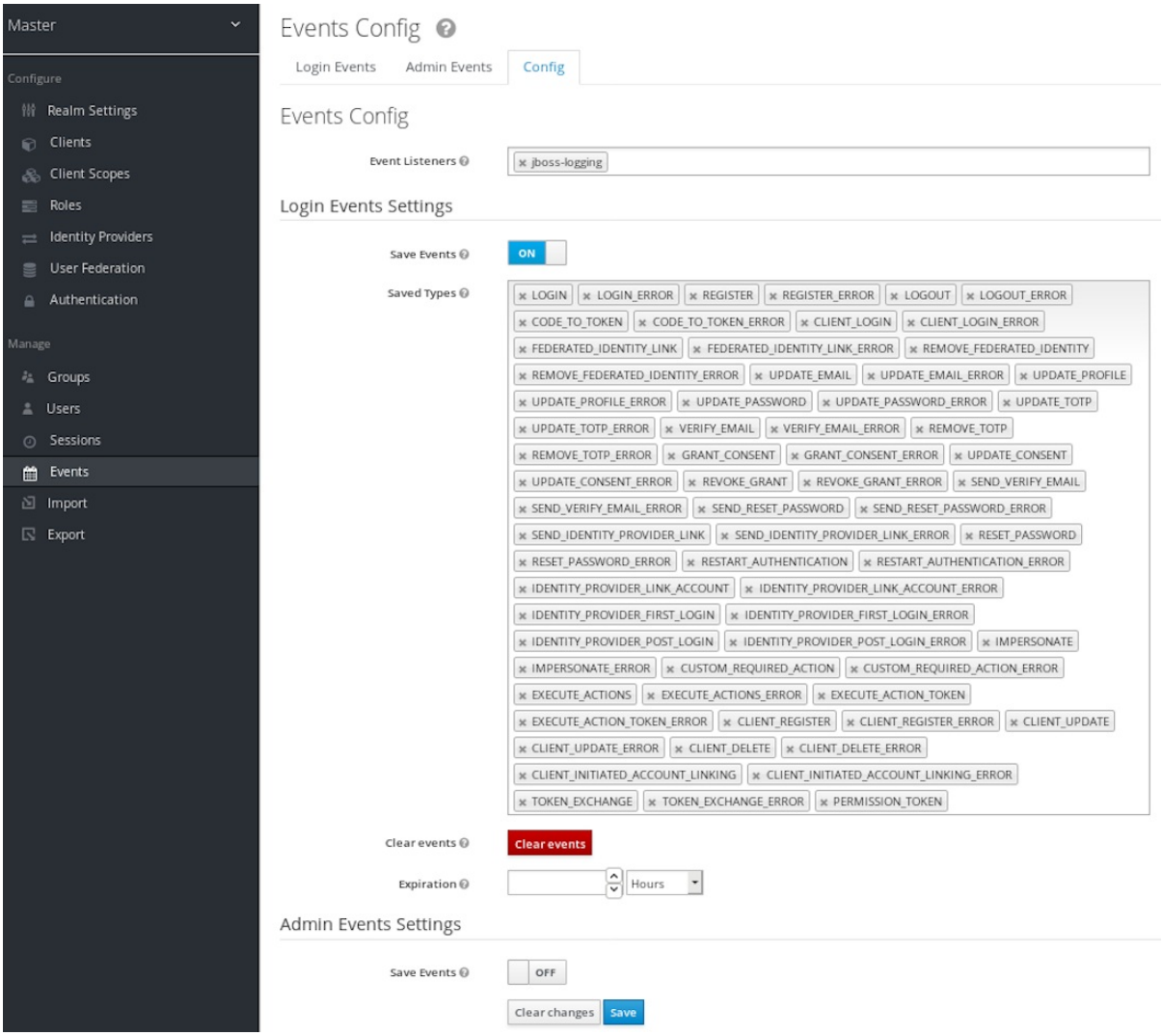
1. 点菜单中的 **Events**。
2. 点 **Config** 选项卡。

事件配置



3. *将事件 保存到 ON。*

保存事件



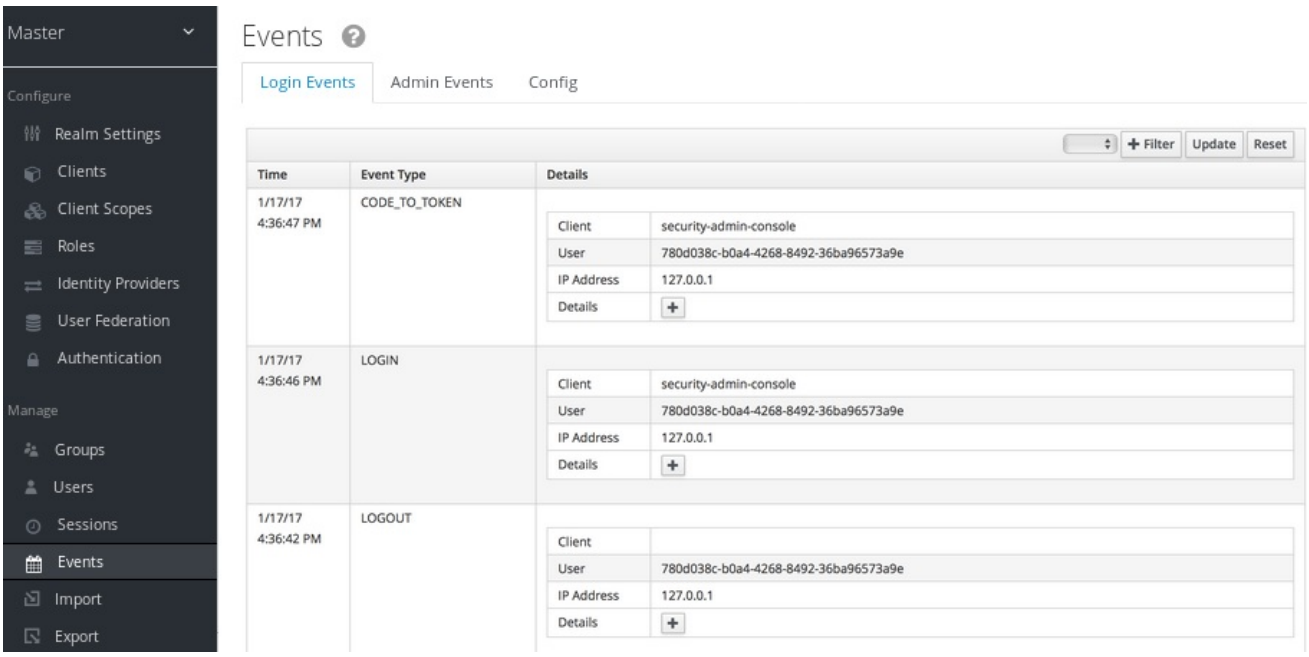
4. *在 Saved Types 字段中指定要存储的事件。*

您可以点击 清除的事件 按钮删除所有事件。

在到期字段中指定存储事件 的时间长度。当您启用登录事件存储并启用设置时，点 Save 按钮。

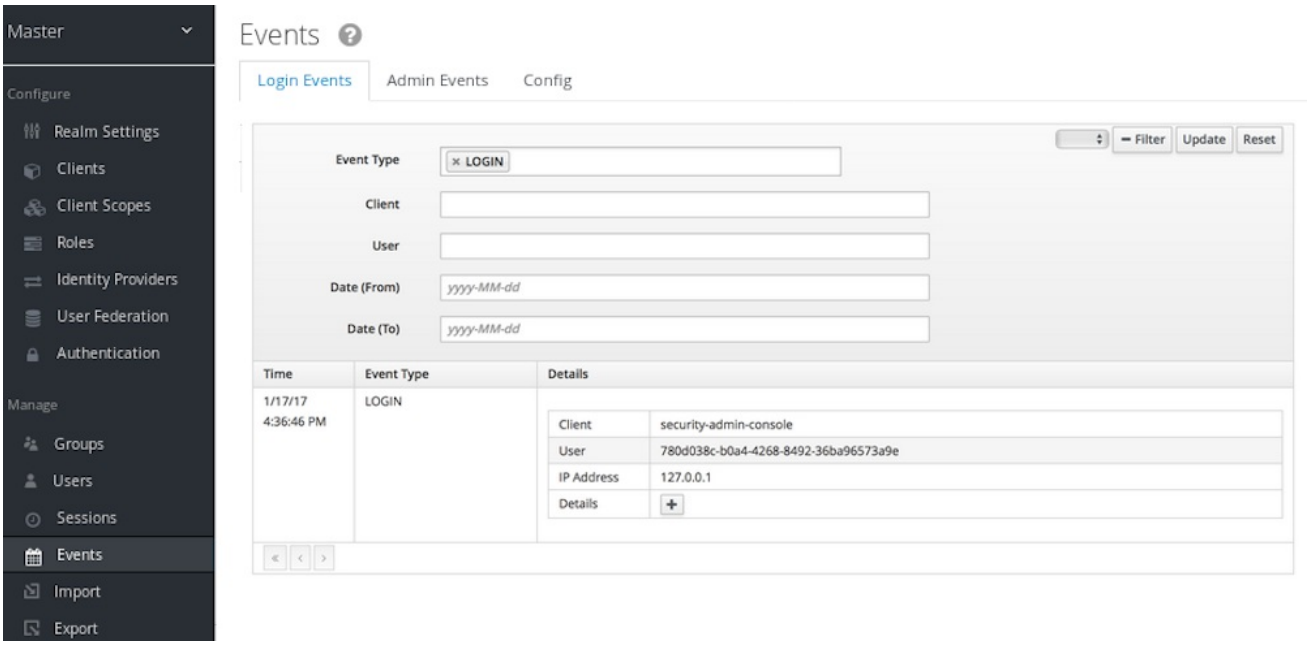
点 Login Events 选项卡查看事件。

登录事件



*您可以使用 **Filter** 按钮过滤事件。*

登录事件过滤器



*在这个示例中，我们只过滤 登录 事件。点 **Update** 运行过滤器。*

14.1.1. 事件类型

登录事件：

事件	描述
登录	用户登录。
注册	用户注册。
退出	用户注销。
令牌的代码	应用程序或客户端会交换令牌的代码。
刷新令牌	应用程序或客户端会刷新令牌。

帐户事件：

事件	描述
社交链接	用户帐户链接到社会媒体提供程序的链接。
删除社交链接	来自社交媒体帐户的链接到用户帐户位。
更新电子邮件	帐户更改的电子邮件地址。
更新配置集	帐户更改的配置集。
发送密码重置	Red Hat Single Sign-On 发送密码重置电子邮件。
更新密码	帐户更改的密码。
更新 TOTP	帐户更改基于时间的一次性密码(TOTP)设置。
删除 TOTP	Red Hat Single Sign-On 从帐户中删除 TOTP。
发送验证电子邮件	Red Hat Single Sign-On 发送一封电子邮件验证电子邮件。
验证电子邮件	红帽单点登录验证帐户的电子邮件地址。

每个事件都有对应的错误事件。

14.1.2. 事件监听程序

事件监听器侦听事件，并根据该事件执行操作。Red Hat Single Sign-On 包括两个内置监听程序，即 Logging Event Listener 和 Email Event Listener。

14.1.2.1. 日志记录事件监听程序

当启用 Logging Event Listener 时，此监听器会在发生错误事件时写入日志文件。

Logging Event Listener 中的日志消息示例：

```
11:36:09,965 WARN  [org.keycloak.events] (default task-51) type=LOGIN_ERROR, realmId=master,
        clientId=myapp,
        userId=19aeb848-96fc-44f6-b0a3-59a17570d374, ipAddress=127.0.0.1,
        error=invalid_user_credentials, auth_method=openid-connect, auth_type=code,
        redirect_uri=http://localhost:8180/myapp,
        code_id=b669da14-cdbb-41d0-b055-0810a0334607, username=admin
```

您可以使用日志记录事件 Listener 来防止黑客 bot 攻击：

1. 解析 LOGIN_ERROR 事件的日志文件。
2. 提取失败的登录事件的 IP 地址。
3. 将 IP 地址发送到入侵检测软件框架工具。

Logging Event Listener 日志事件到 org.keycloak.events 日志类别。默认情况下，Red Hat Single Sign-On 在服务器日志中不包含调试日志事件。

在服务器日志中包含 debug 日志事件：

1. 编辑 standalone.xml 文件。
2. 更改日志记录事件监听程序使用的日志级别。

另外，您可以为 org.keycloak.events 配置日志级别。

例如，要更改日志级别并添加以下内容：

```
<subsystem xmlns="urn:jboss:domain:logging:...">
  ...
  <logger category="org.keycloak.events">
    <level name="DEBUG"/>
  </logger>
</subsystem>
```

要更改日志记录事件监听程序使用的日志级别，请添加以下内容：

```
<subsystem xmlns="urn:jboss:domain:keycloak-server:...">
  ...
  <spi name="eventsListener">
    <provider name="jboss-logging" enabled="true">
      <properties>
        <property name="success-level" value="info"/>
        <property name="error-level" value="error"/>
      </properties>
    </provider>
  </spi>
</subsystem>
```

日志级别的有效值为 debug、info、warn、error、fatal。

14.1.2.2. 电子邮件事件 Listener

当事件发生时，电子邮件事件 Listener 将电子邮件发送到用户帐户，并支持以下事件：

- 登录错误。
- 更新密码.
- 更新基于时间的一次性密码(TOTP)。

- 删除基于时间的一次性密码(TOTP)。

流程

启用电子邮件 Listener：

1. 点菜单中的 Events。
2. 点 Config 选项卡。
3. 点 Event Listeners 字段。
4. 选择 电子邮件。

您可以通过编辑您的分发中包含的 standalone.xml、standalone-ha.xml 或 domain.xml 配置文件来排除事件。例如：

```
<spi name="eventsListener">
  <provider name="email" enabled="true">
    <properties>
      <property name="exclude-events" value="
[&amp;quot;UPDATE_TOTP&amp;quot;,&amp;quot;REMOVE_TOTP&amp;quot;]"/>
    </properties>
  </provider>
</spi>
```

您可以通过编辑 standalone.xml、standalone-ha.xml 或 domain.xml 配置文件，为数据库设置最大长度。如果某个字段（例如 redirect_uri）是较长的，则此设置很有用。例如：

```
<spi name="eventsStore">
    <provider name="jpa" enabled="true">
        <properties>
            <property name="max-detail-length" value="1000"/>
        </properties>
    </provider>
</spi>
```

有关 standalone.xml、standalone-ha.xml 或 domain.xml 文件的位置的详细信息，请参阅 服务器安装和配置指南。

14.2. 管理员事件

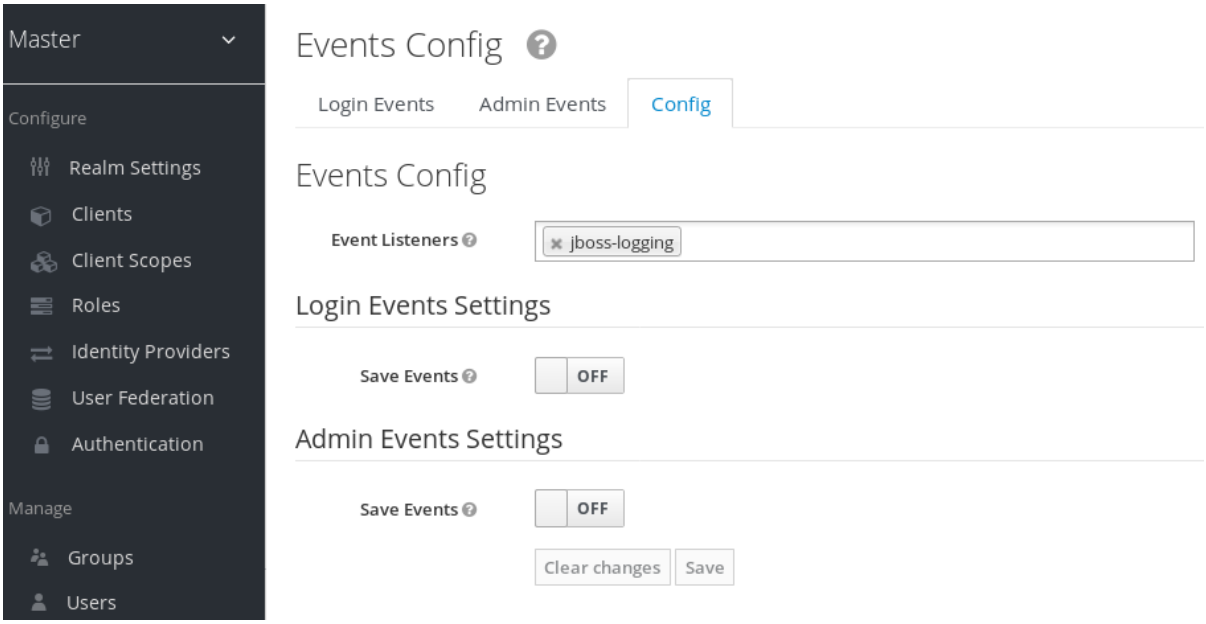
您可以记录管理员在管理控制台中由管理员执行的所有操作。Admin Console 通过调用 Red Hat Single Sign-On REST 接口和 Red Hat Single Sign-On 审计这些 REST 调用来执行管理操作。您可以在 Admin Console 中查看生成的事件。

启用 Admin 操作审计：

流程

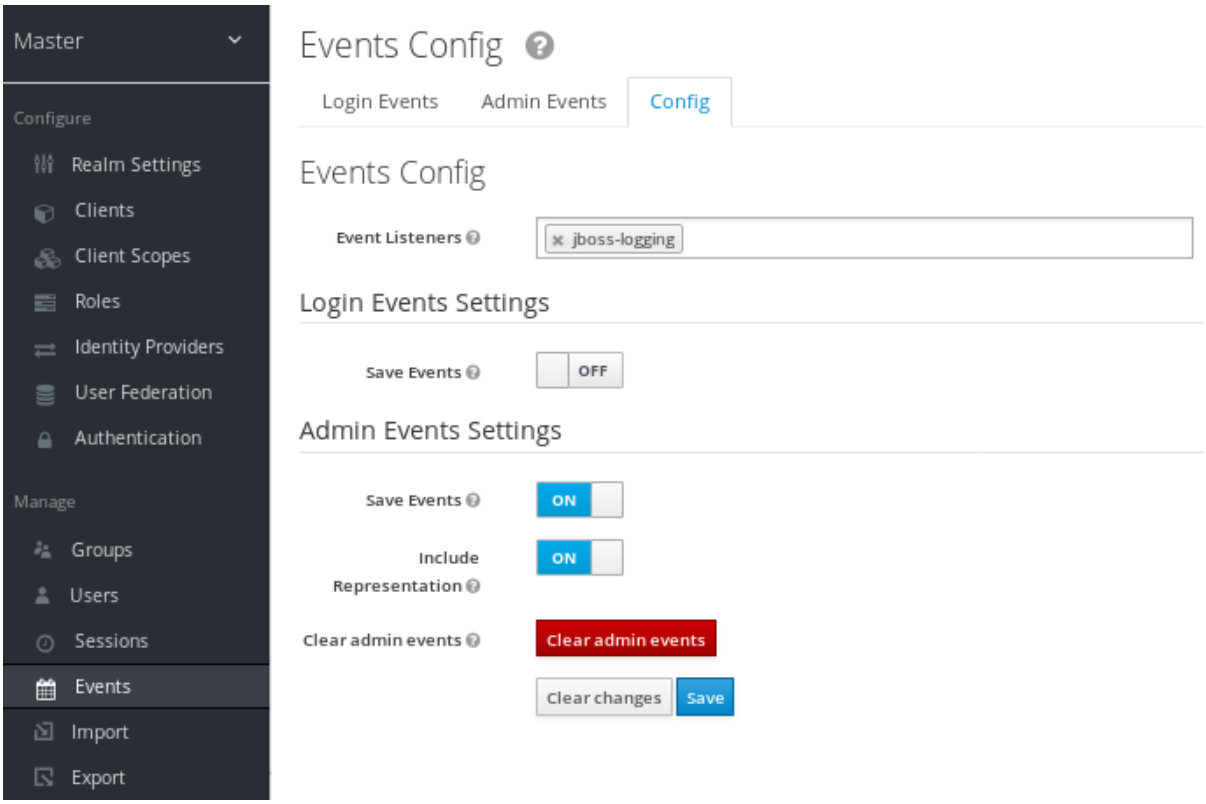
1. 点菜单中的 Events。
2. 点 Config 选项卡。

事件配置



3. 在 Admin Events Settings 部分中，将 Events 保存到 ON。Red Hat Single Sign-On 显示 Include Representation 开关。

管理员事件配置



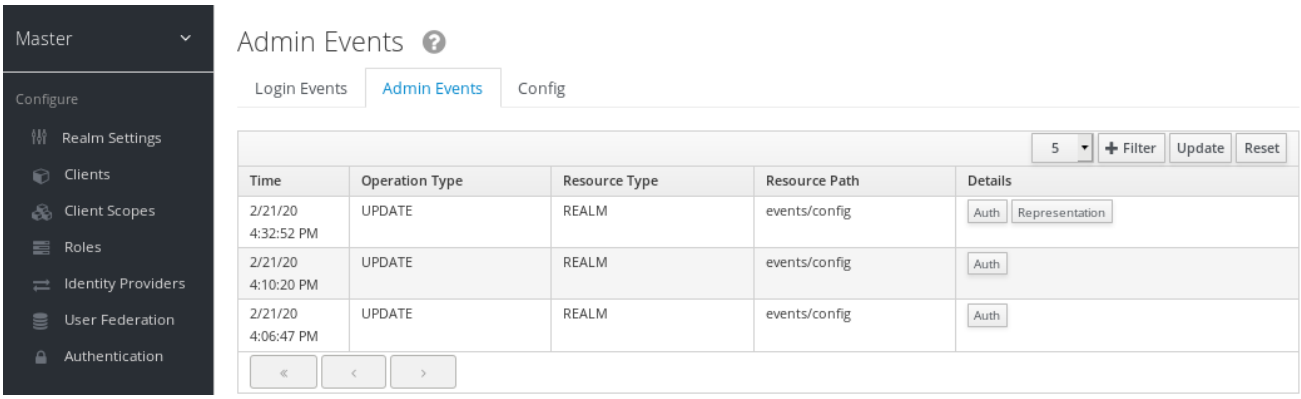
4.

*切换 **Include Representation to ON**。*

***Include Representation** 开关包括通过管理员 **REST API** 发送的 **JSON** 文档，以便您可以查看管理员的操作。若要清除存储操作的数据库，可单击 **Clear admin events**。*

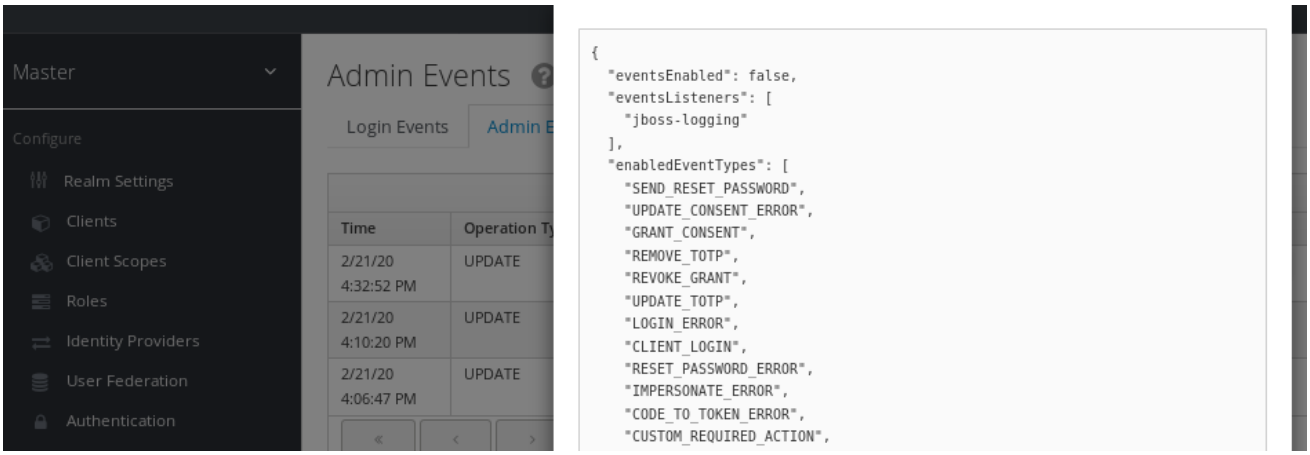
*若要查看 **admin** 事件，请单击 **Admin Events** 选项卡。*

管理员事件



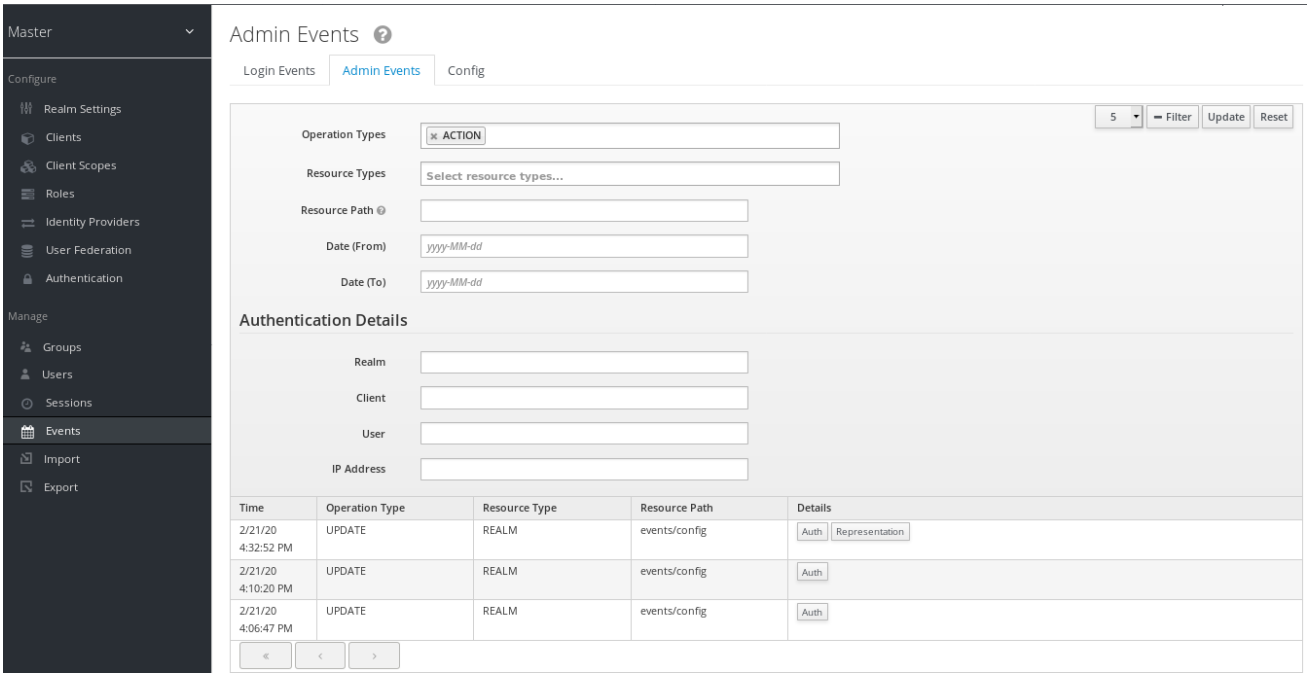
如果 Details 列有一个 Representation 按钮，请点击 代表 按钮来查看通过该操作发送的 JSON Red Hat Sign-On。

管理员表示法



点 Filter 查看特定的事件。

admin 事件过滤器



第 15 章 导入和导出数据库

红帽单点登录包括导出和导入其整个数据库的功能。

您可以将整个 Red Hat Single Sign-On 数据库从一个环境迁移到另一个环境，或迁移到另一个数据库。服务器引导时的 export/import 触发，其参数通过 Java 属性传递。

注意

由于服务器启动时导入和导出触发器，因此在导出/导入过程中对服务器或数据库执行任何操作。

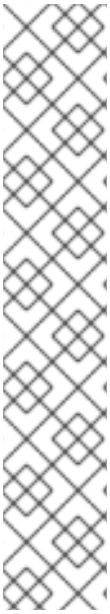
您可以导出/导入数据库以：

- 文件系统上的目录。
- 文件系统上的一个 JSON 文件。

从目录导入时，文件名必须遵循以下命名规则：

- <REALM_NAME>-realm.json.例如，名为 "acme-roadrunner-affairs-affairs-realm.json" 的域的 "acme-roadrunner-affairs"。
- <REALM_NAME>-users-<INDEX>.json.例如：名为 "acme-roadrunner-affairs-usersairs-users-0.json" 的域文件 "acme-roadrunner-affairs"

如果导出到某个目录，您可以指定每个 JSON 文件中存储的用户数量。



注意

导出到单个文件可生成大文件，因此如果您的数据库包含超过 500 个用户，则导出到一个目录而不是单个文件。将多个用户导出到目录中最佳执行，因为目录提供程序对每个"页面"（用户的文件）使用单独的事务。

每个文件的默认用户数和每个事务为五次，但您可以覆盖这个数量。如需更多信息，请参阅 keycloak.migration.usersPerFile。

导出到或从一个文件导出或导入一个事务，如果数据库包含多个用户，则可能会对性能造成影响。

导出到未加密的目录中：

```
bin/standalone.sh -Dkeycloak.migration.action=export
-Dkeycloak.migration.provider=dir -Dkeycloak.migration.dir=<DIR TO EXPORT TO>
```

导出到单个 JSON 文件：

```
bin/standalone.sh -Dkeycloak.migration.action=export
-Dkeycloak.migration.provider=singleFile -Dkeycloak.migration.file=<FILE TO EXPORT TO>
```

同样地，对于导入，请使用 **-Dkeycloak.migration.action=import**，而不是 导出。例如：

```
bin/standalone.sh -Dkeycloak.migration.action=import
-Dkeycloak.migration.provider=singleFile -Dkeycloak.migration.file=<FILE TO IMPORT>
-Dkeycloak.migration.strategy=OVERWRITE_EXISTING
```

其他命令行选项包括：

-Dkeycloak.migration.realmName

使用此属性导出专门命名的域。如果没有指定此参数，则所有域导出。

-Dkeycloak.migration.usersExportStrategy

此属性指定用户导出到什么位置。可能的值包括：

-

DIFFERENT_FILES - 用户导出到不同的文件，具体取决于 每个文件的最大用户数。DIFFERENT_FILES 是此属性的默认值。

- SKIP - Red Hat Single Sign-On 会跳过导出用户。
- REALM_FILE - 用户使用域设置导出到同一文件。该文件类似于带有域数据和用户的 "foo-realm.json"。
- SAME_FILE - 用户导出到同一文件，但与域文件不同。其结果与带有域数据和 "foo-users.json" 的 "foo-realm.json" 类似。

-Dkeycloak.migration.usersPerFile

此属性指定每个文件和数据库事务的用户数量。默认情况下，其值为 fifty。如果 keycloak.migration.usersExportStrategy 是 DIFFERENT_FILES，Red Hat Single Sign-On 会使用此属性。

-Dkeycloak.migration.strategy

Red Hat Single Sign-On 在导入时使用此属性。它指定如何在数据库中已存在具有相同名称的域。

可能的值有：

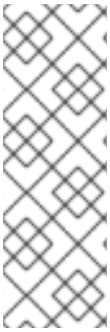
- IGNORE_EXISTING - 如果名称相同的域已存在，则不要导入一个域。
- OVERWRITE_EXISTING - 删除现有的域并使用 JSON 文件中的新数据再次导入域。使用此值从一个环境完全迁移到另一个环境。

如果您要导入不是来自 Red Hat Single Sign-On 导出的文件，请使用 keycloak.import 选项。如果您导入多个域文件，请指定以逗号分隔的文件名列表。文件名列表比前面的情况更合适，因为在 Red Hat Single Sign-On 初始化主域后会出现这种情况。

示例：

- *-Dkeycloak.import=/tmp/realm1.json*

- *-Dkeycloak.import=/tmp/realm1.json,/tmp/realm2.json*

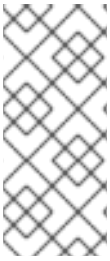


注意

您不能将 keycloak.import 参数与 keycloak.migration.X 参数一起使用。如果您将这些参数一起使用，Red Hat Single Sign-On 会忽略 keycloak.import 参数。keycloak.import 机制会忽略 Red Hat Single Sign-On 中已存在的域。keycloak.import 机制对于开发目的非常方便，但如果需要更大的灵活性，请使用 keycloak.migration.X 参数。

15.1. 管理控制台导出/导入

Red Hat Single Sign-On 从管理控制台导入大多数资源，以及导出大部分资源。Red Hat Single Sign-On 不支持用户导出。



注意

Red Hat Single Sign-On masks 属性，其中包含导出文件中的 secret 或私有信息。从管理控制台导出文件不适合在服务器之间备份或数据传输。只有引导时导出适合在服务器之间备份或数据传输。

您可以使用导出期间创建的文件从管理控制台导入。您可以从一个域导出并导入到另一个域，也可以从一个服务器导出并导入到另一个域。

注意

管理控制台导出/导入仅允许每个文件一个域。

警告

Admin Console 导入可覆盖资源。请谨慎使用这个功能，特别是生产服务器上。Admin Console Export 操作中的 JSON 文件不适用于数据导入，因为它们包含 secret 的无效值。

警告

您可以使用管理控制台导出客户端、组和角色。如果您的域中的数据库包含多个客户端、组和角色，则导出可能需要很长时间才能结束，并且 Red Hat Single Sign-On 服务器可能无法响应用户请求。请谨慎使用这个功能，特别是生产服务器上。

第 16 章 缓解安全威胁

任何验证服务器中都存在安全漏洞。如需更多信息，请参阅互联网工程任务 Force 的(IETF) OAuth 2.0 Threat Model 和 OAuth 2.0 安全最佳实践。

16.1. 主机

Red Hat Single Sign-On 以多种方式使用公共主机名，如令牌签发者字段和密码重置电子邮件中的 URL。

默认情况下，主机名从请求标头衍生而来。不存在验证以确保主机名有效。如果您不使用负载均衡器或代理，使用 Red Hat Single Sign-On 以防止无效的主机标头，请配置可接受的主机名。

主机名的服务提供商接口(SPI)提供了为请求配置主机名的方法。您可以使用此内置供应商为 frontend 请求设置固定 URL，同时根据请求 URI 允许后端请求。如果内置供应商没有所需功能，您可以开发自定义供应商。

16.2. 管理端点和管理控制台

默认情况下，Red Hat Single Sign-On 会公开管理 REST API 和 Web 控制台与非管理员用户使用情况相同的端口。如果不需要外部访问，请不要在外部公开管理端点。如果您需要外部公开管理端点，您可以直接在 Red Hat Single Sign-On 中公开它们，或使用代理。

要使用代理公开端点，请参阅代理的文档。您需要控制对 /auth/admin 端点的请求的访问。

Red Hat Single Sign-On 中有两个选项直接公开端点，IP 限制和单独的端口。

当在 Red Hat Single Sign-On 的前端 URL 上无法访问 Admin Console 时，在默认主机名供应商中配置固定的 admin URL。

16.2.1. IP 限制

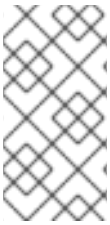
您可以将 /auth/admin 访问权限限制为特定的 IP 地址。例如，将 /auth/admin 的访问权限限制为 10.0.0.1 到 10.0.0.255 的 IP 地址。

```
<subsystem xmlns="urn:jboss:domain:undertow:12.0">
```

```
...
<server name="default-server">
    ...
    <host name="default-host" alias="localhost">
        ...
        <filter-ref name="ipAccess"/>
    </host>
</server>
<filters>
    <expression-filter name="ipAccess" expression="path-prefix('/auth/admin') -> ip-access-control(acl={'10.0.0.0/24 allow'})"/>
</filters>
...
</subsystem>
```

您还可以使用这些 CLI 命令限制对特定 IP 地址的访问：

```
/subsystem=undertow/configuration=filter/expression-filter=ipAccess:add(,expression="path-prefix[/auth/admin] -> ip-access-control(acl={'10.0.0.0/24 allow'})")
/subsystem=undertow/server=default-server/host=default-host/filter-ref=ipAccess:add()
```



注意

对于使用代理的 IP 限制，请配置代理以确保 Red Hat Single Sign-On 接收客户端 IP 地址，而不是代理 IP 地址。

16.2.2. 端口限制

您可以将 /auth/admin 公开给不同的非公开端口。例如，在端口 8444 上公开 /auth/admin，并阻止访问默认端口 8443。

```
<subsystem xmlns="urn:jboss:domain:undertow:12.0">
    ...
    <server name="default-server">
        ...
        <https-listener name="https" socket-binding="https" security-realm="ApplicationRealm" enable-http2="true"/>
        <https-listener name="https-admin" socket-binding="https-admin" security-realm="ApplicationRealm" enable-http2="true"/>
        <host name="default-host" alias="localhost">
            ...
            <filter-ref name="portAccess"/>
        </host>
    </server>
    <filters>
        <expression-filter name="portAccess" expression="path-prefix('/auth/admin') and not equals(%p, 8444) -> response-code(403)"/>
    </filters>
```

```
    ...
</subsystem>

...

<socket-binding-group name="standard-sockets" default-interface="public" port-
offset="${jboss.socket.binding.port-offset:0}">
    ...
    <socket-binding name="https" port="${jboss.https.port:8443}"/>
    <socket-binding name="https-admin" port="${jboss.https.port:8444}"/>
    ...
</socket-binding-group>
```

您可以在端口 8444 上公开 /auth/admin，并使用 CLI 命令阻止访问默认端口 8443 ：

```
/socket-binding-group=standard-sockets/socket-binding=https-admin/:add(port=8444)

/subsystem=undertow/server=default-server/https-listener=https-admin:add(socket-
binding=https-admin, security-realm=ApplicationRealm, enable-http2=true)

/subsystem=undertow/configuration=filter/expression-
filter=portAccess:add(,expression="path-prefix('/auth/admin') and not equals(%p, 8444) ->
response-code(403)")
/subsystem=undertow/server=default-server/host=default-host/filter-ref=portAccess:add()
```

16.3. 暴力攻击

通过多次尝试多次登录，会强制攻击尝试猜测用户密码。Red Hat Single Sign-On 具有暴力检测功能，并在登录失败次数超过指定阈值时暂时禁用用户帐户。

注意

默认情况下，Red Hat Single Sign-On 禁用暴力强制检测。启用此功能以防御暴力攻击。

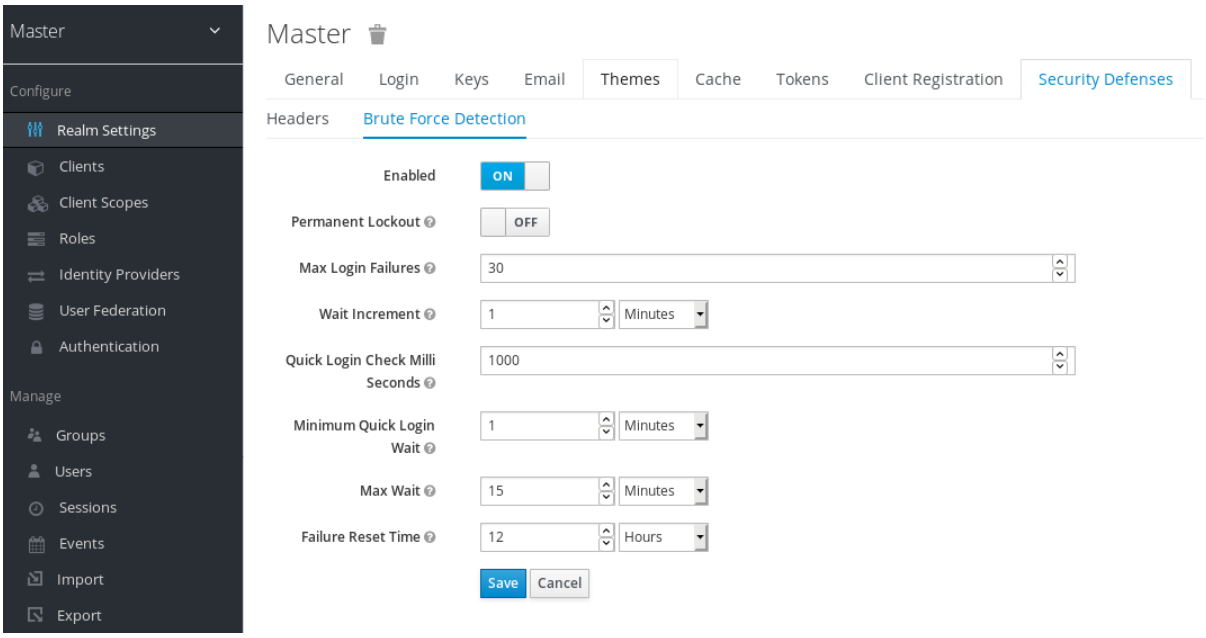
流程

启用此保护：

1. *点菜单中的 Realm Settings*
2. *点 Security Defenses 选项卡。*

3. 点 Brute Force Detection 选项卡。

暴力强制检测



Red Hat Single Sign-On 可在检测到攻击时部署永久锁定和临时锁定操作。永久锁定会禁用用户帐户，直到管理员重新启用它。临时锁定会禁用特定时间的用户帐户。禁用该帐户的时间周期会随着攻击继续继续而增加。

注意

当用户临时锁定并尝试登录时，Red Hat Single Sign-On 会显示默认的 Invalid username 或 password 错误消息。这个消息与为无效用户名或无效密码显示的消息相同，以确保攻击者不知道帐户已被禁用。

常用参数

Name	描述	默认
最大登录失败数	登录失败的最大数量。	30 个故障。
快速登录检查毫秒	登录尝试之间的最短时间。	1000 毫秒.

Name	描述	默认
最少快速登录等待	当登录尝试的速度快于 *Quick Login Check Milliseconds* 时，用户会被禁用的最短时间。	1 分钟.

永久锁定流

1. ***成功登录时***
    a. ***重置 计数***
2. ***登录失败时***
    a. ***递增 计数***
    b. ***如果 计数 大于 Max Login Failure***
        i. ***永久禁用用户***
    c. ***否则，如果这个失败和最后一个失败之间的时间小于 Quick Login Check Milliseconds***
        i. ***临时禁用用户以进行 最小快速登录 Wait***

当 Red Hat Single Sign-On 禁用一个用户时，在管理员启用该用户前无法登录。启用帐户重置 计数。

临时锁定参数

Name	描述	默认
wait Increment	当用户尝试超过 *Max Login Failure 时，在用户登录尝试超过 Max Login* Failure 时，会临时禁用用户的时间。	1 分钟.
max Wait	用户临时禁用的最长时间。	15 分钟.
故障重置时间	故障计数重置的时间。计时器从最后一次失败的登录运行。	12 小时.

临时锁定算法

1. ***成功登录时***

   a. ***重置 计数***

2. ***登录失败时***

   a. ***如果此故障和最后一次故障之间的时间大于 Failure 的重置时间***

      i. ***重置 计数***

   b. ***递增 计数***

   c. ***使用 Wait Increment * 计算等待 (计数 / Max Login Failures)。这个部门是一个整数划分到一个整数***

   d. ***如果 等待 0，且这个失败之间的时间小于 Quick Login Check Milliseconds，则把 wait 设置为 Minimum Quick Login Wait。***

      i. ***暂时禁用用户以获得较小的 等待 和 Max Wait 秒***

当临时禁用的帐户提交登录失败时，'count' 不会递增。

Red Hat Single Sign-On brute 强制检测的缺点是服务器容易受到拒绝服务攻击的影响。在实施拒绝服务攻击时，攻击者可以通过猜测其知道的帐户的密码登录，并最终导致红帽单点登录禁用帐户。

考虑使用入侵检测软件(IPS)。Red Hat Single Sign-On 会记录每个登录失败和客户端 IP 地址失败。您可以将 IPS 指向 Red Hat Single Sign-On 服务器的日志文件，IPS 可以修改防火墙来阻止来自这些 IP 地址的连接。

16.3.1. 密码策略

确保您有复杂的密码策略，以强制用户选择复杂的密码。如需更多信息，请参阅密码策略章节。???通过将红帽单点登录服务器设置为使用一次性密码，防止密码猜测。

16.4. 只读用户属性

存储在 Red Hat Single Sign-On 中的典型用户都有各种与用户配置文件相关的属性。这些属性包括电子邮件、firstName 或 lastName。但是，用户也可能具有属性，它们不是典型配置集数据，而是元数据。元数据属性通常是用户的只读，且典型的用户绝不应该有从 Red Hat Single Sign-On 用户界面或帐户 REST API 更新这些属性的方法。使用 Admin REST API 创建或更新用户时，管理员应当甚至是只读属性。

元数据属性通常是来自这些组的属性：

- 与用户存储供应商相关的各种链接或元数据。例如，如果 LDAP 集成，LDAP_ID 属性包含 LDAP 服务器中的用户的 ID。
- 用户存储调配的元数据。例如，从 LDAP 置备的 createdTimestamp 应该始终由用户或管理员只读。
- 与各种验证器相关的元数据。例如，KERBEROS_PRINCIPAL 属性可以包含特定用户的 kerberos 主体名称。同样地属性 usercertificate 可以包含与 X.509 证书中的数据绑定相关的元数据，这通常在启用了 X.509 证书身份验证时使用。
- 与应用程序/客户端用户识别器相关的元数据。例如，saml.persistent.name.id.my_app 可以包含 SAML NameID，供客户端应用程序 my_app 用作用户标识符。

- 与授权策略相关的元数据，用于基于属性的访问控制(ABAC)。这些属性的值可用于授权决策。因此，务必要能被用户更新这些属性。

从长期角度来看，Red Hat Single Sign-On 将具有适当的用户配置集 SPI，这将允许对每个用户属性进行精细配置。目前，这个功能还没有完全可用。因此，Red Hat Single Sign-On 具有用户的内部用户属性列表，这些属性是只读的，适用于在服务器级别配置的管理员。

这是只读属性的列表，这些属性由 Red Hat Single Sign-On 默认供应商及功能在内部使用，因此始终是只读的：

- 用户：
  KERBEROS_PRINCIPAL,LDAP_ID,LDAP_ENTRY_DN,CREATED_TIMESTAMP,createTimestamp,modifyTimestamp,userCertificate,saml.persistent.name.id.for.*
  ENABLED,EMAIL_VERIFIED

- 管理员：
  KERBEROS_PRINCIPAL,LDAP_ID,LDAP_ENTRY_DN,CREATED_TIMESTAMP,createTimestamp,modifyTimestamp

系统管理员可以向此列表中添加其他属性。配置目前在服务器级别可用。

您可以将此配置添加到 standalone （-*.xml 文件中到 Red Hat Single Sign-On server subsystem 的配置中：

```
<spi name="userProfile">
    <provider name="legacy-user-profile" enabled="true">
        <properties>
            <property name="read-only-attributes" value="[&amp;quot;foo&amp;quot;,&amp;quot;bar*&amp;quot;]"/>
            <property name="admin-read-only-attributes" value="[&amp;quot;foo&amp;quot;]"/>
        </properties>
    </provider>
</spi>
```

可使用 JBoss CLI 命令配置相同的方法：

```
/subsystem=keycloak-server/spi=userProfile/:add
/subsystem=keycloak-server/spi=userProfile/provider=legacy-user-profile/:add(properties=
{},enabled=true)
```

```
/subsystem=keycloak-server/spi=userProfile/provider=legacy-user-profile/:map-
put(name=properties,key=read-only-attributes,value=[foo,bar*])
/subsystem=keycloak-server/spi=userProfile/provider=legacy-user-profile/:map-
put(name=properties,key=admin-read-only-attributes,value=[foo])
```

在本例中，用户和管理员无法更新属性 foo。用户无法从 栏 编辑任何属性。因此，如 bar 或 barrier。配置不区分大小写，因此将拒绝 FOO 或 BarRier 等属性。通配符 * 仅在属性名称的末尾被支持，因此管理员可以有效地拒绝以指定字符开头的所有属性。属性中间的 * 被视为常规字符。

16.5. 点JACKING

点jacking 是用户点击用户界面元素与感应不同的用户的技术。恶意站点会在一个透明的 iFrame 中载入目标站点，覆盖一组虚拟按钮，直接放置在目标站点上的重要按钮下。当用户点击可见按钮时，会单击隐藏页面中的按钮。攻击者可以利用这个方法获取用户的身份验证凭证并访问其资源。

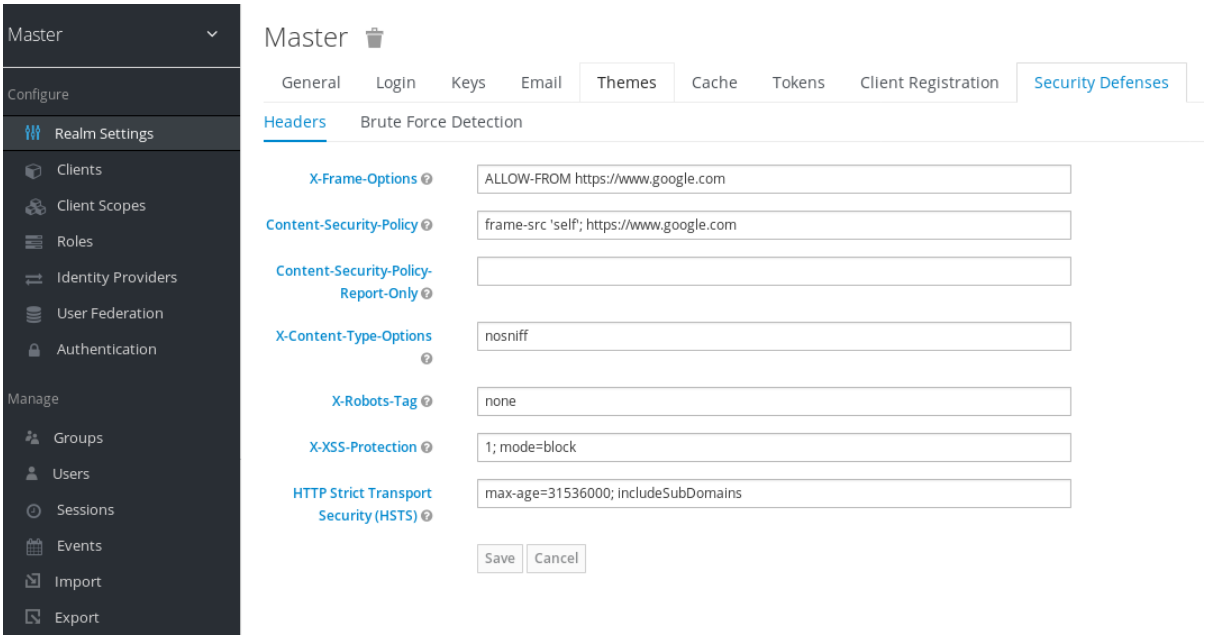
默认情况下，Red Hat Single Sign-On 的每个响应都会设置一些特定的 HTTP 标头，以防止发生这种情况。特别是，它会设置 X-Frame-Options 和 Content-Security-Policy。您应该了解这两个标头的定义，因为您可以控制大量精细的浏览器访问权限。

流程

在管理控制台中，您可以指定 X-Frame-Options 和 Content-Security-Policy 标头的值。

1. 点 Realm Settings 菜单项。
2. 点 Security Defenses 选项卡。

安全防御



默认情况下，Red Hat Single Sign-On 只为 iframes 设置 相同的 原始策略。

16.6. SSL/HTTPS 要求

OAuth 2.0/OpenID Connect 使用访问令牌来提高安全性。攻击者可以扫描您的网络以获取访问令牌，并使用它们执行令牌有权限的恶意操作。这个攻击被称为中间人攻击。使用 SSL/HTTPS 进行 Red Hat Single Sign-On auth 服务器与红帽单点登录安全的客户端之间的通信，以防止中间人攻击。

Red Hat Single Sign-On 有三种 SSL/HTTPS 模式。SSL 易于设置，因此 Red Hat Single Sign-On 允许通过 localhost、192.x.x 和其他专用 IP 地址等私有 IP 地址进行非 HTTPS 通信。在生产中，请确保为所有操作启用 SSL 和 SSL。

在适配器/客户端中，您可以禁用 SSL 信任管理器。信任管理器确保 Red Hat Single Sign-On 通信人的身份有效，并确保 DNS 域名针对服务器证书。在生产环境中，请确保每个客户端适配器都使用信任存储来防止 DNS man-in-the-middle 攻击。

16.7. CSRF 攻击

Cross-site request forgery (CSRF)攻击使用来自该网站的用户的 HTTP 请求已通过身份验证。使用基于 Cookie 的验证的任何站点都容易受到 CSRF 攻击。您可以通过将状态 Cookie 与发布的形式或查询参数匹配来缓解这些攻击。

OAuth 2.0 登录规格要求状态 Cookie 与传输的 state 参数匹配。Red Hat Single Sign-On 会完全实

现此部分规格，因此所有登录都受到保护。

Red Hat Single Sign-On Admin Console 是一个 JavaScript/HTML5 应用程序，它为后端的 Red Hat Single Sign-On admin REST API 发出 REST 调用。这些调用都需要 bearer 令牌身份验证，并且由 JavaScript Ajax 调用组成，因此 CSRF 无法进行。您可以配置 admin REST API 以验证 CORS 源。

Red Hat Single Sign-On 中的用户帐户管理部分可能会受到 CSRF 的影响。为防止 CSRF 攻击，Red Hat Single Sign-On 会设置一个 state cookie，并以隐藏的形式字段或查询参数嵌入到操作链接中。Red Hat Single Sign-On 会根据状态 Cookie 检查查询/form 参数，以验证用户是否进行调用。

16.8. UNSPECIFIC REDIRECT URI

使您注册的重定向 URI 为可行的。注册对授权代码流的 URI 可以允许恶意客户端模拟另一客户端具有更大访问权限。???如果两个客户端同时位于同一域下，则可能会出现模拟情况，例如：

16.9. FAPI 合规性

为确保 Red Hat Single Sign-On 服务器将验证您的客户端是否更加安全且兼容 FAPI，您可以为 FAPI 支持配置客户端策略。有关保护应用程序和服务 指南中的 FAPI 部分的详细信息。另外，这有助于确保上述某些安全最佳实践（如客户端所需的 SSL）、安全重定向 URI 以及类似最佳实践的保护。

16.10. 入侵访问并刷新令牌

Red Hat Single Sign-On 包含多个操作，以防止恶意执行者窃取访问令牌并刷新令牌。关键操作是强制红帽单点登录及其客户端和应用程序之间的 SSL/HTTPS 通信。默认情况下，Red Hat Single Sign-On 不会启用 SSL。

缓解泄漏访问令牌的破坏的另一个操作是缩短令牌的寿命。您可以在 超时页面中 指定令牌生命周期。访问令牌的短寿命将强制客户端和应用在短时间内刷新其访问令牌。如果管理员检测到泄漏，管理员可以注销所有用户会话以无效这些刷新令牌或设置撤销策略。

确保刷新令牌始终保持与客户端的私有，并且永远不会传输。

您可以将这些令牌作为密钥令牌的拥有者签发，从而缓解泄漏访问令牌的影响，并刷新令牌。如需更多信息，请参阅 OAuth 2.0 通用 TLS 客户端证书颁发机构 访问令牌。

如果访问令牌或刷新令牌被破坏，访问 Admin Console，并将 not- before before revocation 策略推送到所有应用程序。推送非前策略可确保在那个时间失效之前发出的任何令牌。推送新的非前策略可确保应用程序必须从 Red Hat Single Sign-On 下载新的公钥，并缓解来自已被破坏的域签名密钥的破坏。如需更多信息，请参阅 密钥章节。

如果特定应用程序、客户端或用户被破坏，您可以禁用它们。

16.11. 被破坏的授权代码

对于 OIDC Auth Code 流，Red Hat Single Sign-On 会为其授权代码生成加密随机值。授权代码仅使用一次来获取访问令牌。

在 管理控制台的超时页面中，您可以指定授权代码有效的时长。确保长度小于 10 秒，客户端有足够的时间从代码请求令牌。

您还可以通过将代码 交换(PKCE) 应用到客户端，防止泄漏授权代码。

16.12. OPEN REDIRECTORS

Open redirector 是一个端点，使用参数自动将用户代理重定向到参数值指定的位置，而无需验证。攻击者可以使用最终用户授权端点和重定向 URI 参数，使用授权服务器以开放重定向形式使用授权服务器，在授权服务器中使用用户信任的攻击。

Red Hat Single Sign-On 要求所有注册的应用程序和客户端至少注册一个重定向 URI 模式。当 Red Hat Single Sign-On 执行重定向的客户端请求时，Red Hat Single Sign-On 会针对有效注册 URI 模式列表检查重定向 URI。客户端和应用程序必须尽可能注册为特定 URI 模式，以减少开放重定向器攻击。

如果应用程序需要一个非 http (s)自定义方案，它应该是验证模式的显式部分（如 custom:/app8:0:1::）。为了安全起见，常规模式（如 * ）并不涵盖非 http。

16.13. 密码数据库已被破坏

Red Hat Single Sign-On 不会将密码存储在原始文本中，而是以哈希文本形式使用 PBKDF2 哈希算法。Red Hat Single Sign-On 执行 27,500 个散列迭代，这是安全社区所建议的迭代数量。这种哈希迭代数量可能会对性能造成负面影响，因为 PBKDF2 哈希则使用大量 CPU 资源。

16.14. 限制范围

默认情况下，新的客户端应用程序具有无限 角色范围映射。该客户端的每个访问令牌都包含用户拥有的所有权限。如果攻击者破坏客户端并获取客户端的访问令牌，则每个用户可以访问的每个系统都会被破坏。

使用每个客户端的 Scope 菜单 来限制访问令牌的角色。或者，您可以在客户端范围级别设置角色范围映射，并使用 Client Scope 菜单 为客户端分配客户端范围映射。

16.15. 限制令牌对象

在服务间具有低信任程度的环境中，限制令牌的受众。如需更多信息，请参阅 OAuth2 Threat Model 和 Audience Support 部分。

16.16. 限制身份验证会话

在 Web 浏览器中第一次打开登录页面时，Red Hat Single Sign-On 会创建一个名为 authentication session 的对象，以存储请求的一些有用信息。每当从同一浏览器中的不同标签页打开新的登录页面时，Red Hat Single Sign-On 会创建一个名为 authentication sub-session （存储在身份验证会话内）的新记录。身份验证请求可以来自任何类型的客户端，如管理 CLI。在这种情况下，也会使用一个身份验证子会话创建一个新的身份验证会话。请注意，除使用浏览器流外，也可以以其他方式创建身份验证会话。无论源流是什么，以下文本都适用。

注意

本节论述了使用 RHDG 供应商进行身份验证会话的部署。

身份验证会话在内部存储为 RootAuthenticationSessionEntity。每个 RootAuthenticationSessionEntity 可将多个身份验证子会话存储在 RootAuthenticationSessionEntity 中，作为 AuthenticationSessionEntity 对象的集合。红帽单点登录将身份验证会话存储在专用的 RHDG 缓存中。Root AuthenticationSessionEntity 每 RootAuthenticationSessionEntity 的 AuthenticationSessionEntity 有助于每个缓存条目的大小。身份验证会话缓存的内存占用总量由存储 RootAuthenticationSessionEntity 数以及每个 Root AuthenticationSessionEntity 中的 AuthenticationSessionEntity 数决定。

维护的 RootAuthenticationSessionEntity 对象数量与浏览器中未完成登录流的数量对应。要保持 RootAuthenticationSessionEntity 的数量，建议使用高级防火墙控制来限制入站网络流量。

部署可能会出现较高的内存用量，其中有很多活跃的 RootAuthenticationSessionEntity 有很多 AuthenticationSessionEntity。如果负载均衡器不支持或者没有为 会话粘性 配置，则集群中的网络的负

载可能会显著提高。造成此负载的原因是，每个请求都位于不拥有适当身份验证会话的节点上，需要检索和更新所有者节点中的身份验证会话记录，该记录涉及检索和存储的不同网络传输。

通过设置属性 authSessionsLimit，可以在 authenticationSessions SPI 中配置每个 Root AuthenticationSessionEntity 的最大数量。默认值按照 Root AuthenticationSessionEntity 被设置为 300 AuthenticationSessionEntity。达到这个限制时，新的身份验证会话请求后将删除最旧的身份验证子会话。

以下示例演示了如何将每个 RootAuthenticationSessionEntity 的活跃 AuthenticationSessionEntity 数限制为 100。

```
<subsystem xmlns="urn:jboss:domain:keycloak-server:1.2">
    ...
    <spi name="authenticationSessions">
        <default-provider>infinispan</default-provider>
        <provider name="infinispan" enabled="true">
            <properties>
                <property name="authSessionsLimit" value="100"/>
            </properties>
        </provider>
    </spi>
    ...
</subsystem>
```

使用 CLI 命令进行等同的配置：

```
/subsystem=keycloak-server/spi=authenticationSessions:add(default-provider=infinispan)
/subsystem=keycloak-server/spi=authenticationSessions/provider=infinispan:add(properties=
{authSessionsLimit => "100"},enabled=true)
```

16.17. SQL 注入攻击

目前，Red Hat Single Sign-On 没有已知的 SQL 注入漏洞。

第 17 章 帐户控制台

红帽单点登录用户可以通过帐户控制台管理其帐户。用户可以配置其配置文件，添加双因素身份验证，包括身份提供程序帐户，以及监督设备活动。

其他资源

- 帐户控制台可根据外观和语言首选项进行配置。例如，点击 个人信息链接并完成并完成 并保存的详情， 可以在个人信息 页面中添加属性。如需更多信息，请参阅《参考 https://access.redhat.com/documentation/zh-cn/red_hat_single_sign-on/7.6/html-single/server_developer_guide/[Server 开发人员指南》]。

17.1. 访问帐户控制台

任何用户都可以访问帐户控制台。

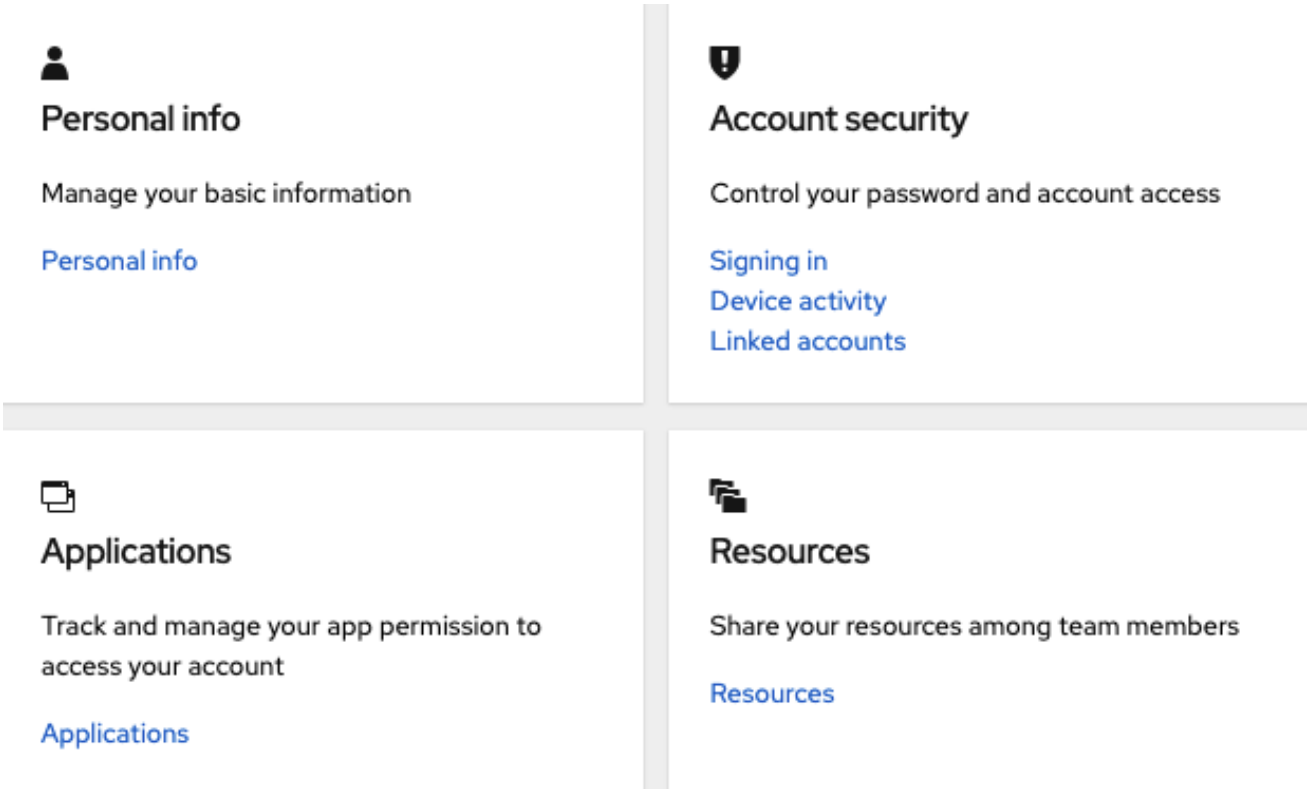
流程

1. 记录您的帐户存在的 Red Hat Single Sign-On 服务器的域名和 IP 地址。

2. 在 Web 浏览器中，以以下格式输入 URL： < server-root>/auth/realms/{realm-name}/account。

3. 输入您的登录名和密码。

帐户控制台



17.2. 配置登录方法

您可以使用基本身份验证（用户名和密码）或双因素身份验证登录到此控制台。对于双因素身份验证，请使用以下步骤之一。

17.2.1. 使用 OTP 进行双因素身份验证

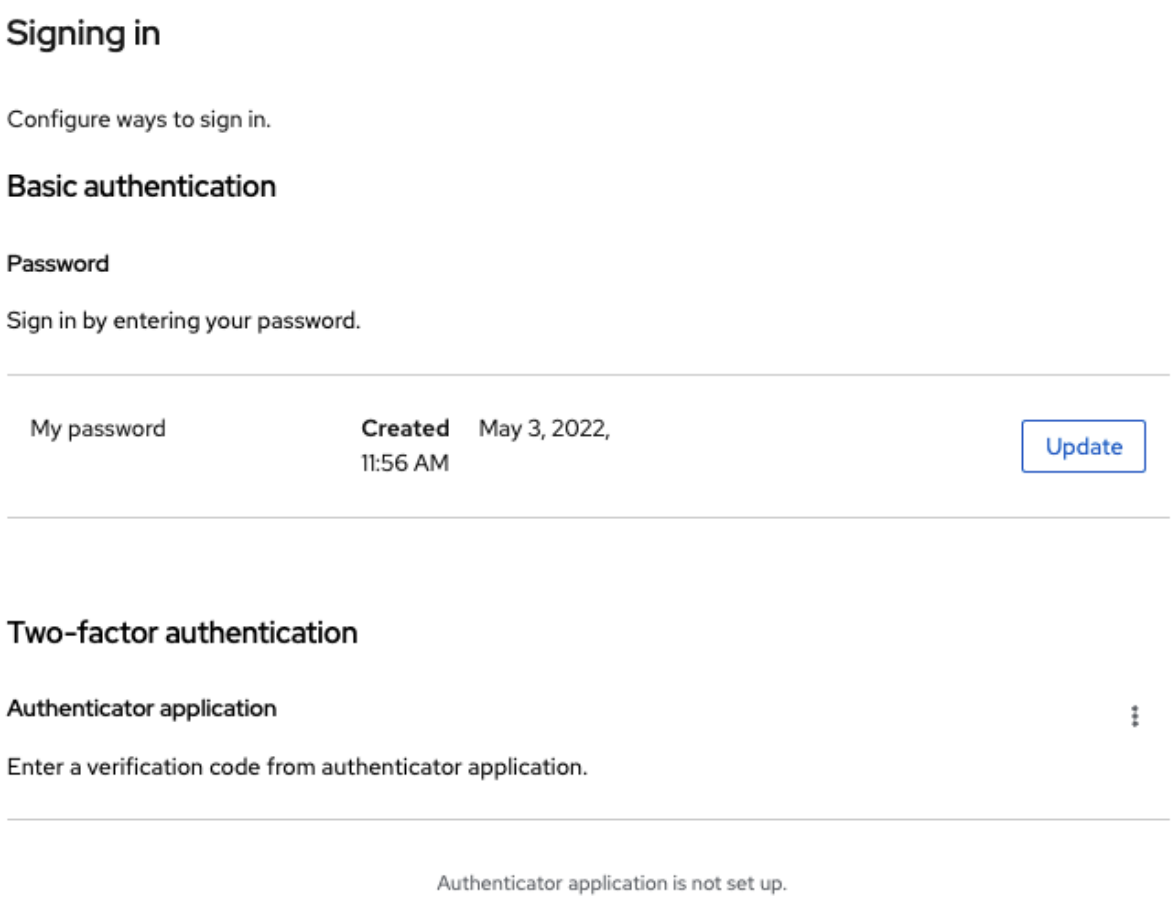
先决条件

- OTP 是您的域的有效身份验证机制。

流程

1. 在菜单中点 **Account security**。
2. 单击 中的 **Signing**。
3. 点 **Set up authenticator** 应用程序。

登录



4. 按照屏幕上出现的指示，使用移动设备中的 FreeOTP 或 Google Authenticator 作为 OTP 生成器。

5. 将屏幕中的 QR 代码扫描到移动设备上的 OTP 生成器中。

6. 注销并重新登录。

7. 通过输入在您的移动设备上提供的 OTP 来响应提示。

17.2.2. 使用 WebAuthn 的双因素身份验证

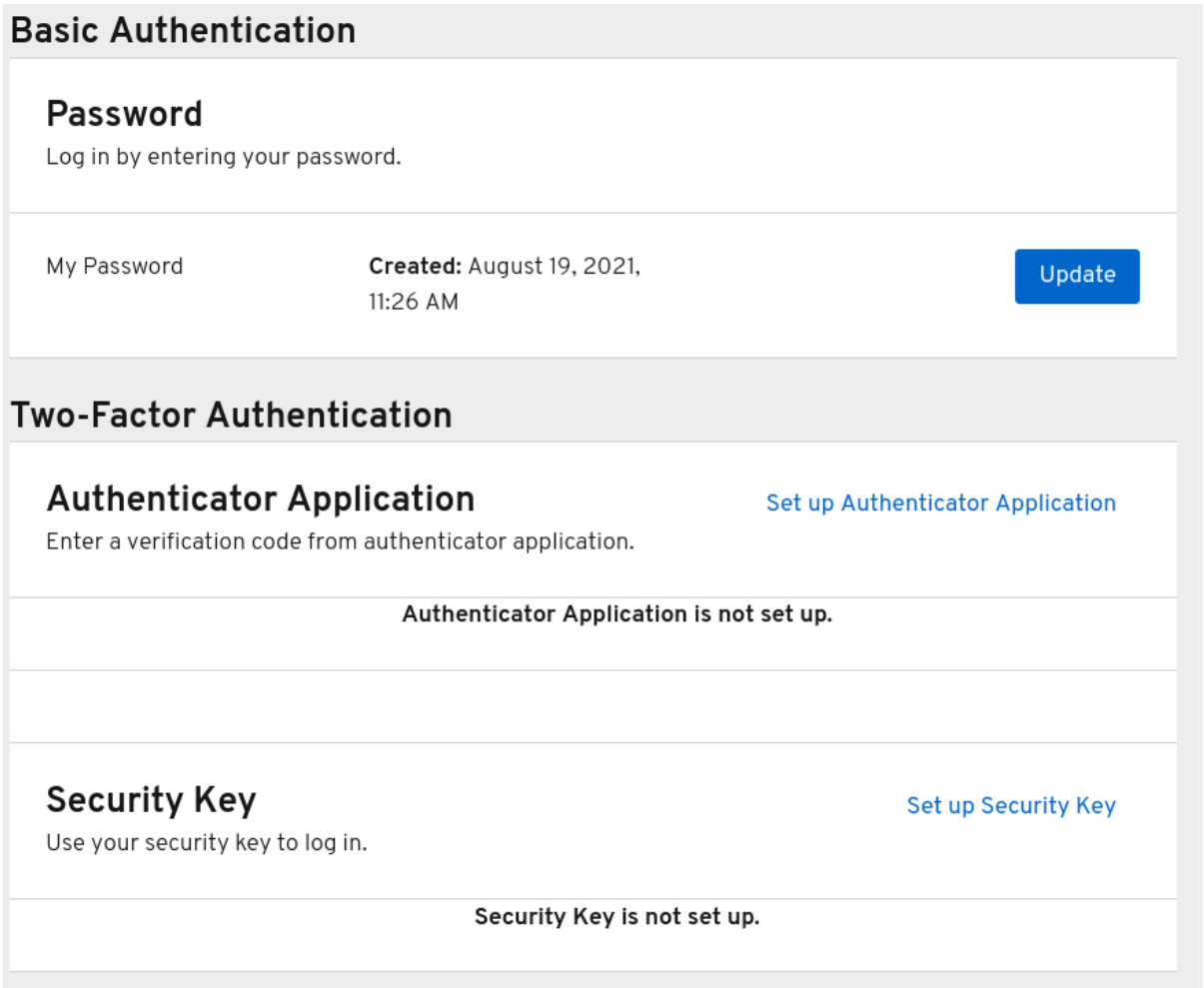
先决条件

- *WebAuthn 是您的域的有效双因素身份验证机制。请按照 WebAuthn 部分了解更多详情。*

流程

1. *在菜单中点 Account Security。*
2. *点 Signing In。*
3. *单击 Set up Security Key。*

   签署至



4. *准备 WebAuthn 安全密钥。您准备此密钥的方式取决于您使用的 WebAuthn 安全密钥的类型。例如，对于基于 USB 的 Yubikey，可能需要将密钥放在笔记本电脑的 USB 端口中。*

5. 点 Register 注册您的安全密钥。

6. 注销并重新登录。

7. 假设正确设置了身份验证流，会出现一条信息，要求您使用您的安全密钥作为第二个因素进行身份验证。

17.2.3. 使用 WebAuthn 进行免密码身份验证

先决条件

- WebAuthn 是您域的有效免密码身份验证机制。请参阅 "无密码 WebAuthn"部分 以了解更多详细信息。

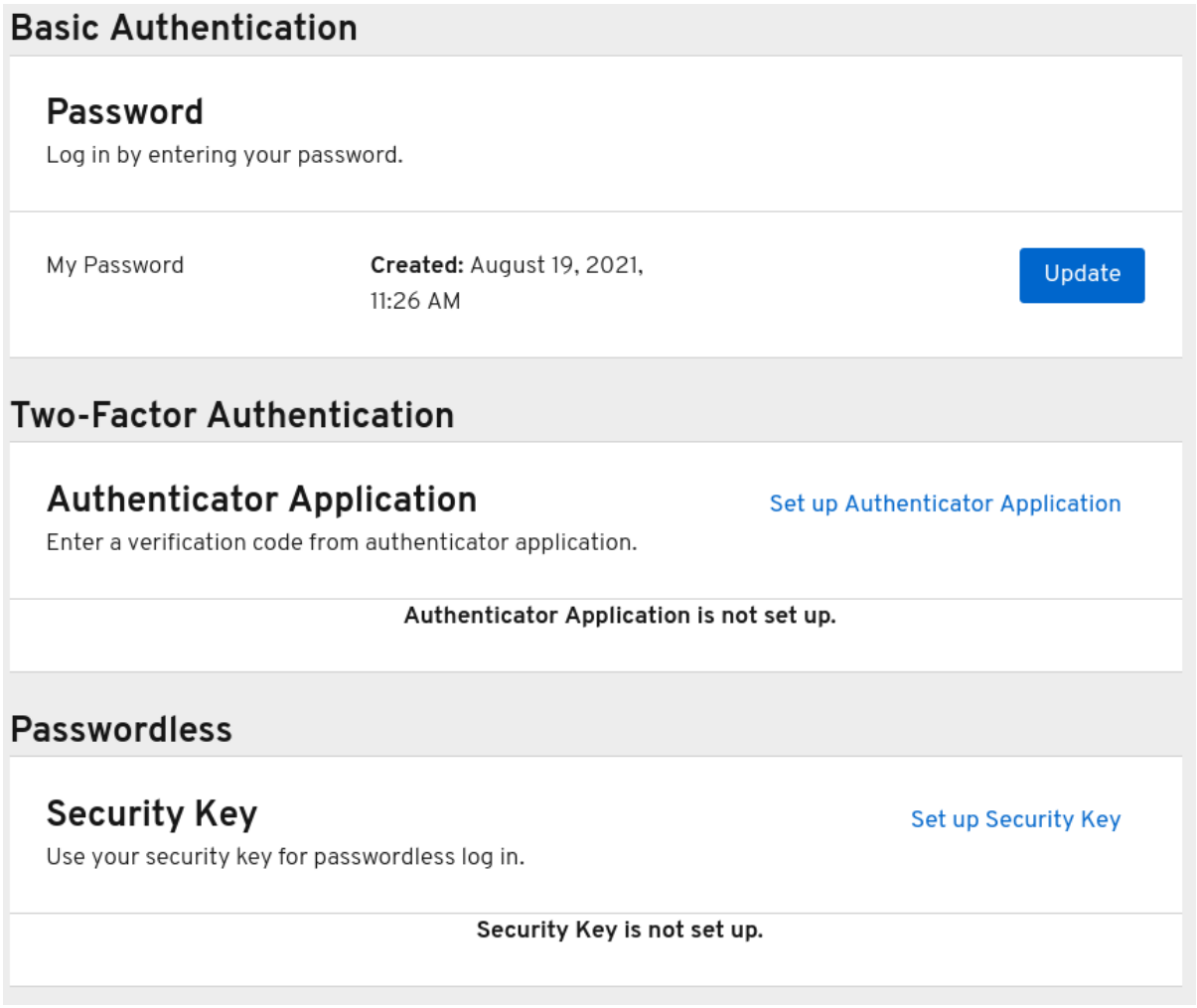
流程

1. 在菜单中点 Account Security。

2. 点 Signing In。

3. 在 Passwordless 部分中，单击 Set up Security Key。

签署至



4. *准备 WebAuthn 安全密钥。您准备此密钥的方式取决于您使用的 WebAuthn 安全密钥的类型。例如，对于基于 USB 的 Yubikey，可能需要将密钥放在笔记本电脑的 USB 端口中。*

5. *点 Register 注册您的安全密钥。*

6. *注销并重新登录。*

7. *假设正确设置了身份验证流，会出现一条信息，要求您使用您的安全密钥作为第二个因素进行身份验证。您不再需要提供密码才能登录。*

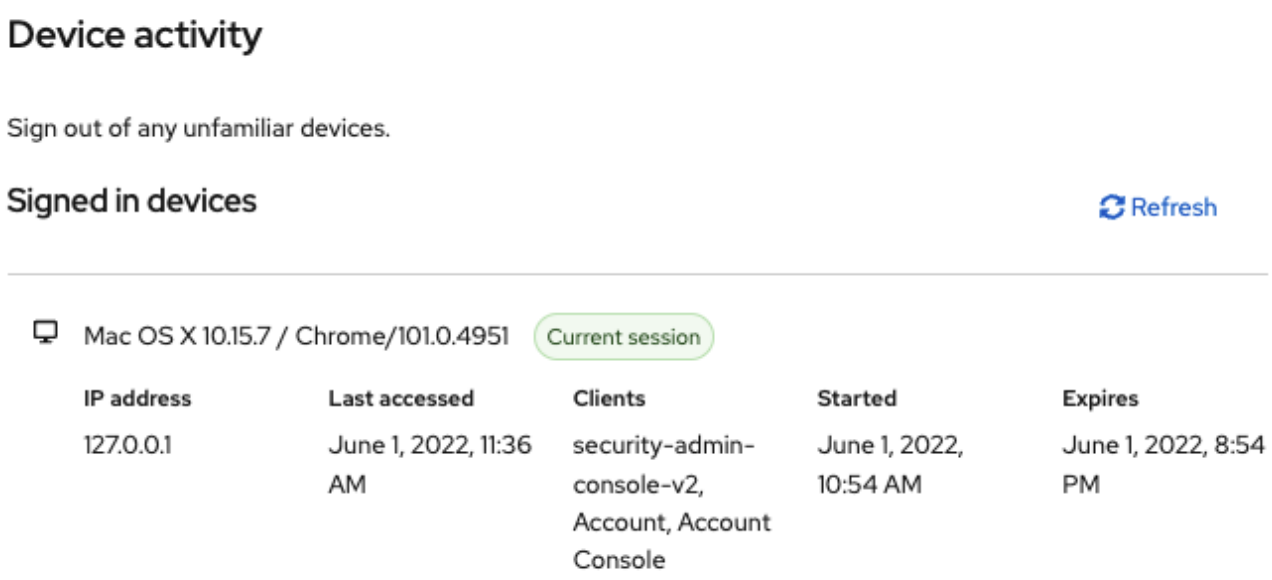
17.3. 查看设备活动

您可以查看登录到您的帐户的设备。

流程

1. *在菜单中点 Account security。*
2. *点 Device activity。*
3. *如果设备看起来可疑，请注销设备。*

Devices



17.4. 添加身份提供程序帐户

您可以将您的帐户与 身份代理 链接。这个选项通常用于链接社交供应商帐户。

流程

1. *登录 Admin 控制台。*
2. *在菜单中，单击 Identity provider。*

3. 选择供应商并填写字段。

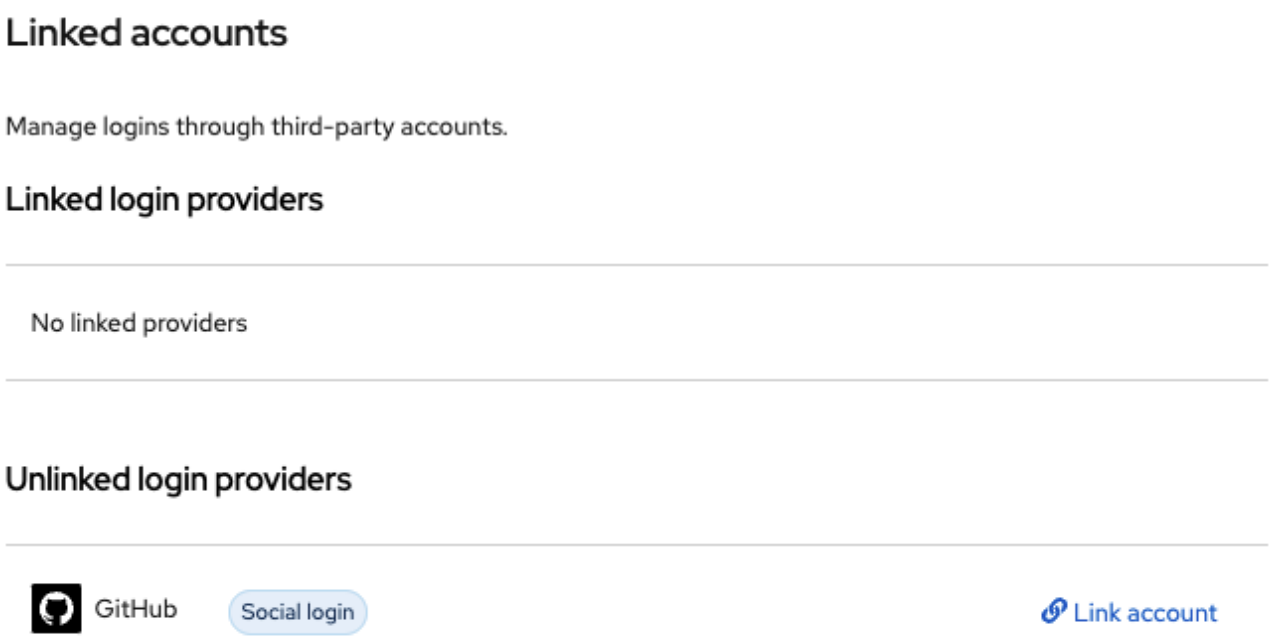
4. 返回到帐户控制台。

5. 在菜单中点 Account security。

6. 点 Linked accounts。

您添加的身份供应商会出现在本页中。

链接的帐户



17.5. 访问其他应用程序

Applications 菜单项显示您可以访问哪些应用程序。在这种情况下，只有帐户控制台可用。

应用程序

Applications

Manage your application permissions.

	Name	Application type	Status
›	Security-admin-console-v2	Internal	In use
›	Account	Internal	In use
›	Account Console	Internal	In use

第 18 章 ADMIN CLI

使用红帽单点登录，您可以使用 CLI 命令行工具从命令行界面(CLI)执行管理任务。

18.1. 安装管理 CLI

Red Hat Single Sign-On 使用 bin 目录中的执行脚本打包 Admin CLI 服务器分发。

Linux 脚本称为 kcadm.sh，Windows 的脚本名为 kcadm.bat。将 Red Hat Single Sign-On 服务器目录添加到您的 PATH 中，以使用客户端来自文件系统的任意位置。

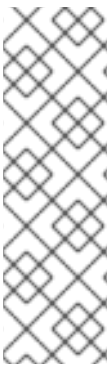
例如：

- Linux：

```
$ export PATH=$PATH:$KEYCLOAK_HOME/bin
$ kcadm.sh
```

- Windows：

```
c:\> set PATH=%PATH%;%KEYCLOAK_HOME%\bin
c:\> kcadm
```



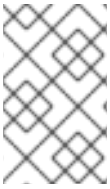
注意

您必须将 KEYCLOAK_HOME 环境变量设置为提取红帽单点登录服务器分发的路径。

为避免重复操作，本文档的其余部分只使用 Windows 示例，其中 CLI 差别不仅仅在 kcadm 命令名称中。

18.2. 使用 ADMIN CLI

Admin CLI 向 Admin REST 端点发出 HTTP 请求。访问 Admin REST 端点需要进行身份验证。



注意

有关特定端点的 JSON 属性的详细信息，请参阅 Admin REST API 文档。

1. 通过登录来启动经过身份验证的会话。现在，您可以执行创建、读取、更新和删除(CRUD)操作。

例如：

- Linux：

```
$ kcadm.sh config credentials --server http://localhost:8080/auth --realm demo --user
admin --client admin
$ kcadm.sh create realms -s realm=demorealm -s enabled=true -o
$ CID=$(kcadm.sh create clients -r demorealm -s clientId=my_client -s 'redirectUris=
["http://localhost:8980/myapp/*"]' -i)
$ kcadm.sh get clients/$CID/installation/providers/keycloak-oidc-keycloak-json
```

- Windows：

```
c:\> kcadm config credentials --server http://localhost:8080/auth --realm demo --user
admin --client admin
c:\> kcadm create realms -s realm=demorealm -s enabled=true -o
c:\> kcadm create clients -r demorealm -s clientId=my_client -s "redirectUris=
[\"http://localhost:8980/myapp/*\"]" -i > clientid.txt
c:\> set /p CID=<clientid.txt
c:\> kcadm get clients/%CID%/installation/providers/keycloak-oidc-keycloak-json
```

2. 在生产环境中，使用 https 访问 Red Hat Single Sign-On 以避免公开令牌。如果一个可信证书颁发机构（包含在 Java 的默认证书信任存储中）未发布服务器证书，请准备 truststore.jks 文件并指示 Admin CLI 使用它。

例如：

- Linux：

```
$ kcadm.sh config truststore --trustpass $PASSWORD ~/.keycloak/truststore.jks
```

- **Windows：**

```
c:\> kcadm config truststore --trustpass %PASSWORD%
%HOMEPATH%\.keycloak\truststore.jks
```

18.3. 身份验证

使用 Admin CLI 登录时，可以指定：

- 服务器端点 URL
- 一个域
- 用户名

另一个选项是仅指定 clientId，它为您创建唯一的服务帐户。

使用用户名登录时，使用指定用户的密码。当使用 clientId 登录时，您只需要客户端 secret，而不是用户密码。您还可以使用已签名的 JWT 而不是客户端机密。

确保用于会话的帐户具有调用 Admin REST API 操作的适当权限。例如，realm-management 客户端的 realm-admin 角色可以管理该用户的域。

有两种主要机制可用于验证。种机制使用 kcadm 配置凭证来启动经过身份验证的会话。

```
$ kcadm.sh config credentials --server http://localhost:8080/auth --realm master --user admin --
password admin
```

这种机制通过在 kcadm 命令调用之间维护一个经过身份验证的会话，方法是保存获取的访问令牌及其关联的刷新令牌。它可以在私有配置文件中维护其他 secret。如需更多信息，请参阅 下一章节。

第二个机制在调用期间验证每个命令调用。这种机制会增加服务器上的负载，往返用时所用的时间获取令牌。这种方法的优势在于，在调用之间不需要保存令牌，因此不会保存到磁盘。当指定了 --no-config

参数时，Red Hat Single Sign-On 会使用此模式。

例如，在执行操作时，指定身份验证所需的所有信息。

```
$ kcadm.sh get realms --no-config --server http://localhost:8080/auth --realm master --user admin --password admin
```

运行 kcadm.sh help 命令，以了解有关使用 Admin CLI 的更多信息。

运行 kcadm.sh config credentials --help 命令，以了解有关启动经过身份验证的会话的更多信息。

18.4. 使用其他配置

默认情况下，Admin CLI 维护一个名为 kcadm.config 的配置文件。Red Hat Single Sign-On 将此文件放置在用户的主目录中。在基于 Linux 的系统中，完整路径名称为 $HOME/.keycloak/kcadm.config。在 Windows 中，完整路径名是 %HOMEPATH%\.keycloak\kcadm.config。

您可以使用 --config 选项指向其他文件或位置，以便您可以并行维护多个经过身份验证的用户会话。

注意

从单个线程执行与单个配置文件关联的操作。

确保配置文件对系统上的其他用户不可见。它包含必须私有的访问令牌和 secret。Red Hat Single Sign-On 会自动创建 ~/.keycloak 目录及其内容，并具有适当的访问权限。如果目录已存在，Red Hat Single Sign-On 不会更新目录的权限。

可以避免将 secret 存储在配置文件中，但这样做并不方便，而且会增加令牌请求的数量。使用 --no-config 选项及所有命令，并指定配置 凭证 命令所需的身份验证信息，每个调用 kcadm。

18.5. 基本操作和资源 URI

Admin CLI 可以通过为简化特定任务的额外命令对 Admin REST API 端点通常执行 CRUD 操作。

此处列出了主要使用模式：

```
$ kcadm.sh create ENDPOINT [ARGUMENTS]
$ kcadm.sh get ENDPOINT [ARGUMENTS]
$ kcadm.sh update ENDPOINT [ARGUMENTS]
$ kcadm.sh delete ENDPOINT [ARGUMENTS]
```

创建、get、update 和 delete 命令分别映射到 HTTP 动词 POST、GET、PUT 和 DELETE。ENDPOINT 是一个目标资源 URI，可以是绝对（使用 http: 或 https:启动）或相对，Red Hat Single Sign-On 使用 用以下格式编写绝对 URL：

```
SERVER_URI/admin/realms/REALM/ENDPOINT
```

例如：如果您对服务器 http://localhost:8080/auth 进行身份验证，且域为 master，使用 ENDPOINT 的用户 将创建 http://localhost:8080/auth/admin/realms/master/users 资源 URL。

如果您将 ENDPOINT 设置为 客户端，则有效资源 URI 为 http://localhost:8080/auth/admin/realms/master/clients。

Red Hat Single Sign-On 有一个 域 端点，即 realms 的容器。它被解析为：

```
SERVER_URI/admin/realms
```

红帽单点登录具有 serverinfo 端点。此端点独立于 realm。

当您以具有 realm-admin 电源的用户身份进行验证时，您可能需要在多个域上执行命令。若是如此，请指定 -r 选项，告知 CLI 如何明确执行命令。命令不使用 kcadm.sh 配置凭证 的 --realm 选项 指定的 REALM，而是使用 TARGET_REALM。

```
SERVER_URI/admin/realms/TARGET_REALM/ENDPOINT
```

例如：

```
$ kcadm.sh config credentials --server http://localhost:8080/auth --realm master --user admin --
password admin
$ kcadm.sh create users -s username=testuser -s enabled=true -r demorealm
```

在本例中，您将启动一个以 admin 用户身份在 master realm 中验证的会话。然后，您可以针对资源 URL http://localhost:8080/auth/admin/realms/demorealm/users 执行 POST 调用。

创建或更新 命令会将 JSON 正文发送到服务器。您可以使用 -f FILENAME 从文件中读取预报文档。当您可以使用 -f - 选项时，Red Hat Single Sign-On 从标准输入中读取邮件正文。您可以指定单个属性及其值，如 创建用户示例中所示。Red Hat Single Sign-On 将属性组成 JSON 正文，并将其发送到服务器。

Red Hat Single Sign-On 中提供了几种方法，以使用 update 命令更新资源。您可以确定资源的当前状态并将其保存到文件中，编辑该文件并将其发送到服务器以进行更新。

例如：

```
$ kcadm.sh get realms/demorealm > demorealm.json
$ vi demorealm.json
$ kcadm.sh update realms/demorealm -f demorealm.json
```

此方法使用发送的 JSON 文档中的属性更新服务器上的资源。

另一种方法是通过使用 -s --set 选项设置 新值来执行持续更新。

例如：

```
$ kcadm.sh update realms/demorealm -s enabled=false
```

此方法将 enabled 属性设置为 false。

默认情况下，update 命令执行 get，然后将新属性值与现有的值合并。在某些情况下，端点可能支持 put 命令，但不支持 get 命令。您可以使用 -n 选项执行 no-merge 更新，这会在不先运行 get 命令的前提下执行 put 命令。

18.6. REALM 操作

创建新域

在 realms 端点上 使用 create 命令创建一个新的启用的域。将属性设置为 realm 并已启用。

```
$ kcadm.sh create realms -s realm=demorealm -s enabled=true
```

默认情况下，Red Hat Single Sign-On 会禁用域。您可以通过启用域来立即进行身份验证。

新对象的描述也可以是 JSON 格式。

```
$ kcadm.sh create realms -f demorealm.json
```

您可以使用文件直接向标准输入发送带有 realm 属性的 JSON 文档，或者将文档传送到标准输入。

例如：

- Linux：

```
$ kcadm.sh create realms -f - << EOF
{ "realm": "demorealm", "enabled": true }
EOF
```

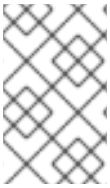
- Windows：

```
c:\> echo { "realm": "demorealm", "enabled": true } | kcadm create realms -f -
```

列出现有域

此命令返回所有域的列表。

```
$ kcadm.sh get realms
```



注意

Red Hat Single Sign-On 会过滤服务器中 realm 列表，以返回用户只能看到的域。

所有 realm 属性的列表可以是详细状态，大多数用户对属性子集感兴趣，如 realm 名称和 realm 的已启用状态。您可以使用 --fields 选项指定要返回的属性。

```
$ kcadm.sh get realms --fields realm,enabled
```

-

您可以以逗号分隔的值显示结果。

```
$ kcadm.sh get realms --fields realm --format csv --noquotes
```

获取特定域

将 realm 名称附加到集合 URI 以获取单个域。

```
$ kcadm.sh get realms/master
```

更新域

1. 当您不想更改所有域属性时，使用 **-s** 选项为属性设置新值。

   例如：

   ```
   $ kcadm.sh update realms/demorealm -s enabled=false
   ```

2. 如果要将所有可写属性设置为新值：

   a. 运行 **get** 命令。

   b. 编辑 **JSON** 文件中的当前值。

   c. 重新提交.

   例如：

   ```
   $ kcadm.sh get realms/demorealm > demorealm.json
   $ vi demorealm.json
   $ kcadm.sh update realms/demorealm -f demorealm.json
   ```

删除域

运行以下命令以删除域：

```
$ kcadm.sh delete realms/demorealm
```

打开域的所有登录页面选项

*将控制特定功能的属性设置为 **true**。*

例如：

```
$ kcadm.sh update realms/demorealm -s registrationAllowed=true -s
registrationEmailAsUsername=true -s rememberMe=true -s verifyEmail=true -s
resetPasswordAllowed=true -s editUsernameAllowed=true
```

列出域密钥

*在目标域的密钥 端点上 使用 **get** 操作。*

```
$ kcadm.sh get keys -r demorealm
```

生成新域密钥

1. *在添加新的 **RSA** 生成的密钥对前，获取目标域的 **ID**。*

   例如：

   ```
   $ kcadm.sh get realms/demorealm --fields id --format csv --noquotes
   ```

2. *与 **kcadm.sh** 获取键 **-r demorealm** 所发现的那样，添加优先级高于现有提供程序的新密钥提供程序。*

   例如：

   - ***Linux：***

     ```
     $ kcadm.sh create components -r demorealm -s name=rsa-generated -s providerId=rsa-
     generated -s providerType=org.keycloak.keys.KeyProvider -s parentId=959844c1-d149-
     41d7-8359-6aa527fca0b0 -s 'config.priority=["101"]' -s 'config.enabled=["true"]' -s
     'config.active=["true"]' -s 'config.keySize=["2048"]'
     ```

   - ***Windows：***

     ```
     c:\> kcadm create components -r demorealm -s name=rsa-generated -s providerId=rsa-
     generated -s providerType=org.keycloak.keys.KeyProvider -s parentId=959844c1-d149-
     ```

```
41d7-8359-6aa527fca0b0 -s "config.priority=[\"101\"]" -s "config.enabled=[\"true\"]" -s
"config.active=[\"true\"]" -s "config.keySize=[\"2048\"]"
```

3. 将 parentId 属性设置为目标域 ID 的值。

现在，新添加的密钥是活动密钥，如 kcadm.sh get keys -r demorealm 所示。

从 Java 密钥存储文件添加新域密钥

1. 添加新密钥提供程序，以添加新密钥对预先准备为 JKS 文件。

例如，在：

- Linux：

```
$ kcadm.sh create components -r demorealm -s name=java-keystore -s providerId=java-
keystore -s providerType=org.keycloak.keys.KeyProvider -s parentId=959844c1-d149-
41d7-8359-6aa527fca0b0 -s 'config.priority=["101"]' -s 'config.enabled=["true"]' -s
'config.active=["true"]' -s 'config.keystore=["/opt/keycloak/keystore.jks"]' -s
'config.keystorePassword=["secret"]' -s 'config.keyPassword=["secret"]' -s
'config.keyAlias=["localhost"]'
```

- Windows：

```
c:\> kcadm create components -r demorealm -s name=java-keystore -s providerId=java-
keystore -s providerType=org.keycloak.keys.KeyProvider -s parentId=959844c1-d149-
41d7-8359-6aa527fca0b0 -s "config.priority=[\"101\"]" -s "config.enabled=[\"true\"]" -s
"config.active=[\"true\"]" -s "config.keystore=[\"/opt/keycloak/keystore.jks\"]" -s
"config.keystorePassword=[\"secret\"]" -s "config.keyPassword=[\"secret\"]" -s
"config.keyAlias=[\"localhost\"]"
```

2. 确保更改 密钥存储 的属性值、密钥存储密码、密钥 密码和 别名 以匹配特定的密钥存储。

3. 将 parentId 属性设置为目标域 ID 的值。

使键被动或禁用密钥

1. 确定您要进行被动的键。

```
$ kcadm.sh get keys -r demorealm
```

2. 使用键的 providerId 属性构造端点 URI，如 components/PROVIDER_ID。

3. 执行 更新。

   例如：

   - Linux：

     ```
     $ kcadm.sh update components/PROVIDER_ID -r demorealm -s 'config.active=["false"]'
     ```

   - Windows：

     ```
     c:\> kcadm update components/PROVIDER_ID -r demorealm -s "config.active=[\"false\"]"
     ```

     您可以更新其他关键属性：

4. 设置一个一个新的 enabled 值来禁用这个键，如 config.enabled=["false"]。

5. 设置一个新的 优先级值，以更改键的优先级，如 config.priority=["110"]。

删除旧密钥

1. 请确定您删除的操作是不活跃的，并禁用了它。这个操作是防止应用程序和用户丢失的现有令牌失败。

2. 确定要删除的密钥。

   ```
   $ kcadm.sh get keys -r demorealm
   ```

3. 使用密钥的 providerId 来执行删除。

   ```
   $ kcadm.sh delete components/PROVIDER_ID -r demorealm
   ```

-

为域配置事件日志

对 events/config 端点使用 update 命令。

eventsListeners 属性包含 EventListenerProviderFactory ID 列表，指定接收事件的所有事件监听程序。属性可用于控制内置事件存储，因此您可以使用 Admin REST API 查询过去的事件。Red Hat Single Sign-On 对服务调用(启用的事件)的日志记录和管理控制台或 Admin REST API 触发的审计事件具有独立的控制(adminEventsEnabled)。您可以将 eventsExpiration 事件设置为过期，以防止您的数据库填满。Red Hat Single Sign-On 将 eventsExpiration 设置为生存时间（以秒表示）。

您可以设置一个内置事件监听程序，通过 JBoss-logging 接收所有事件和记录事件。使用 org.keycloak.events 日志记录器，红帽单点登录日志错误事件作为 WARN，其他事件则作为 DEBUG。

例如：

- Linux：

```
$ kcadm.sh update events/config -r demorealm -s 'eventsListeners=["jboss-logging"]'
```

- Windows：

```
c:\> kcadm update events/config -r demorealm -s "eventsListeners=[\"jboss-logging\"]"
```

例如：

您可以为所有可用 ERROR 事件（不包括审计事件）打开存储，以便您可以在管理员 REST 检索事件。

- Linux：

```
$ kcadm.sh update events/config -r demorealm -s eventsEnabled=true -s 'enabledEventTypes=
["LOGIN_ERROR","REGISTER_ERROR","LOGOUT_ERROR","CODE_TO_TOKEN_ERROR","CLIE
NT_LOGIN_ERROR","FEDERATED_IDENTITY_LINK_ERROR","REMOVE_FEDERATED_IDENTIT
Y_ERROR","UPDATE_EMAIL_ERROR","UPDATE_PROFILE_ERROR","UPDATE_PASSWORD_ER
ROR","UPDATE_TOTP_ERROR","VERIFY_EMAIL_ERROR","REMOVE_TOTP_ERROR","SEND_V
ERIFY_EMAIL_ERROR","SEND_RESET_PASSWORD_ERROR","SEND_IDENTITY_PROVIDER_LI
NK_ERROR","RESET_PASSWORD_ERROR","IDENTITY_PROVIDER_FIRST_LOGIN_ERROR","I
```

```
DENTITY_PROVIDER_POST_LOGIN_ERROR","CUSTOM_REQUIRED_ACTION_ERROR","EXEC
UTE_ACTIONS_ERROR","CLIENT_REGISTER_ERROR","CLIENT_UPDATE_ERROR","CLIENT_D
ELETE_ERROR"]' -s eventsExpiration=172800
```

- **Windows：**

```
c:\> kcadm update events/config -r demorealm -s eventsEnabled=true -s "enabledEventTypes=
[\"LOGIN_ERROR\",\"REGISTER_ERROR\",\"LOGOUT_ERROR\",\"CODE_TO_TOKEN_ERROR\",\
"CLIENT_LOGIN_ERROR\",\"FEDERATED_IDENTITY_LINK_ERROR\",\"REMOVE_FEDERATED_I
DENTITY_ERROR\",\"UPDATE_EMAIL_ERROR\",\"UPDATE_PROFILE_ERROR\",\"UPDATE_PASS
WORD_ERROR\",\"UPDATE_TOTP_ERROR\",\"VERIFY_EMAIL_ERROR\",\"REMOVE_TOTP_ERR
OR\",\"SEND_VERIFY_EMAIL_ERROR\",\"SEND_RESET_PASSWORD_ERROR\",\"SEND_IDENTIT
Y_PROVIDER_LINK_ERROR\",\"RESET_PASSWORD_ERROR\",\"IDENTITY_PROVIDER_FIRST_
LOGIN_ERROR\",\"IDENTITY_PROVIDER_POST_LOGIN_ERROR\",\"CUSTOM_REQUIRED_ACTI
ON_ERROR\",\"EXECUTE_ACTIONS_ERROR\",\"CLIENT_REGISTER_ERROR\",\"CLIENT_UPDA
TE_ERROR\",\"CLIENT_DELETE_ERROR\"]" -s eventsExpiration=172800
```

您可以将存储的事件类型重置为 所有可用事件类型。将值设为空列表与迭代所有相同。

```
$ kcadm.sh update events/config -r demorealm -s enabledEventTypes=[]
```

您可以启用审计事件的存储。

```
$ kcadm.sh update events/config -r demorealm -s adminEventsEnabled=true -s
adminEventsDetailsEnabled=true
```

您可以得到最后 100 个事件。事件从最新到最旧的排序。

```
$ kcadm.sh get events --offset 0 --limit 100
```

您可以删除所有已保存的事件。

```
$ kcadm delete events
```

刷新缓存

1. **使用 create 命令和其中一个端点来清除缓存：**

- **clear-realm-cache**

- ***clear-user-cache***
- ***clear-keys-cache***

2. ***将 realm 设置为与目标域相同的值。***

   例如：

```
$ kcadm.sh create clear-realm-cache -r demorealm -s realm=demorealm
$ kcadm.sh create clear-user-cache -r demorealm -s realm=demorealm
$ kcadm.sh create clear-keys-cache -r demorealm -s realm=demorealm
```

从导出的 .json 文件中导入一个域

1. ***在 partialImport 端点上使用 create 命令。***

2. ***将 ifResourceExists 设置为 FAIL、SKIP 或 OVERWRITE。***

3. ***使用 -f 提交导出的域 .json 文件。***

   例如：

```
$ kcadm.sh create partialImport -r demorealm2 -s ifResourceExists=FAIL -o -f
demorealm.json
```

   如果 realm 尚不存在，请首先创建它。

   例如：

```
$ kcadm.sh create realms -s realm=demorealm2 -s enabled=true
```

18.7. 角色操作

创建域角色

使用角色端点创建 realm 角色。

```
$ kcadm.sh create roles -r demorealm -s name=user -s 'description=Regular user with a limited set of permissions'
```

创建客户端角色

1. 识别客户端。

2. 使用 get 命令列出可用的客户端。

```
$ kcadm.sh get clients -r demorealm --fields id,clientId
```

3. 使用 clientId 属性构建端点 URI （如 client /ID/roles ）来创建新角色。

例如：

```
$ kcadm.sh create clients/a95b6af3-0bdc-4878-ae2e-6d61a4eca9a0/roles -r demorealm -s name=editor -s 'description=Editor can edit, and publish any article'
```

列出域角色

在 roles 端点上使用 get 命令列出现有的域角色。

```
$ kcadm.sh get roles -r demorealm
```

您还可以使用 get-roles 命令。

```
$ kcadm.sh get-roles -r demorealm
```

列出客户端角色

红帽单点登录具有一个专用的 get-roles 命令，可以简化域和客户端角色的列表。命令是对 get 命令的一个扩展，其行为与 get 命令相同，但还有额外的语义列表角色。

通过传递 clientId (--cclientid)选项或 id (--c id)选项，使用 get-roles 命令来识别客户端以列出客户端角色。

例如：

```
$ kcadm.sh get-roles -r demorealm --cclientid realm-management
```

获取特定域角色

使用 get 命令和角色名称，为特定域角色(roles/ROLE_NAME)构建端点 URI，其中 user 是现有角色的名称。

例如：

```
$ kcadm.sh get roles/user -r demorealm
```

您可以使用 get-roles 命令，将它传递角色名称(--rolename 选项)或 ID (--roleid 选项)。

例如：

```
$ kcadm.sh get-roles -r demorealm --rolename user
```

获取特定的客户端角色

使用 get-roles 命令，将它传递 clientId 属性(--cclientid 选项)或 ID 属性(--cid 选项)来识别客户端，并传递角色名称(--rolename 选项)或角色 ID 属性(--roleid)来识别特定的客户端角色。

例如：

```
$ kcadm.sh get-roles -r demorealm --cclientid realm-management --rolename manage-clients
```

更新域角色

使用带有您用来获取特定域角色的端点 URI 的 update 命令。

例如：

```
$ kcadm.sh update roles/user -r demorealm -s 'description=Role representing a regular user'
```

更新客户端角色

使用 update 命令以及您用来获取特定客户端角色的端点 URI。

例如：

```
$ kcadm.sh update clients/a95b6af3-0bdc-4878-ae2e-6d61a4eca9a0/roles/editor -r demorealm -s
'description=User that can edit, and publish articles'
```

删除域角色

使用 delete 命令以及您用来获取特定域角色的端点 URI。

例如：

```
$ kcadm.sh delete roles/user -r demorealm
```

删除客户端角色

使用 delete 命令以及您用来获取特定客户端角色的端点 URI。

例如：

```
$ kcadm.sh delete clients/a95b6af3-0bdc-4878-ae2e-6d61a4eca9a0/roles/editor -r demorealm
```

为复合角色列出分配的、可用和有效的域角色

使用 get-roles 命令，列出为复合角色分配的、可用且有效的域角色。

1. 要列出复合角色分配的域角色，按名称(--rname 选项)或 ID (--rid 选项)指定目标复合角色。

   例如：

   ```
   $ kcadm.sh get-roles -r demorealm --rname testrole
   ```

2. 使用 --effective 选项列出 有效的 域角色。

   例如：

```
$ kcadm.sh get-roles -r demorealm --rname testrole --effective
```

3. 使用 **--available** 选项列出您可以添加到复合角色的 **realm** 角色。

   例如：

```
$ kcadm.sh get-roles -r demorealm --rname testrole --available
```

为复合角色列出分配的、可用和有效的客户端角色

使用 **get-roles** 命令，列出为复合角色分配的、可用且有效的客户端角色。

1. 要列出为复合角色分配的客户端角色，您可以按名称(**--rname** 选项)或 **ID** (**--rid** 选项)指定目标复合角色(**--cclientid** 选项)或 **ID** (**--cid** 选项)。

   例如：

```
$ kcadm.sh get-roles -r demorealm --rname testrole --cclientid realm-management
```

2. 使用 **--effective** 选项列出 有效的 域角色。

   例如：

```
$ kcadm.sh get-roles -r demorealm --rname testrole --cclientid realm-management --effective
```

3. 使用 **--available** 选项列出您可以在目标复合角色中添加的 **realm** 角色。

   例如：

```
$ kcadm.sh get-roles -r demorealm --rname testrole --cclientid realm-management --
available
```

将 realm 角色添加到复合角色

Red Hat Single Sign-On 提供了一个 **add-roles** 命令，用于添加域角色和客户端角色。

本例 将用户 角色添加到复合角色 **testrole** 中。

```
$ kcadm.sh add-roles --rname testrole --rolename user -r demorealm
```

从复合角色中删除域角色

Red Hat Single Sign-On 提供了一个 remove-roles 命令，用于删除域角色和客户端角色。

下例从目标复合角色 testrole 中删除了用户角色。

```
$ kcadm.sh remove-roles --rname testrole --rolename user -r demorealm
```

将客户端角色添加到域角色

Red Hat Single Sign-On 提供了一个 add-roles 命令，用于添加域角色和客户端角色。

以下示例将客户端 realm-management、create-client 和 view-users 中定义的角色添加到 testrole 复合角色。

```
$ kcadm.sh add-roles -r demorealm --rname testrole --cclientid realm-management --rolename
create-client --rolename view-users
```

将客户端角色添加到客户端角色

1. **使用 get-roles 命令，确定复合客户端角色的 ID。**

   例如：

   ```
   $ kcadm.sh get-roles -r demorealm --cclientid test-client --rolename operations
   ```

2. **假定客户端存在具有名为 test-client 的 clientId 属性、名为 support 的客户端角色，以及名为 operations 的客户端角色，它成为一个复合角色，其 ID 为 "fc400897-ef6a-4e8c-872b-1581b7fa8a71"。**

3. **使用以下示例将另一个角色添加到复合角色。**

   ```
   $ kcadm.sh add-roles -r demorealm --cclientid test-client --rid fc400897-ef6a-4e8c-872b-
   1581b7fa8a71 --rolename support
   ```

4. **使用 get-roles --all 命令列出复合角色的角色。**

例如：

```
$ kcadm.sh get-roles --rid fc400897-ef6a-4e8c-872b-1581b7fa8a71 --all
```

从复合角色删除客户端角色

使用 **remove-roles** 命令，从复合角色中删除客户端角色。

使用以下示例，从 **testrole** 复合角色中删除 **client realm-management**、**create-client** 角色和 **view-users** 角色中定义的两个角色。

```
$ kcadm.sh remove-roles -r demorealm --rname testrole --cclientid realm-management --rolename
create-client --rolename view-users
```

将客户端角色添加到组

使用 **add-roles** 命令添加 **realm** 角色和客户端角色。

以下示例将客户端 **realm-management**、**create-client** 和 **view-users** 中定义的角色添加到组组(**--gname** 选项)。或者，您也可以按 **ID** 指定组(**--gid** 选项)。

如需更多信息，请参阅组操作。

```
$ kcadm.sh add-roles -r demorealm --gname Group --cclientid realm-management --rolename
create-client --rolename view-users
```

从组中删除客户端角色

使用 **remove-roles** 命令，从组中删除客户端角色。

以下示例从 **Group** 组移除了客户端域管理 中定义的两个角色：**create-client** 和 **view-users**。

如需更多信息，请参阅组操作。

```
$ kcadm.sh remove-roles -r demorealm --gname Group --cclientid realm-management --rolename
create-client --rolename view-users
```

18.8. 客户端操作

创建客户端

1. *在客户端端点上运行 create 命令，以创建新客户端。*

   例如：

   ```
   $ kcadm.sh create clients -r demorealm -s clientId=myapp -s enabled=true
   ```

2. *如果为适配器设置 secret 进行身份验证，请指定 secret。*

   例如：

   ```
   $ kcadm.sh create clients -r demorealm -s clientId=myapp -s enabled=true -s
   clientAuthenticatorType=client-secret -s secret=d0b8122f-8dfb-46b7-b68a-f5cc4e25d000
   ```

列出客户端

在客户端端点上使用 get 命令来列出客户端。

本例过滤输出以仅列出 id 和 clientId 属性：

```
$ kcadm.sh get clients -r demorealm --fields id,clientId
```

获取特定客户端

使用客户端 ID 来构建以特定 客户端（如 client/ID ）为目标的端点 URI。

例如：

```
$ kcadm.sh get clients/c7b8547f-e748-4333-95d0-410b76b3f4a3 -r demorealm
```

获取特定客户端的当前 secret

使用客户端 ID 来构造端点 URI，如 client /ID/client-secret。

例如：

```
$ kcadm.sh get clients/$CID/client-secret
```

为特定客户端生成新 secret

使用客户端 ID 来构造端点 URI，如 client /ID/client-secret。

例如：

```
$ kcadm.sh create clients/$CID/client-secret
```

更新特定客户端的当前 secret

使用客户端 ID 来构造端点 URI，如 client /ID。

例如：

```
$ kcadm.sh update clients/$CID -s "secret=newSecret"
```

为特定客户端获取适配器配置文件(keycloak.json)

使用客户端 ID 构造以特定 客户端为目标的端点 URI，如 client/ID/installation/providers/keycloak-oidc-keycloak-json。

例如：

```
$ kcadm.sh get clients/c7b8547f-e748-4333-95d0-410b76b3f4a3/installation/providers/keycloak-oidc-keycloak-json -r demorealm
```

获取特定客户端的 WildFly 子系统适配器配置

使用客户端 ID 构造以特定 客户端为目标的端点 URI，如 client/ID/installation/providers/keycloak-oidc-jboss-subsystem。

例如：

```
$ kcadm.sh get clients/c7b8547f-e748-4333-95d0-410b76b3f4a3/installation/providers/keycloak-oidc-jboss-subsystem -r demorealm
```

获取特定客户端的 Docker-v2 示例配置

使用客户端 ID 来构建以特定 客户端为目标的端点 URI，如 client/ID/installation/providers/docker-v2-compose-yaml。

响应采用 .zip 格式。

例如：

```
$ kcadm.sh get http://localhost:8080/auth/admin/realms/demorealm/clients/8f271c35-44e3-446f-8953-b0893810ebe7/installation/providers/docker-v2-compose-yaml -r demorealm > keycloak-docker-compose-yaml.zip
```

更新客户端

使用您用来获取特定客户端的同一端点 URI 的 update 命令。

例如：

- *Linux：*

```
$ kcadm.sh update clients/c7b8547f-e748-4333-95d0-410b76b3f4a3 -r demorealm -s enabled=false -s publicClient=true -s 'redirectUris=["http://localhost:8080/myapp/*"]' -s baseUrl=http://localhost:8080/myapp -s adminUrl=http://localhost:8080/myapp
```

- *Windows：*

```
c:\> kcadm update clients/c7b8547f-e748-4333-95d0-410b76b3f4a3 -r demorealm -s enabled=false -s publicClient=true -s "redirectUris=[\"http://localhost:8080/myapp/*\"]" -s baseUrl=http://localhost:8080/myapp -s adminUrl=http://localhost:8080/myapp
```

删除客户端

使用您用来获取特定客户端的同一端点 URI 的 delete 命令。

例如：

```
$ kcadm.sh delete clients/c7b8547f-e748-4333-95d0-410b76b3f4a3 -r demorealm
```

为客户端服务帐户添加或删除角色

客户端的服务帐户是使用用户名 service-account-CLIENT_ID 的用户帐户。您可以对此帐户执行与常规帐户相同的用户操作。

18.9. 用户操作

创建用户

在用户 端点上运行 create 命令，以创建新用户。

例如：

```
$ kcadm.sh create users -r demorealm -s username=testuser -s enabled=true
```

列出用户

使用用户端点 列出用户。目标用户下次登录时必须更改其密码。

例如：

```
$ kcadm.sh get users -r demorealm --offset 0 --limit 1000
```

您可以根据 用户名、firstName、lastName 或 email 来过滤用户。

例如：

```
$ kcadm.sh get users -r demorealm -q email=google.com
$ kcadm.sh get users -r demorealm -q username=testuser
```

注意

过滤不使用完全匹配。这个示例根据 *testuser* 模式匹配 username 属性的值。

您可以通过指定多个 -q 选项来过滤多个属性。Red Hat Single Sign-On 返回符合所有属性条件的用户。

获取特定用户

使用用户 ID 来编写端点 URI，如 用户/USER_ID。

例如：

```
$ kcadm.sh get users/0ba7a3fd-6fd8-48cd-a60b-2e8fd82d56e2 -r demorealm
```

更新用户

使用您用来获得特定用户的同一端点 URI 的 update 命令。

例如：

- *Linux：*

```
$ kcadm.sh update users/0ba7a3fd-6fd8-48cd-a60b-2e8fd82d56e2 -r demorealm -s
'requiredActions=
["VERIFY_EMAIL","UPDATE_PROFILE","CONFIGURE_TOTP","UPDATE_PASSWORD"]'
```

- *Windows：*

```
c:\> kcadm update users/0ba7a3fd-6fd8-48cd-a60b-2e8fd82d56e2 -r demorealm -s "requiredActions=
[\"VERIFY_EMAIL\",\"UPDATE_PROFILE\",\"CONFIGURE_TOTP\",\"UPDATE_PASSWORD\"]"
```

删除用户

使用您用来获得特定用户的相同端点 URI 的 delete 命令。

例如：

```
$ kcadm.sh delete users/0ba7a3fd-6fd8-48cd-a60b-2e8fd82d56e2 -r demorealm
```

重置用户的密码

使用专用的 set-password 命令重置用户的密码。

例如：

```
$ kcadm.sh set-password -r demorealm --username testuser --new-password NEWPASSWORD --
temporary
```

这个命令为用户设置临时密码。目标用户下次登录时必须更改密码。

您可以使用 --userid 使用 id 属性指定用户。

您可以使用由您用来获取特定用户（如 用户/USER_ID/reset-password ）的端点上的 update 命令获得相同的结果。

例如：

```
$ kcadm.sh update users/0ba7a3fd-6fd8-48cd-a60b-2e8fd82d56e2/reset-password -r demorealm -s
type=password -s value=NEWPASSWORD -s temporary=true -n
```

n 参数确保 Red Hat Single Sign-On 执行 PUT 命令，而不在 PUT 命令前面执行 GET 命令。这是必要的，因为 reset-password 端点不支持 GET。

列出分配给用户的、可用和有效的域角色

您可以使用 get-roles 命令列出用户的已分配、可用和有效的域角色。

1. 根据用户名或 ID 指定目标用户，以列出用户的 分配 域角色。

   例如：

   ```
   $ kcadm.sh get-roles -r demorealm --uusername testuser
   ```

2. 使用 --effective 选项列出 有效的 域角色。

   例如：

   ```
   $ kcadm.sh get-roles -r demorealm --uusername testuser --effective
   ```

3. 使用 --available 选项列出您可以添加到用户的 realm 角色。

   例如：

   -

```
$ kcadm.sh get-roles -r demorealm --uusername testuser --available
```

列出分配给用户的、可用和有效的客户端角色

使用 **get-roles** 命令，列出用户的已分配、可用且有效的客户端角色。

1. 根据用户名(**--uusername** 选项)或 **ID** (**--uid** 选项)和客户端指定目标用户，使用 **clientId** 属性 (**--cclientid** 选项)或 **ID** (**--cid** 选项)来列出 已为用户分配 的客户端角色。

   例如：

   ```
   $ kcadm.sh get-roles -r demorealm --uusername testuser --cclientid realm-management
   ```

2. 使用 **--effective** 选项列出 有效的 域角色。

   例如：

   ```
   $ kcadm.sh get-roles -r demorealm --uusername testuser --cclientid realm-management --effective
   ```

3. 使用 **--available** 选项列出您可以添加到用户的 **realm** 角色。

   例如：

   ```
   $ kcadm.sh get-roles -r demorealm --uusername testuser --cclientid realm-management --available
   ```

将 realm 角色添加到用户

使用 **add-roles** 命令，将 **realm** 角色添加到用户。

使用以下示例将用户角色添加到用户 **testuser** 中：

```
$ kcadm.sh add-roles --uusername testuser --rolename user -r demorealm
```

从用户中删除域角色

使用 **remove-roles** 命令，从用户中删除域角色。

使用以下示例从用户 testuser 中删除用户角色：

```
$ kcadm.sh remove-roles --uusername testuser --rolename user -r demorealm
```

将客户端角色添加到用户

使用 add-roles 命令，向用户添加客户端角色。

使用以下示例，将客户端域管理 中定义的两个角色(create-client 角色和 view-users 角色)添加到用户 testuser 中。

```
$ kcadm.sh add-roles -r demorealm --uusername testuser --cclientid realm-management --rolename
create-client --rolename view-users
```

从用户中删除客户端角色

使用 remove-roles 命令，从用户中删除客户端角色。

使用以下示例删除域管理客户端中定义的两个角色：

```
$ kcadm.sh remove-roles -r demorealm --uusername testuser --cclientid realm-management --
rolename create-client --rolename view-users
```

列出用户会话

1. 确定用户的 ID、
2. 使用 ID 来编写端点 URI，如 users/ID/sessions。
3. 使用 get 命令检索用户会话的列表。

   例如：

   ```
   $kcadm get users/6da5ab89-3397-4205-afaa-e201ff638f9e/sessions
   ```

从特定会话中注销用户

1. 如前面所述确定会话的 ID。

2. 使用会话的 ID 来编写端点 URI，如 sessions/ID。

3. 使用 delete 命令使会话无效。

例如：

```
$ kcadm.sh delete sessions/d0eaa7cc-8c5d-489d-811a-69d3c4ec84d1
```

从所有会话注销用户

使用用户的 ID 来构造端点 URI，如 用户/ID/注销。

使用 create 命令在端点 URI 上执行 POST。

例如：

```
$ kcadm.sh create users/6da5ab89-3397-4205-afaa-e201ff638f9e/logout -r demorealm -s
realm=demorealm -s user=6da5ab89-3397-4205-afaa-e201ff638f9e
```

18.10. 组操作

创建组

对 组 端点使用 create 命令创建新组。

例如：

```
$ kcadm.sh create groups -r demorealm -s name=Group
```

列出组

对 组 端点使用 get 命令列出组。

例如：

```
$ kcadm.sh get groups -r demorealm
```

获取特定组

使用组的 ID 来构造端点 URI，如 groups/GROUP_ID。

例如：

```
$ kcadm.sh get groups/51204821-0580-46db-8f2d-27106c6b5ded -r demorealm
```

更新组

使用您用来获取特定组的相同端点 URI 的 update 命令。

例如：

```
$ kcadm.sh update groups/51204821-0580-46db-8f2d-27106c6b5ded -s 'attributes.email=
["group@example.com"]' -r demorealm
```

删除组

使用您用来获取特定组的相同端点 URI 的 delete 命令。

例如：

```
$ kcadm.sh delete groups/51204821-0580-46db-8f2d-27106c6b5ded -r demorealm
```

创建子组

通过列出组来查找父组的 ID。使用该 ID 构建端点 URI，如 groups/GROUP_ID/children。

例如：

```
$ kcadm.sh create groups/51204821-0580-46db-8f2d-27106c6b5ded/children -r demorealm -s
name=SubGroup
```

在另一个组下移动组

1. *查找现有父组和现有子组的 ID。*

2. 使用父组的 ID 来构造端点 URI，如 groups/PARENT_GROUP_ID/children。

3. 在此端点上运行 create 命令，并将子组的 ID 作为参数传递。

例如：

```
$ kcadm.sh create groups/51204821-0580-46db-8f2d-27106c6b5ded/children -r demorealm -s
id=08d410c6-d585-4059-bb07-54dcb92c5094 -s name=SubGroup
```

获取特定用户的组

使用用户的 ID 来确定组中的用户成员资格来编写端点 URI，如 用户/USER_ID/groups。

例如：

```
$ kcadm.sh get users/b544f379-5fc4-49e5-8a8d-5cfb71f46f53/groups -r demorealm
```

将用户添加到组中

使用 update 命令以及由用户 ID 和组 ID （如 用户/USER_ID/groups/GROUP_ID ）组成的端点 URI 将用户添加到组中。

例如：

```
$ kcadm.sh update users/b544f379-5fc4-49e5-8a8d-5cfb71f46f53/groups/ce01117a-7426-4670-
a29a-5c118056fe20 -r demorealm -s realm=demorealm -s userId=b544f379-5fc4-49e5-8a8d-
5cfb71f46f53 -s groupId=ce01117a-7426-4670-a29a-5c118056fe20 -n
```

从组中删除用户

对用于添加用户的端点 URI （如 用户/USER_ID/groups/GROUP_ID ）上使用 delete 命令，从组中删除用户。

例如：

```
$ kcadm.sh delete users/b544f379-5fc4-49e5-8a8d-5cfb71f46f53/groups/ce01117a-7426-4670-a29a-
5c118056fe20 -r demorealm
```

列出组分配的、可用和有效的域角色

使用专用的 get-roles 命令，列出组的已分配、可用且有效的域角色。

1. 按名称指定目标组(--gname 选项)、路径(--gpath 选项)或 ID (--gid 选项)来列出组 分配的 realm 角色。

   例如：

   ```
   $ kcadm.sh get-roles -r demorealm --gname Group
   ```

2. 使用 --effective 选项列出 有效的 域角色。

   例如：

   ```
   $ kcadm.sh get-roles -r demorealm --gname Group --effective
   ```

3. 使用 --available 选项列出您可以添加到组中的 realm 角色。

   例如：

   ```
   $ kcadm.sh get-roles -r demorealm --gname Group --available
   ```

列出组分配的、可用和有效的客户端角色

使用 get-roles 命令，列出分配给组的已分配、可用且有效的客户端角色。

1. 按名称指定目标组(--gname 选项)或 ID (--gid 选项)。

2. 通过 clientId 属性(--cclientid 选项)或 ID (--id 选项)指定客户端，以列出 已为用户分配 的客户端角色。

   例如：

   ```
   $ kcadm.sh get-roles -r demorealm --gname Group --cclientid realm-management
   ```

3. 使用 --effective 选项列出 有效的 域角色。

例如：

```
$ kcadm.sh get-roles -r demorealm --gname Group --cclientid realm-management --effective
```

4. *使用 --available 选项列出您仍然可以添加到组中的域角色。*

   例如：

```
$ kcadm.sh get-roles -r demorealm --gname Group --cclientid realm-management --available
```

18.11. 身份提供程序操作

列出可用的身份提供程序

使用 serverinfo 端点列出可用的身份提供程序。

例如：

```
$ kcadm.sh get serverinfo -r demorealm --fields 'identityProviders(*)'
```

注意

红帽单点登录处理 serverinfo 端点与 realms 端点类似。Red Hat Single Sign-On 无法解析相对于目标域的端点，因为它存在于任何特定域之外。

列出配置的身份提供程序

使用 identity-provider/instances 端点。

例如：

```
$ kcadm.sh get identity-provider/instances -r demorealm --fields alias,providerId,enabled
```

获取特定配置的身份提供程序

使用身份提供程序的别名属性构造端点 URI，如 identity-provider/instances/ALIAS 来获取特定的身份提供程序。

例如：

```
$ kcadm.sh get identity-provider/instances/facebook -r demorealm
```

删除特定配置的身份提供程序

使用 delete 命令以及您用来获取特定配置的身份提供程序相同的端点 URI。

例如：

```
$ kcadm.sh delete identity-provider/instances/facebook -r demorealm
```

配置 Keycloak OpenID Connect 身份提供程序

1. 在创建新身份提供程序实例时，使用 keycloak-oidc 作为 providerId。

2. 提供配置属性： authorizationUrl、tokenUrl、clientId 和 clientSecret。

   例如：

```
$ kcadm.sh create identity-provider/instances -r demorealm -s alias=keycloak-oidc -s
providerId=keycloak-oidc -s enabled=true -s 'config.useJwksUrl="true"' -s
config.authorizationUrl=http://localhost:8180/auth/realms/demorealm/protocol/openid-
connect/auth -s
config.tokenUrl=http://localhost:8180/auth/realms/demorealm/protocol/openid-connect/token -
s config.clientId=demo-oidc-provider -s config.clientSecret=secret
```

配置 OpenID Connect 身份提供程序

配置通用 OpenID Connect 供应商的方式与配置 Keycloak OpenID Connect 供应商相同，但您需要将 providerId 属性值设置为 oidc。

配置 SAML 2 身份提供程序

1. 使用 saml 作为 providerId。

2. 提供配置属性： singleSignOnServiceUrl、nameIDPolicyFormat，以及 signatureAlgorithm。

例如：

```
$ kcadm.sh create identity-provider/instances -r demorealm -s alias=saml -s providerId=saml -s
enabled=true -s 'config.useJwksUrl="true"' -s
config.singleSignOnServiceUrl=http://localhost:8180/auth/realms/saml-broker-realm/protocol/saml -s
config.nameIDPolicyFormat=urn:oasis:names:tc:SAML:2.0:nameid-format:persistent -s
config.signatureAlgorithm=RSA_SHA256
```

配置 Facebook 身份提供程序

1. *使用 facebook 作为 providerId。*

2. *提供 config attributes: clientId 和 clientSecret。您可以在 Facebook Developers 应用程序配置页面中找到这些属性。如需更多信息，请参阅 facebook identity broker 页面。*

   例如：

   ```
   $ kcadm.sh create identity-provider/instances -r demorealm -s alias=facebook -s
   providerId=facebook -s enabled=true  -s 'config.useJwksUrl="true"' -s
   config.clientId=FACEBOOK_CLIENT_ID -s
   config.clientSecret=FACEBOOK_CLIENT_SECRET
   ```

配置 Google 身份提供程序

1. *使用 google 作为 providerId。*

2. *提供 config attributes: clientId 和 clientSecret。您可以在应用程序的 Google Developers 应用程序配置页面中找到这些属性。如需更多信息，请参阅 Google identity broker 页面。*

   例如：

   ```
   $ kcadm.sh create identity-provider/instances -r demorealm -s alias=google -s
   providerId=google -s enabled=true  -s 'config.useJwksUrl="true"' -s
   config.clientId=GOOGLE_CLIENT_ID -s config.clientSecret=GOOGLE_CLIENT_SECRET
   ```

配置 Twitter 身份提供程序

1. *使用 twitter 作为 providerId。*

2. *提供配置属性 client Id 和 clientSecret。您可以在应用程序的 Twitter 应用程序配置页面中*

找到这些属性。如需更多信息，请参阅 sVirt 身份代理 页面。

例如：

```
$ kcadm.sh create identity-provider/instances -r demorealm -s alias=google -s
providerId=google -s enabled=true  -s 'config.useJwksUrl="true"' -s
config.clientId=TWITTER_API_KEY -s config.clientSecret=TWITTER_API_SECRET
```

配置 GitHub 身份提供程序

1. 使用 github 作为 providerId。

2. 提供配置属性 client Id 和 clientSecret。您可以在应用程序的 GitHub Developer Application Settings 页面中找到这些属性。如需更多信息，请参阅 Github 身份代理 页面。

例如：

```
$ kcadm.sh create identity-provider/instances -r demorealm -s alias=github -s
providerId=github -s enabled=true  -s 'config.useJwksUrl="true"' -s
config.clientId=GITHUB_CLIENT_ID -s config.clientSecret=GITHUB_CLIENT_SECRET
```

配置 LinkedIn 身份提供程序

1. 使用 链接 作为 providerId。

2. 提供配置属性 client Id 和 clientSecret。您可以在应用程序的 LinkedIn Developer Console 应用程序页面中找到这些属性。如需更多信息，请参阅 LinkedIn 身份代理 页面。

例如：

```
$ kcadm.sh create identity-provider/instances -r demorealm -s alias=linkedin -s
providerId=linkedin -s enabled=true  -s 'config.useJwksUrl="true"' -s
config.clientId=LINKEDIN_CLIENT_ID -s config.clientSecret=LINKEDIN_CLIENT_SECRET
```

配置 Microsoft Live 身份提供程序

1. 使用 microsoft 作为 providerId。

2. 提供配置属性 client Id 和 clientSecret。您可以在应用程序的 Microsoft Application Registration Portal 页面中找到这些属性。如需更多信息，请参阅 Microsoft 身份代理 页面。

例如：

```
$ kcadm.sh create identity-provider/instances -r demorealm -s alias=microsoft -s
providerId=microsoft -s enabled=true  -s 'config.useJwksUrl="true"' -s
config.clientId=MICROSOFT_APP_ID -s config.clientSecret=MICROSOFT_PASSWORD
```

配置 Stack Overflow 身份提供程序

1. 使用 **stackoverflow** 命令作为 **providerId**。

2. 提供配置属性 **client Id**、**clientSecret** 和 键。您可以在应用程序的 **Stack Apps OAuth** 页面中找到这些属性。如需更多信息，请参阅 **Stack Overflow 身份代理** 页面。

   例如：

```
$ kcadm.sh create identity-provider/instances -r demorealm -s alias=stackoverflow -s
providerId=stackoverflow -s enabled=true  -s 'config.useJwksUrl="true"' -s
config.clientId=STACKAPPS_CLIENT_ID -s
config.clientSecret=STACKAPPS_CLIENT_SECRET -s config.key=STACKAPPS_KEY
```

18.12. 存储供应商操作

配置 Kerberos 存储供应商

1. 对 组件 端点使用 **create** 命令。

2. 将 **realm id** 指定为 **parentId** 属性的值。

3. 将 **kerberos** 指定为 **providerId** 属性的值，将 **org.keycloak.storage.UserStorageProvider** 指定为 **providerType** 属性的值。

4. *例如：*

```
$ kcadm.sh create components -r demorealm -s parentId=demorealmId -s id=demokerberos
-s name=demokerberos -s providerId=kerberos -s
providerType=org.keycloak.storage.UserStorageProvider -s 'config.priority=["0"]' -s
'config.debug=["false"]' -s 'config.allowPasswordAuthentication=["true"]' -s 'config.editMode=
["UNSYNCED"]' -s 'config.updateProfileFirstLogin=["true"]' -s
```

```
'config.allowKerberosAuthentication=["true"]' -s 'config.kerberosRealm=["KEYCLOAK.ORG"]'
-s 'config.keyTab=["http.keytab"]' -s 'config.serverPrincipal=
["HTTP/localhost@KEYCLOAK.ORG"]' -s 'config.cachePolicy=["DEFAULT"]'
```

配置 LDAP 用户存储供应商

1. 对 组件 端点使用 **create** 命令。

2. 指定 **ldap** 作为 **providerId** 属性的值，将 **org.keycloak.storage.UserStorageProvider** 作为 **providerType** 属性的值。

3. 提供 **realm ID** 作为 **parentId** 属性的值。

4. 使用以下示例创建 **Kerberos** 集成的 **LDAP** 供应商。

```
$ kcadm.sh create components -r demorealm -s name=kerberos-ldap-provider -s
providerId=ldap -s providerType=org.keycloak.storage.UserStorageProvider -s
parentId=3d9c572b-8f33-483f-98a6-8bb421667867  -s 'config.priority=["1"]' -s
'config.fullSyncPeriod=["-1"]' -s 'config.changedSyncPeriod=["-1"]' -s 'config.cachePolicy=
["DEFAULT"]' -s config.evictionDay=[] -s config.evictionHour=[] -s config.evictionMinute=[] -s
config.maxLifespan=[] -s 'config.batchSizeForSync=["1000"]' -s 'config.editMode=
["WRITABLE"]' -s 'config.syncRegistrations=["false"]' -s 'config.vendor=["other"]' -s
'config.usernameLDAPAttribute=["uid"]' -s 'config.rdnLDAPAttribute=["uid"]' -s
'config.uuidLDAPAttribute=["entryUUID"]' -s 'config.userObjectClasses=["inetOrgPerson,
organizationalPerson"]' -s 'config.connectionUrl=["ldap://localhost:10389"]'  -s
'config.usersDn=["ou=People,dc=keycloak,dc=org"]' -s 'config.authType=["simple"]' -s
'config.bindDn=["uid=admin,ou=system"]' -s 'config.bindCredential=["secret"]' -s
'config.searchScope=["1"]' -s 'config.useTruststoreSpi=["ldapsOnly"]' -s
'config.connectionPooling=["true"]' -s 'config.pagination=["true"]' -s
'config.allowKerberosAuthentication=["true"]' -s 'config.serverPrincipal=
["HTTP/localhost@KEYCLOAK.ORG"]' -s 'config.keyTab=["http.keytab"]' -s
'config.kerberosRealm=["KEYCLOAK.ORG"]' -s 'config.debug=["true"]' -s
'config.useKerberosForPasswordAuthentication=["true"]'
```

删除用户存储供应商实例

1. 使用存储供应商实例的 **id** 属性编写端点 **URI**，如 组件/**ID**。

2. 针对此端点运行 **delete** 命令。

   例如：

```
$ kcadm.sh delete components/3d9c572b-8f33-483f-98a6-8bb421667867 -r demorealm
```

-

为特定用户存储供应商触发所有用户的同步

1. *使用存储提供程序的 id 属性编写端点 URI，如 user-storage/ID_OF_USER_STORAGE_INSTANCE/sync。*

2. *添加 action=triggerFullSync 查询参数。*

3. *运行 create 命令。*

   例如：

```
$ kcadm.sh create user-storage/b7c63d02-b62a-4fc1-977c-947d6a09e1ea/sync?
action=triggerFullSync
```

为特定用户存储供应商触发更改用户的同步

1. *使用存储提供程序的 id 属性编写端点 URI，如 user-storage/ID_OF_USER_STORAGE_INSTANCE/sync。*

2. *添加 action=triggerChangedUsersSync 查询参数。*

3. *运行 create 命令。*

   例如：

```
$ kcadm.sh create user-storage/b7c63d02-b62a-4fc1-977c-947d6a09e1ea/sync?
action=triggerChangedUsersSync
```

测试 LDAP 用户存储连接

1. *在 testLDAPConnection 端点上运行 get 命令。*

2. *提供查询参数 bindCredential,bindDn,connectionUrl，并使用TrustStoreSpi。*

3. 将 action query 参数设置为 testConnection。

例如：

```
$ kcadm.sh create testLDAPConnection -s action=testConnection -s bindCredential=secret -s
bindDn=uid=admin,ou=system -s connectionUrl=ldap://localhost:10389 -s
useTruststoreSpi=ldapsOnly
```

测试 LDAP 用户身份验证

1. 在 testLDAPConnection 端点上运行 get 命令。

2. 提供查询参数 bindCredential,bindDn,connectionUrl，并使用TruststoreSpi。

3. 将 action 查询参数设置为 testAuthentication。

例如：

```
$ kcadm.sh create testLDAPConnection -s action=testAuthentication -s
bindCredential=secret -s bindDn=uid=admin,ou=system -s
connectionUrl=ldap://localhost:10389 -s useTruststoreSpi=ldapsOnly
```

18.13. 添加映射程序

添加硬编码角色 LDAP 映射器

1. 在 组件 端点上运行 create 命令。

2. 将 providerType 属性设置为 org.keycloak.storage.ldap.mappers.LDAPStorageMapper。

3. 将 parentId 属性设置为 LDAP 提供程序实例的 ID。

4. 将 providerId 属性设置为 hardcoded-ldap-role-mapper。确保提供 角色 配置参数值。

例如：

```
$ kcadm.sh create components -r demorealm -s name=hardcoded-ldap-role-mapper -s
providerId=hardcoded-ldap-role-mapper -s
providerType=org.keycloak.storage.ldap.mappers.LDAPStorageMapper -s
parentId=b7c63d02-b62a-4fc1-977c-947d6a09e1ea -s 'config.role=["realm-
management.create-client"]'
```

添加 MS Active Directory 映射器

1. 在 组件 端点上运行 **create** 命令。

2. 将 **providerType** 属性设置为 **org.keycloak.storage.ldap.mappers.LDAPStorageMapper**。

3. 将 **parentId** 属性设置为 **LDAP** 提供程序实例的 **ID**。

4. 将 **providerId** 属性设置为 **msad-user-account-control-mapper**。

   例如：

```
$ kcadm.sh create components -r demorealm -s name=msad-user-account-control-mapper -
s providerId=msad-user-account-control-mapper -s
providerType=org.keycloak.storage.ldap.mappers.LDAPStorageMapper -s
parentId=b7c63d02-b62a-4fc1-977c-947d6a09e1ea
```

添加用户属性 LDAP 映射器

1. 在 组件 端点上运行 **create** 命令。

2. 将 **providerType** 属性设置为 **org.keycloak.storage.ldap.mappers.LDAPStorageMapper**。

3. 将 **parentId** 属性设置为 **LDAP** 提供程序实例的 **ID**。

4. 将 **providerId** 属性设置为 **user-attribute-ldap-mapper**。

   例如：

```
$ kcadm.sh create components -r demorealm -s name=user-attribute-ldap-mapper -s
providerId=user-attribute-ldap-mapper -s
providerType=org.keycloak.storage.ldap.mappers.LDAPStorageMapper -s
parentId=b7c63d02-b62a-4fc1-977c-947d6a09e1ea -s 'config."user.model.attribute"=
["email"]' -s 'config."ldap.attribute"=["mail"]' -s 'config."read.only"=["false"]' -s
'config."always.read.value.from.ldap"=["false"]' -s 'config."is.mandatory.in.ldap"=["false"]'
```

添加组 LDAP 映射器

1. 在 组件 端点上运行 **create** 命令。

2. 将 **providerType** 属性设置为 **org.keycloak.storage.ldap.mappers.LDAPStorageMapper**。

3. 将 **parentId** 属性设置为 **LDAP 提供程序实例的 ID**。

4. 将 **providerId** 属性设置为 **group-ldap-mapper**。

   例如：

```
$ kcadm.sh create components -r demorealm -s name=group-ldap-mapper -s
providerId=group-ldap-mapper -s
providerType=org.keycloak.storage.ldap.mappers.LDAPStorageMapper -s
parentId=b7c63d02-b62a-4fc1-977c-947d6a09e1ea -s 'config."groups.dn"=[]' -s
'config."group.name.ldap.attribute"=["cn"]' -s 'config."group.object.classes"=
["groupOfNames"]' -s 'config."preserve.group.inheritance"=["true"]' -s
'config."membership.ldap.attribute"=["member"]' -s 'config."membership.attribute.type"=
["DN"]' -s 'config."groups.ldap.filter"=[]' -s 'config.mode=["LDAP_ONLY"]' -s
'config."user.roles.retrieve.strategy"=["LOAD_GROUPS_BY_MEMBER_ATTRIBUTE"]' -s
'config."mapped.group.attributes"=["admins-group"]' -s
'config."drop.non.existing.groups.during.sync"=["false"]' -s 'config.roles=["admins"]' -s
'config.groups=["admins-group"]' -s 'config.group=[]' -s 'config.preserve=["true"]' -s
'config.membership=["member"]'
```

添加完整名称 LDAP 映射器

1. 在 组件 端点上运行 **create** 命令。

2. 将 **providerType** 属性设置为 **org.keycloak.storage.ldap.mappers.LDAPStorageMapper**。

3. 将 parentId 属性设置为 LDAP 提供程序实例的 ID。

4. 将 providerId 属性设置为 full-name-ldap-mapper。

例如：

```
$ kcadm.sh create components -r demorealm -s name=full-name-ldap-mapper -s
providerId=full-name-ldap-mapper -s
providerType=org.keycloak.storage.ldap.mappers.LDAPStorageMapper -s
parentId=b7c63d02-b62a-4fc1-977c-947d6a09e1ea -s 'config."ldap.full.name.attribute"=
["cn"]' -s 'config."read.only"=["false"]' -s 'config."write.only"=["true"]'
```

18.14. 身份验证操作

设置密码策略

1. 将 realm 的 passwordPolicy 属性设置为 enumeration 表达式，其中包含特定的策略提供程序 ID 和可选配置。

2. 使用以下示例将密码策略设置为默认值。默认值包括：

- 27,500 个哈希迭代
- 至少一个特殊字符
- 至少一个大写字母字符
- 至少一个数字字符
- 不等于 用户的用户名
- 至少 8 个字符

```
$ kcadm.sh update realms/demorealm -s 'passwordPolicy="hashIterations and
specialChars and upperCase and digits and notUsername and length"'
```

3. 要使用与默认值不同的值，请在括号中传递配置。

4. 使用以下示例将密码策略设置为：

- 25,000 个哈希迭代
- 至少两个特殊字符
- 至少两个大写字符
- 至少两个小写字母字符
- 至少两个数字
- 至少 9 个字符
- 不等于 用户的用户名
- 不对至少四个更改重复

```
$ kcadm.sh update realms/demorealm -s 'passwordPolicy="hashIterations(25000) and specialChars(2) and upperCase(2) and lowerCase(2) and digits(2) and length(9) and notUsername and passwordHistory(4)"'
```

获取当前密码策略

您可以通过过滤除 passwordPolicy 属性以外的所有输出来获取当前的域配置。

例如，显示 demorealm 的 passwordPolicy。

```
$ kcadm.sh get realms/demorealm --fields passwordPolicy
```

列出验证流

在 authentication/flows 端点上运行 get 命令。

例如：

```
$ kcadm.sh get authentication/flows -r demorealm
```

获取特定的身份验证流程

在 authentication/flows/FLOW_ID 端点上运行 get 命令。

例如：

```
$ kcadm.sh get authentication/flows/febfd772-e1a1-42fb-b8ae-00c0566fafb8 -r demorealm
```

列出流的执行

在 authentication/flows/FLOW_ALIAS/executions 端点上运行 get 命令。

例如：

```
$ kcadm.sh get authentication/flows/Copy%20of%20browser/executions -r demorealm
```

在执行中添加配置

1. *获取流程的执行。*
2. *记下流的 ID。*
3. *对 authentication/executions/{executionId}/config 端点运行 create 命令。*

例如：

```
$ kcadm create "authentication/executions/a3147129-c402-4760-86d9-3f2345e401c7/config" -r
examplerealm -b '{"config":{"x509-cert-auth.mapping-source-selection":"Match SubjectDN using
regular expression","x509-cert-auth.regular-expression":"(.*?)(?:$)","x509-cert-auth.mapper-
selection":"Custom Attribute Mapper","x509-cert-auth.mapper-selection.user-attribute-
```

```
name":"usercertificate","x509-cert-auth.crl-checking-enabled":"","x509-cert-auth.crldp-checking-enabled":false,"x509-cert-auth.crl-relative-path":"crl.pem","x509-cert-auth.ocsp-checking-enabled":"","x509-cert-auth.ocsp-responder-uri":"","x509-cert-auth.keyusage":"","x509-cert-auth.extendedkeyusage":"","x509-cert-auth.confirmation-page-disallowed":""},"alias":"my_otp_config"}'
```

获取执行配置

1. 获取流程的执行。
2. 请注意其 **authenticationConfig** 属性，其中包含配置 **ID**。
3. 在 **authentication/config/ID** 端点上运行 **get** 命令。

例如：

```
$ kcadm get "authentication/config/dd91611a-d25c-421a-87e2-227c18421833" -r examplerealm
```

更新执行的配置

1. 获取流程的执行。
2. 获取流程的 **authenticationConfig** 属性。
3. 请注意属性中的配置 **ID**。
4. 在 **authentication/config/ID** 端点上运行 **update** 命令。

例如：

```
$ kcadm update "authentication/config/dd91611a-d25c-421a-87e2-227c18421833" -r examplerealm -b '{"id":"dd91611a-d25c-421a-87e2-227c18421833","alias":"my_otp_config","config":{"x509-cert-auth.extendedkeyusage":"","x509-cert-auth.mapper-selection.user-attribute-name":"usercertificate","x509-cert-auth.ocsp-responder-uri":"","x509-cert-auth.regular-expression":"(.*?)(?:$)","x509-cert-auth.crl-checking-enabled":"true","x509-cert-auth.confirmation-page-disallowed":"","x509-cert-auth.keyusage":"","x509-cert-auth.mapper-selection":"Custom Attribute
```

```
Mapper","x509-cert-auth.crl-relative-path":"crl.pem","x509-cert-auth.crldp-checking-enabled":"false","x509-cert-auth.mapping-source-selection":"Match SubjectDN using regular expression","x509-cert-auth.ocsp-checking-enabled":""}}'
```

删除执行的配置

1. *获取流程的执行。*
2. *获取 flows authenticationConfig 属性。*
3. *请注意属性中的配置 ID。*
4. *在 authentication/config/ID 端点上运行 delete 命令。*

例如：

```
$ kcadm delete "authentication/config/dd91611a-d25c-421a-87e2-227c18421833" -r examplerealm
```